Highlights in Theoretical Computer Science

Jin-Yi Cai*
jyc@cs.wisc.edu

Abstract

These are some brief lecture notes for the summer course at SHUFE. My purpose for these lectures is to introduce the students to some elegant ideas in the Theory of Computing. The assumption is that the students have had a very minimal prior exposure to the Theory of Computing. I hope the selected topics can provide the students with some *taste* for this fascinating subject beyond a standard introductory course. However, due to the short duration of the summer course, I cannot provide a more in-depth introduction.

These notes are quite rough and uneven; some discussions are more detailed and others are quite sketchy. No attempt is made to be complete, or fully representative. For lack of time I have not polished them; errors are my responsibility.

1 Some Efficient Algorithms

Efficient algorithms are the life blood of computer science. It is the focus of perhaps one half of Theoretical Computer Science (and probably the bigger half; the other half is complexity theory).

I will describe a couple of efficient algorithms.

1.1 Integer Multiplication

Consider the ancient problem of integer multiplication: Given two n bit integers x and y, we want to compute their product $x \cdot y$. The "school method" will multiply each bit of y to the whole of x, one bit at a time, and then add these quantities up with appropriate shifts. This algorithm takes roughly $O(n^2)$ many bit operations.

Instead, the *Divide & Conquer* strategy says that we partition x and y into roughly $\frac{n}{2}$ each:

$$x = x_1 2^{\frac{n}{2}} + x_2, \quad y = y_1 2^{\frac{n}{2}} + y_2.$$

Now we notice that

$$x \cdot y = (x_1 \cdot y_1)2^n + (x_1 \cdot y_2 + x_2 \cdot y_1)2^{\frac{n}{2}} + (x_2 \cdot y_2).$$

We can (recursively) multiply the four products $(x_1 \cdot y_1)$, $(x_1 \cdot y_2)$, $(x_2 \cdot y_1)$, and $(x_2 \cdot y_2)$. Then the time complexity of this algorithm is roughly $T(n) = 4T(\frac{n}{2})$. Solving this recurrence we get, alas, $O(n^2)$. With no gain at all!

*Department of Computer Sciences, University of Wisconsin-Madison.

However, *all is not lost.* We will (recursively) multiply $(x_1 \cdot y_1)$ and $(x_2 \cdot y_2)$. But the next step sounds crazy: We form the *sum*(!)

$$s_x = x_1 + x_2, \qquad \text{and} \qquad s_y = y_1 + y_2,$$

(knowing that integer sum is cheap, in linear time.) And then also (recursively) multiply $(s_x \cdot s_y)$. Note that

$$s_x \cdot s_y = x_1 \cdot y_1 + x_1 \cdot y_2 + x_2 \cdot y_1 + x_2 \cdot y_2,$$

and since we already have $(x_1 \cdot y_1)$ and $(x_2 \cdot y_2)$, we can get $(x_1 \cdot y_2 + x_2 \cdot y_1)$ by subtraction:

$$s_x \cdot s_y - (x_1 \cdot y_1 + x_2 \cdot y_2).$$

Thus we obtain $(x \cdot y)$ by doing three multiplications of $\frac{n}{2}$ -bit integers. This yields an algorithm that runs in time $T(n) \approx 3T(n/2) + O(n)$. Solving this recurrence we get $T(n) = O(n^{\log_2 3})$, which is about $O(n^{1.59})$.

Notice the "totally mangled quantities" $s_x = (x_1 + x_2)$ and $s_y = (y_1 + y_2)$, which turned out to be the key to this success. So, in reasoning about efficient algorithms, it is not a good idea (and unsound reasoning) to think that the only good moves are the intuitively natural moves.

1.2 Matrix Multiplication

Our second example for the *Divide & Conquer* strategy is matrix multiplication. Suppose X and Y are two $n \times n$ integer matrices. We want to compute $X \cdot Y$, where we count integer multiplications (and additions) as unit operations.

We write X and Y in block form of $\frac{n}{2}$ size:

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \qquad \text{and} \qquad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix}.$$

Then

$$X \cdot Y = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}.$$

If we naively apply Divide & Conquer, we would get

$$T(n) = 8T(\frac{n}{2}) + O(n^2),$$

which gives $O(n^3)$, with no gain from "the definitional algorithm".

Instead, according to the following Strassen algorithm, we form

$$\begin{aligned} P_1 &= A(F - H) \\ P_2 &= (A + B)H \\ P_3 &= (C + D)E \\ P_4 &= D(G - E) \\ P_5 &= (A + D)(E + H) \\ P_6 &= (B - D)(G + H) \\ P_7 &= (A - C)(E + F) \end{aligned}$$

and then

$$X \cdot Y = \begin{bmatrix} P_4 + P_5 + P_6 - P_2 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_7 - P_3 \end{bmatrix}.$$

This gives us a running time of

$$T(n) = 7T(\frac{n}{2}) + O(n^2),$$

which is $O(n^{\log_2 7})$, roughly $O(n^{2.81})$.

The world record currently is about $O(n^{2.3728639})$ (due to Coppersmith and Winograd). Some believe that it can be as low as $O(n^{2+\epsilon})$ for any $\epsilon > 0$. No lower bound of the above form is known.

1.3 Fast Fourier Transform

Our third example (exemplar) algorithm is *fast Fourier transform*. Consider two polynomials of degree (at most) d:

$$A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_d x^d \quad \text{and} \quad B(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_d x^d.$$

We want to compute their product

$$C(x) = A(x)B(x) = c_0 + c_1 x + a_2 x^2 + \ldots + c_{2d} x^{2d},$$

where $c_k = \sum_{i+j=k} a_i b_j = a_0 b_k + a_1 b_{k-1} + \ldots + a_k b_0$ (with a_j and $b_j = 0$ if $j > d$).

Key Observation: A polynomial of degree (at most) d is uniquely determined by its values at any $d+1$ distinct points.

$$\begin{bmatrix} A(x_0) \\ A(x_1) \\ A(x_2) \\ \vdots \\ A(x_d) \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^d \\ 1 & x_1 & x_1^2 & \ldots & x_1^d \\ 1 & x_2 & x_2^2 & \ldots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_d & x_d^2 & \ldots & x_d^d \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix}.$$

This is a linear system with a Vandermonde matrix, having determinant $\prod_{i<j}(x_j - x_i)$.

In other words: $(d+1)$ *unknown coefficients correspond to* $(d+1)$ *conditions.*

So a polynomial $f(x) = \sum_{i=0}^{d} a_i x^i$ is given either by the coefficients $a_0, a_1, a_2, \ldots, a_d$, or alternatively by the values $f(x_0), f(x_1), f(x_2), \ldots, f(x_d)$.

If we think of $A(x)$ and $B(x)$ as given by the values $A(x_i)$ and $B(x_i)$, then their product polynomial $C(x)$ is given by the values $A(x_i)B(x_i)$. This representation for polynomials is ideal for getting the product of two polynomials.

But this is useful for multiplying two polynomials only if we can get from one representation to the other quickly. This can be done very efficiently if we choose the points $x_0, x_1, x_2, \ldots, x_d$ very carefully.

Assume $d + 1 = n$ is a power of 2. If we pick

$$\pm x_0, \pm x_1, \ldots, \pm x_{n/2-1}$$

and collect the even-degree terms and odd-degree terms together

$$\begin{aligned} A_e(x) &= \sum_i a_{2i} x^{2i} = a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2}, \\ A_o(x) &= \sum_i a_{2i+1} x^{2i+1} = a_1 x^1 + a_3 x^3 + \ldots + a_{n-1} x^{n-1}, \end{aligned}$$

then $A_e(x)$ is a polynomial of x^2 of degree $n/2-1$, and

$$A_o(x) = x[a_1 + a_3x^2 + \ldots + a_{n-1}x^{n-2}],$$

which is a product of x with another polynomial of x^2 of degree $n/2-1$. Furthermore, for any polynomial $p(x^2)$ of x^2, clearly $p(x^2) = p((\pm x)^2)$. Thus if we choose $\pm x_0, \pm x_1, \ldots, \pm x_{n/2-1}$, there will be a lot of savings in computation.

To wit:

$$\begin{aligned} A(x_i) &= A_e(x_i^2) + x_i A_o(x_i^2) \\ A(-x_i) &= A_e(x_i^2) - x_i A_o(x_i^2) \end{aligned}$$

However, once we used $\pm x_i$, say ± 1, this trick cannot be repeated in real numbers. How to repeat this recursively? This naturally leads to the use of complex numbers, in particular roots of unity $e^{2\pi ki/n}$, for $k = 0, 1, 2, \ldots, n-1$. Denote by $\omega = e^{2\pi i/n}$ the nth primitive root of unity.

We will rewrite the Vandermonde matrix

$$M_n(\omega) = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^j & \omega^{2j} & \ldots & \omega^{j(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \ldots & \omega^{(n-1)(n-1)} \end{bmatrix},$$

by grouping even indexed rows and columns first, and the odd indexed rows and columns second. The linear system

$$M_n(\omega) \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} A(\omega^0) \\ A(\omega^1 \\ \vdots \\ A(\omega^{n-1}) \end{bmatrix}$$

becomes two linear systems on $M_{n/2}(\omega^2)$, where for $0 \le j < n/2$ we have the following equations: the value $A(\omega^j)$ is the jth row of

$$M_{n/2} \begin{bmatrix} a_0 \\ a_2 \\ \vdots \\ a_{n-2} \end{bmatrix} + \omega^j M_{n/2} \begin{bmatrix} a_1 \\ a_3 \\ \vdots \\ a_{n-1} \end{bmatrix},$$

and the value $A(\omega^{j+n/2})$ is the jth row of

$$M_{n/2} \begin{bmatrix} a_0 \\ a_2 \\ \vdots \\ a_{n-2} \end{bmatrix} - \omega^j M_{n/2} \begin{bmatrix} a_1 \\ a_3 \\ \vdots \\ a_{n-1} \end{bmatrix}.$$

Thus the size n problem of *discrete Fourier transform* of

$$(a_0, a_1, a_2, a_3, \ldots, a_{n-1}) \longrightarrow (A(\omega^0), A(\omega^1), A(\omega^2), A(\omega^3), \ldots, A(\omega^{n-1}))$$

is reduced to two subproblems of $n/2$ size each: Transforming $(a_0, a_2, \ldots, a_{n-2})$ and $(a_1, a_3, \ldots, a_{n-1})$ via $M_{n/2}(\omega^2)$. Note that $\omega^2 = e^{2\pi i/(n/2)}$, exactly the primitive root of order $n/2$ for the $n/2$ size problem.

This yields a complexity of $T(n) = 2T(n/2) + O(n) = O(n \log n)$.

Clearly the fast Fourier transform gives a super fast algorithm for integer multiplication.

Note that $F = \frac{1}{\sqrt{n}} M_n(\omega)$ is an Hermitian matrix: $\overline{F}^{\mathsf{T}} F = I_n$, where $\overline{F}_{j,k} = \overline{\omega^{jk}} = \omega^{-jk}$.

2 Some Theorems on Matchings and Coverings

There are several elegant theorems concerning matchings and coverings etc. These theorems are in some sense all equivalent. They have far reaching algorithmic consequences. I will describe some of these.

A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E, which are 2-element (unordered) subsets of V.

We say a graph G is bipartite if V can be partitioned into two disjoint sets $X \uplus Y$ such that all edges in E are between X and Y. We denote a bipartite graph as $G = (X, Y, E)$.

A vertex cover, or a covering, of G is a subset $C \subseteq V$ such that each edge in E contains at least one vertex of C (and the edge is said to be covered by that vertex). A matching of G is a subset $M \subseteq E$, such that no two edges in M share a vertex. A matching M is maximal if M is not properly contained in another matching M'. A matching M is maximum if $|M|$ is the largest among all matchings of G. A maximum matching is a maximal matching, but not vice versa. A perfect matching of G is a matching that matches every vertex. A perfect matching is a maximum matching. If a graph contains a perfect matching then a maximum matching is the same as a perfect matching; otherwise (when no perfect matching exists in a graph) a maximum matching still exists (but is not a perfect matching).

For any matching M of G, every vertex cover C must spend at least one vertex per each matching edge. Hence $|C| \geq |M|$. Thus

$$\min\{|C| : C \text{ is a vertex cover of } G\} \geq \max\{|M| : M \text{ is a matching of } G\}.$$

On the other hand, if M is a maximal matching of G, then there are no edges $e = \{u, v\} \in E$ such that both u and v are not matched by some edge in M (or else we could enlarge M by getting a larger matching $M' = M \cup \{e\}$.) Hence every edge $e = \{u, v\} \in E$ has at least one vertex matched by M. Therefore if we take all vertices in M, we get a vertex cover of G. Thus

$$\min\{|C|\} \leq 2 \max\{|M|\} \leq 2 \min\{|C|\}.$$

2.1 König's Theorem

In 1931, Dénes Kőnig * proved the following theorem:

*Dénes Kőnig was the son of Gyula Kőnig, also a well-known mathematician. His last name is spelled as Kőnig with the Hungarian double acute accent, while his theorem is usually written with a German umlaut, König. Dénes

Theorem 2.1. *For any bipartite graph, the maximum size of a matching is equal to the minimum size of a vertex cover.*

We already know that $\min\{|C|\} \geq \max\{|M|\}$, without assuming G is bipartite. The claim is that for any bipartite graph, if M is a maximum matching, then there is a vertex cover C of size at most $|M|$ (thus it must be of the same cardinality). The proof is constructive.

Let $G = (X, Y, E)$ be a bipartite graph, and suppose M is a maximum matching for G. To construct a vertex cover of size $|M|$, we let $U \subseteq X$ be the set of unmatched vertices on LHS. Now consider all possible *alternating paths* starting from vertices in U. A path is *alternating* with respect to M if it alternates between matching and non-matching edges. Let Z be the set of all vertices reachable from U by alternating paths. This includes all of U, and all neighbors of U in Y (all by necessarily non-matching edges, by the definition of U), and their neighbors in X by (uniquely determined, if any) matching edges, and their neighbors by non-matching edges, and so on. Let

$$K = (X - Z) \cup (Y \cap Z).$$

We claim that K is a vertex cover of G. Consider any edge $e = \{x, y\} \in E$. We show that if $x \in X \cap Z$ then $y \in Y \cap Z$. First suppose $e \in M$. Then it must be that $x \notin U$, since no matching edge can touch U. Since $x \in X \cap Z$, x must be reached by an alternating path (of positive even length) from U, and its last edge is in M. And so e *is* that matching edge (since there is a unique matching edge incident to x), thus $y \in Y \cap Z$, because the path that reached x passed through y. Next suppose $e \notin M$. Then by $x \in X \cap Z$, x is reachable by an alternating path (of length ≥ 0) from U. We can extend this path by e to form an alternating path to reach y. This extension is an alternating path because $e \notin M$, and either $x \in U$, or the alternating path (of positive length) that reaches x ends with a matching edge. Then, y is also reachable by an alternating path from U. Thus $y \in Y \cap Z$.

Now we claim that K is a minimum vertex cover. (This is where we will use the fact that M is a maximum matching; by the inequality $\max\{|M|\} \leq \min\{|C|\}$ this use is also necessary.) We note that every $y \in Y \cap Z$ must be matched by some edge in M. Otherwise, there will be an *augmenting path*, i.e., an alternating path that starts and ends both with non-matching edges. For any augmenting path, if we flip every edge along this path, between matching and non-matching, we could enlarge $|M|$, a contradiction. Also, trivially, every $x \in X - Z$ is matched by some edge in M, since all unmatched vertices in X are already in U, a subset of Z. Finally all matching edges

---

Kőnig attributed the idea of studying matchings in bipartite graphs to his father Gyula Kőnig. Dénes Kőnig is also known for the innocent looking lemma that says that every infinite tree either has a vertex of infinite degree or has an infinite path. (Try prove this!) His father Gyula Kőnig is remembered today mainly for his contributions to, and his opposition against, Cantor's set theory. In 1904, Gyula Kőnig announced he has disproved Cantor's continuum hypothesis. Unfortunately his proof depended on a theorem proved in the thesis of Felix Bernstein, but Bernstein's theorem was only valid in a more restricted sense than Bernstein claimed. Ernst Zermelo (of the axiomatized Zermelo-Fraenkel set theory) found the error and Gyula Kőnig withdrew his claim. Bernstein is known for the Schröder-Bernstein theorem that says that if there is an injection from A to B and from B to A, then there is a bijection between A and B. This is an essential ingredient in the notion of cardinality. Cantor's idea of diagonalization led to Turing's formulation of undecidability. Juris Hartmanis used the idea of Schröder-Bernstein theorem to show that all the usual NP-complete sets are polynomial time isomorphic, leading to the yet unproved Hartmanis Conjecture for NP. Cantor on the other hand had mixed ideas about Kőnig's contributions: At one point he said, "the positive contributions from Kőnig himself are well done." But at a later time he said, "What Kronecker and his pupils as well as Gordan have said against set theory, what Kőnig, Poincaré, and Borel have written against it, soon will be recognized by all as rubbish."

incident to $y \in Y \cap Z$ match vertices in $X \cap Z$. Thus all matching edges incident to $x \in X - Z$ and to $y \in Y \cap Z$ are distinct. Thus $|K| \leq |M|$. The proof is complete.

2.2 König-Egerváry Theorem

The *term rank* of a $(0,1)$-matrix is the largest number of 1's that can be chosen from the matrix such that no 2 selected 1's lie on the same line (row or column). A set S of rows and columns is a cover of a $(0,1)$-matrix if the matrix becomes a zero matrix after all the lines in S have been deleted. (In other words, the rows and columns in S contain all 1's.)

Theorem 2.2. *The term rank of a $(0,1)$-matrix is the cardinality of its smallest cover.*

If we represent a bipartite graph by its adjacency $(0,1)$-matrix, then clearly

1. Term rank is the same as the size of a maximum matching.
2. A cover of a $(0,1)$-matrix is a vertex cover of the bipartite graph it represents.

König-Egerváry Theorem applies to rectangular matrices (not necessarily square matrices).

2.3 Hall's Theorem

A Marriage Problem: Given a set of n boys and a set of n girls, let each boy make a list of the girls he is willing to marry. Is there a *perfect* marriage plan where every boy is married to a girl on his list (and every boy is married to exactly one girl and every girl is married to exactly one boy).

Hall's theorem says that such a perfect marriage plan exists if and only if for every subset B of boys, the union of the lists from B has cardinality at least $|B|$.

In terms of a bipartite graph representing the boys and girls and lists (of desirables) this is a problem about the existence of a perfect matching (this is why it's called matching). Hall's Theorem says

Theorem 2.3. *In a bipartite graph $G = (X, Y, E)$, with $|X| = n$ and $|Y| = m$, a matching exists that matxches all vertices in X iff for every $B \subseteq X$, the neighbor set $\Gamma(B) = \{y \in Y \mid (\exists x \in B), \{x, y\} \in E\}$ has cardinality $|\Gamma(B)| \geq |B|$.*

Corollary 2.4. *In a bipartite graph $G = (X, Y, E)$, with $|X| = |Y| = n$, a perfect matching exists iff for every $B \subseteq X$, the neighbor set $\Gamma(B) = \{y \in Y \mid (\exists x \in B), \{x, y\} \in E\}$ has cardinality $|\Gamma(B)| \geq |B|$.*

†

The same ideas can be applied to the so-called

Assignment Problem: Given a set of n employees and n tasks. Each employee fills out a list of the tasks he is be able to preform. Each task requires just one employee to preform and each employee can perform only one task. Then, can we assign each employee a task he is capable of, such that all tasks are fulfilled?

[†]This theorem is due to Philip Hall (1935), not to be confused with Marshall Hall, Jr. (no relation to Philip Hall) who was also a prominent mathematician. Further confusing the matter is the following. By examining Philip Hall's original proof carefully, Marshall Hall was able to prove a generalization, and went on to write a well known book *Combinatorial Theory*. Another confusion is that both Philip Hall and Marshall Hall made significant contributions to group theory, and Philip Hall's contribution to group theory was truly seminal. Marshall Hall was the Ph.D. advisor to Donald Knuth.

Hall's theorem says that such an assignment is possible iff for every subset of k employees, the union of their lists has cardinality at least k.

Later we will consider a weighted version of this Assignment Problem.

The proof of Hall's theorem goes as follows:

Clearly if a matching exists that matches all X, then for every subset $S \subseteq X$, its neighbor set $\Gamma(S)$ has cardinality at least $|S|$. So the condition in Hall's theorem is necessary.

Suppose the condition in Hall's theorem is satisfied. Take a maximum matching M. We wish to prove that M matches all X. Suppose not, and let $u \subseteq X$ be an unmatched vertex in the LHS. Consider the set of all vertices reachable by an alternating path from u. This set can be easily computed by a Breath-First-Search. Let

$$\begin{aligned} X_{2k} &= \{x \in X \mid x \text{ is reachable from } u \text{ by an alternating path of length} \leq 2k\}, \\ Y_{2k-1} &= \{y \in Y \mid y \text{ is reachable by from } u \text{ an alternating path of length} \leq 2k-1\}. \end{aligned}$$

Then $X_0 = \{u\}$. $Y_1 =$ its neighbor set $\Gamma(\{u\})$. We have $|\Gamma(\{u\})| \geq 1$. The connecting edges from $\{u\}$ to $\Gamma(\{u\})$ are all nonmatching edges, by the definition of U.

To gain some intuition, we can unravel this definition. Each vertex in $Y_1 = \Gamma(\{u\})$ must be incident to a matching edge, otherwise we can directly add an edge to M and enlarge M. So we can assume each vertex in Y_1 has a (unique) vertex in X matched by M. Since u is unmatched, the matching vertices to Y_1 are distinct from u. So now we have $|X_2| = 1 + |Y_1|$.

Inductively we have a subset $X_{2k} \subseteq X$ containing $\{u\}$, and a subset $Y_{2k-1} \subseteq Y$, where every $y \in Y_{2k-1}$ has a unique matching edge back to X_{2k}. So $|X_{2k}| = 1 + |Y_{2k-1}|$.

Now by the condition of Hall's Theorem, the neighbor set of X_{2k} has cardinality $\geq |X_{2k}| > |Y_{2k-1}|$. So $|Y_{2k+1}| > |Y_{2k-1}|$. Then the newly added vertices in $Y_{2k+1} - Y_{2k-1}$ must be matched by M, or else we could have enlarged $|M|$ by flipping the matching and non-matching edges on an augmenting path from u to this unmatched vertex. Since M is a maximum matching, this is impossible.

However, we have reached a contradiction, by showing that the sequence $|Y_1| < \ldots < |Y_{2k-1}| < |Y_{2k+1}|$ is an ever increasing sequence of integers (in a finite graph). The contradiction says really says that, whenever we start with some matching M in a graph that satisfies the condition of Hall's Theorem, the process described above will find an augmenting path to increase the size $|M|$, and the process terminates with a matching that matches all vertices of X.

2.4 Hall's Theorem implies König's Theorem

As König's Theorem and König-Egerváry Theorem are obviously equivalent, we will show that Hall's Theorem implies König-Egerváry Theorem.

Let B be an $n \times m$ $(0,1)$-matrix. Let $p =$ term rank of B, and $q =$ minimum cover size of B. By the connection between maximum matching and minimum vertex cover, we have $p \leq q$. We just need to prove $p \geq q$.

Let's permute the rows and columns of B so that the first r rows and first s columns cover all 1's in B, where $q = r + s$. Now consider the submatrix B' of B consisting of the first r rows and last $m - s$ columns. For every row $1 \leq i \leq r$ of B', consider the set of column indices where the entry is 1:

$$A_i = \{j | s < j \leq m, B_{i,j} \neq 0\}.$$

Each $A_i \neq \emptyset$, for otherwise we can remove ith row and still cover all 1's. In fact, for any subset $I \subseteq \{1, \ldots, r\}$, the cardinality $|\bigcup_{i \in I} A_i|$ must be at least $|I|$, for otherwise we could replace those rows in I by fewer columns to cover all 1's. Hence B' satisfies the condition of Hall's Theorem, and therefore there is a matching of size r consisting of 1's in B'.

Similarly we can define B'' to be the submatrix of B consisting of the first s columns and the last $n-r$ rows. By the same argument (every subset of columns of B'' has as many rows with some 1's) we can get a matching of size s consisting of 1's in B''. Together they prove the theorem.

2.5 The Hungarian Algorithm

Now we consider the Assignment Problem with weights. Alternatively, in terms of matchings, we are given an $n \times n$ square matrix A of nonnegative integers. We want to find a perfect matching that has the maximum weight, which is defined to be the sum of the matching edges weights.

Note that essentially the same question can be asked for minimization (e.g., to minimize cost), and can be solved by an easy reduction. Just replace each entry $A_{i,j}$ by $N - A_{i,j}$ for some large number N (e.g., the maximum of all $A_{i,j}$). The same algorithm works. There is no essential difference.

We think of the given matrix A as the weighted adjacency matrix of the complete bipartite graph with n vertices on each side, and the entry $A_{i,j}$ is the weight of the edge (i, j).

To find a maximum weighted perfect matching in a weighted complete bipartite graph $K_{n,n}$ is just as general as finding a maximum weighted matching in a weighted (not necessarily complete) bipartite graph with n vertices on each side, since we can add 0-weight edges. For rectangular matrices (i.e., for bipartite graphs of unequal sides) we can also easily append dummy vertices (and 0-weight edges).

So our problem can also be described as

Maximum weight bipartite matching: Given a bipartite weighted graph $G = (X, Y, E, w)$, where $w : E \to \mathbb{R}$, find a maximum weight matching, i.e., a matching M of G such that $\sum_{e \in M} w(e)$ is maximized.

(The weights need not be nonnegative, although we can increase every weight by an equal amount, so that without loss of generality we can assume they are nonnegative.)

Start with any matching M, an alternating tree is a tree rooted at some unmatched (free) vertex v in which every path from v is an alternating path.

A *feasible labeling* (or a *potential*) is a mapping $\ell : V = X \cup Y \to \mathbb{R}$, such that: For every $x \in X$ and $y \in Y$,

$$\ell(x) + \ell(y) \geq w(x, y).$$

Given a feasible labeling ℓ, an edge (x, y) is *tight* if $\ell(x) + \ell(y) = w(x, y)$. The *equality graph* (with respect to ℓ) is $G_\ell = (X, Y, E_\ell)$ consisting of all tight edges:

$$E_\ell = \{(x, y) \in X \times Y \mid \ell(x) + \ell(y) = w(x, y)\}.$$

Theorem 2.5. *If ℓ is a feasible labeling, and M is a perfect matching in G_ℓ, then M is a max-weight matching.*

This is due to Kuhn and Munkres. Because the ideas behind it are closely related to König's and Egerváry's theorems, the algorithm that follows this theorem is called the Hungarian algorithm.

To prove this theorem of Kuhn and Munkres, consider any perfect matching M' in G (not necessarily confined in G_ℓ).

$$\sum_{e\in M'} w(e) \le \sum_{e=(x,y)\in M'} [\ell(x)+\ell(y)] = \sum_{x\in X} \ell(x) + \sum_{y\in Y} \ell(y) = \sum_{v\in V} \ell(v).$$

since every vertex is contained in exactly one edge of M'.

On the other hand, for M, we have equality, because M is in G_ℓ where every edge is tight. Thus the total weight of M is at least as large as any perfect matching M'. This proves the theorem.

This relation: For every perfect matching M' in G and every feasible labeling ℓ on V we have

$$\sum_{e\in M'} w(e) \le \sum_{v\in V} \ell(v),$$

is the quintessential min-max primal-dual relation in, e.g., linear programming.

Now the algorithm. The goal is to find a perfect matching in G_ℓ. The outline of the idea is as follows: Start with any feasible labeling ℓ and any matching M in G_ℓ.

While M is not a perfect matching, repeat the following:

1. Find an augmenting path for M in G_ℓ. This increases $|M|$.
2. If no augmenting path exists, then improve ℓ.

We can define an initial ℓ by assigning for every $x \in X$, $\ell(x) = \max_{y\in Y} w(x,y)$, and for every $y \in Y$, $\ell(y) = 0$. We can take the initial $M = \emptyset$. We maintain the invariant that all the edges of M are tight. (Thus M is a matching in G_ℓ.)

Suppose M is not yet a perfect matching. Let $R_X \subseteq X$ and $R_Y \subseteq Y$ be the unmatched vertices. Let Z be the set of vertices reachable by *alternating* paths from R_X by edges in G_ℓ. (Thus edges used from X to Y are tight, and from Y to X are not only tight but also in M.) This can be computed by breadth-first search. If $R_Y \cap Z$ is nonempty, then we found an augmenting path. Thus we can modify M so that $|M|$ is increased by 1. If $R_Y \cap Z$ is empty, then let

$$\Delta_\ell = \min\{\ell(x)+\ell(y)-w(x,y) \mid x \in X\cap Z, y \in Y - Z\}.$$

This is the minimum (but positive) slack from $X \cap Z$ to $Y - Z$. Notice that this is positive because none of those edges are tight (if such an edge were tight, that y would be in Z.) Now modify ℓ to ℓ' by decreasing on each $x \in X \cap Z$ by Δ_ℓ and increasing on each $y \in Y \cap Z$ by the same Δ_ℓ.

We claim the updated ℓ' is also a potential: Any edge between $X \cap Z$ and $Y \cap Z$ is clearly OK since $\ell(x)+\ell(y)$ is unchanged. Any edge between $X - Z$ and $Y - Z$ is unaffected. Any edge between $X - Z$ and $Y \cap Z$ is OK, since only the values $\ell(y)$ are increased. The only worry is for edges between $X \cap Z$ and $Y - Z$; but Δ_ℓ is defined to be so that the dominance $w(x,y) \le \ell'(x) + \ell'(y)$ still holds.

The new equality graph $G_{\ell'}$ still contains M. This is because M is in G_ℓ. If $e = (x,y) \in M$ has $y \in Y \cap Z$, then it is between $X \cap Z$ and $Y \cap Z$ (the alternating path comes back from y to x). So e remains tight in $G_{\ell'}$. If $e \in M$ has $y \in Y - Z$ then it must be that $x \in X - Z$; otherwise $x \in X \cap Z$ and e is a matching edge implies that $x \notin R_X$ and x is taken into Z because the unique matching edge e brings it in, but then y must have been already in Z. And so e is between $X - Z$ and $Y - Z$ and then it is unaffected by the change from ℓ to ℓ'.

But with respect to ℓ' there is at least one more reachable vertex in Y, as all edges between $X \cap Z$ and $Y \cap Z$ remain tight under ℓ'. Thus Z increases. So after at most $|V|$ steps we must

increase $|M|$. This gives a polynomial time algorithm. Naive estimate gives a running time of $O(n^4)$. A more careful analysis gives $O(n^3)$.

The Hungarian Method

1. Generate initial labelling ℓ and matching M in G_ℓ.
2. If M is perfect, stop. Otherwise pick a free vertex $x_0 \in X$. Set $S = \{x_0\}$ and $T = \emptyset$.
3. If $\Gamma_{G_\ell}(S) = T$, update ℓ (this forces the updated $\Gamma_{G_\ell}(S) \neq T$.)

$$\begin{aligned} \Delta_\ell &= \min_{x \in S, y \in Y - T} \{\ell(x) + \ell(y) - w(x,y)\} \\ \ell'(v) &= \begin{cases} \ell(v) - \Delta_\ell & \text{if } v \in S \\ \ell(v) + \Delta_\ell & \text{if } v \in T \\ \ell(v) & \text{otherwise} \end{cases} \end{aligned}$$

4. If $\Gamma_{G_\ell}(S) \neq T$, pick $y \in \Gamma_{G_\ell}(S) - T$.
   - If y unmatched, then we found an augmenting path from x_0 to y. Augment M by flipping all edges along M. Goto 2.
   - If y is matched, say to x, then extend the alternating tree: $S := S \cup \{x\}, T := T \cup \{y\}$. Goto 3.

Here is another way the Hungarian Algorithm is often presented. We present a minimization version.

1. Define the matrix of edge costs.

$$\begin{pmatrix} 10 & 19 & 8 & 15 & 19 \\ 10 & 18 & 7 & 17 & 19 \\ 13 & 16 & 9 & 14 & 19 \\ 12 & 19 & 8 & 18 & 19 \\ 14 & 17 & 10 & 19 & 19 \end{pmatrix}$$

2. Reduce the rows by subtracting the minimum value of each row from that row.

$$\begin{pmatrix} 2 & 11 & 0 & 7 & 11 \\ 3 & 11 & 0 & 10 & 12 \\ 4 & 7 & 0 & 5 & 10 \\ 4 & 11 & 0 & 10 & 11 \\ 4 & 7 & 0 & 9 & 9 \end{pmatrix}$$

3. If there are columns without a zero, reduce the columns by subtracting the minimum value of each column from that column.

$$\begin{pmatrix} 0 & 4 & 0 & 2 & 2 \\ 1 & 4 & 0 & 5 & 3 \\ 2 & 0 & 0 & 0 & 1 \\ 2 & 4 & 0 & 5 & 2 \\ 2 & 0 & 0 & 4 & 0 \end{pmatrix}$$

4. Cover the zero elements with the minimum number of lines. (If the number of lines is equal to the number of rows then go to step 9.)

$$\left(\begin{array}{c|c|c|c|c} \cancel{0} & \cancel{4} & \cancel{0} & \cancel{2} & \cancel{2} \\ 1 & 4 & 0 & 5 & 3 \\ \cancel{2} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{1} \\ 2 & 4 & 0 & 5 & 2 \\ \cancel{2} & \cancel{0} & \cancel{0} & \cancel{4} & \cancel{0} \end{array}\right)$$

(This should remind you of König's Theorem.)

5. Add the minimum uncovered element to every covered element. If an element is covered twice, add the minimum element to it twice.

$$\begin{pmatrix} 1 & 5 & 2 & 3 & 2 \\ 1 & 4 & 1 & 5 & 3 \\ 3 & 1 & 2 & 1 & 2 \\ 2 & 4 & 1 & 5 & 2 \\ 3 & 1 & 2 & 5 & 1 \end{pmatrix}$$

6. Subtract that minimum element from every element in the matrix.

$$\begin{pmatrix} 0 & 4 & 1 & 2 & 2 \\ 0 & 3 & 0 & 4 & 2 \\ 2 & 0 & 1 & 0 & 1 \\ 1 & 3 & 0 & 4 & 1 \\ 2 & 0 & 1 & 4 & 0 \end{pmatrix}$$

The effect of these two steps is: For the un-stricken out submatrix, we subtracted the minimum. For the submatrix in the intersection of rows and columns both stricken out, we added that minimum. For the rest no change is done:

$$\left(\begin{array}{c|c|c|c|c} \cancel{0} & \cancel{4} & \cancel{0} & \cancel{2} & \cancel{2} \\ 1 & 4 & 0 & 5 & 3 \\ \cancel{2} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{1} \\ 2 & 4 & 0 & 5 & 2 \\ \cancel{2} & \cancel{0} & \cancel{0} & \cancel{4} & \cancel{0} \end{array}\right)$$

$$\begin{pmatrix} 1 & 4 & 5 & 3 \\ 2 & 4 & 5 & 2 \end{pmatrix} \quad \to \quad \begin{pmatrix} 0 & 3 & 4 & 2 \\ 1 & 3 & 4 & 1 \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \to \quad \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

7. Cover the zero elements again. If the number of lines covering the zero elements is not equal to the number of rows, return to step 5.

$$\left(\begin{array}{|ccccc} 0 & 4 & 1 & 2 & 2 \\ 0 & 3 & 0 & 4 & 2 \\ \hline 2 & 0 & 1 & 0 & 1 \\ 1 & 3 & 0 & 4 & 1 \\ \hline 2 & 0 & 1 & 4 & 0 \end{array}\right)$$

(This example needs to be reduced once more.)

Project: Explain the relationship between these two expositions of the Hungarian Algorithm.

3 Randomization in Algorithms

Randomness, both as a proof technique as well as a computational resource, has a significant role in the modern theory of algorithm and complexity.

3.1 Basic Probability

It is perhaps one of the primal experiences, much like the primal experiences with arithmetic quantity or geometric shape, that people have come to "know" randomness. Any systematic exploration was started much later, starting with Pascal, Fermat and Laplace. But it is always a nettlesome question as to what exactly is "randomness", and what exactly is "probability". I don't think this question has ever been truly satisfactorily answered, despite a lot of work on this topic. Perhaps there is no one single answer. However, brushing aside the "nature" of what is "probability", modern mathematics basically took the following perspective, after Kolmogorov. We start with an arbitrary measure space $(\Omega, \mathbf{E}, \mu)$, where Ω is some underlying set, equipped with a measure function $\mu : \mathbf{E} \to \mathbf{R}_+$, where $\mathbf{E}$ is a collection of subsets of Ω called a σ-algebra satisfying certain closure properties, and (being a probability measure) $\mu(\Omega) = 1$. (Being a measure, μ must satisfy σ-additivity, namely for disjoint countable family $\{S_i\} \subseteq \mathbf{E}$, $\mu(\bigcup_{i=1}^{\infty} S_i) = \sum_{i=1}^{\infty} \mu(S_i)$. $\mathbf{E}$ is a σ-algebra if $\Omega \in \mathbf{E}$, and $\mathbf{E}$ is closed under countable union, intersection, and complement. We will not discuss further the properties of a σ-algebra. For most of what we do, the set Ω will be finite, and $\mathbf{E}$ consists of all subsets of Ω.)

The Kolmogorov foundation of probability theory is a brilliant device to refocus the "theory of probability" as an internal mathematical subject, excluding all issues having to do with what is "randomness" as we experience it in the external world. It essentially gets rid of the issue of any reference to how this "probability theory" is to be related to the every day (or not so every day) notion of "chance" in the external world. A gambler (or any user of probability theory) has to decide what is an appropriate probability space to presume, as a mathematical model for his particular situation at hand. This modeling is external to any mathematical theory of probability. The mathematics of "probability theory" in the sense of Kolmogorov only helps to "calculate" the outcomes once the basic probability of events have been assigned. Thus, in Kolmogorov's probability theory, there is no meaning of a "fair coin"; instead, one merely sets up a suitable probability space, such as $\{H, T\}$, and postulate that $\mu(H) = \mu(T) = 1/2$. It is murkier what a quantum physicist really means when he talks about the "probability" of observing this or that quantum state.

What if we want "two independent fair coin flips"? In Kolmogorov's probability theory, we can set up a product probability space, $\{H,T\}^2$, and assign $\mu(HH) = \mu(HT) = \mu(TH) = \mu(TT) = 1/4$. Note that in this 4-point space, there is, strictly speaking, no notion of time.

While the Kolmogorov foundation is fine, it does seem to be somewhat sterile, and lacking certain intuition of "probabilistic thinking". For one thing, we do like to have a primitive notion of an independent coin flip. Moreover, if after several independent fair coin flips, we decided to have another one. This should not disturb any previous probability calculation. Instead in Kolmogorov's framework we must re-constitute a new probability space all over again. Technically all previous probability calculations must be carried out now in this new probability space. Of course they take the same values, but conceptually they now take place in a different probability space. This, I find unnatural and unsatisfactory. Frequently in an algorithm we want to be able to flip more coins as we go along. And this notion of time step is natural and intuitively helpful. It is somewhat unnatural to suppose we must have a super probability space in place a priori, which in its very definition has a built-in structure of how many coin flips we can have. (There are ways around this in Kolmogorov's framework, but they all seem contrived and unnatural.)

Luckily, for almost everything we discuss here, it will be over some finite (or at most countably infinite and recursive) space. Most of the time it will just be over some $\{0,1\}^n$, or something similar. There will be no philosophical difficulties by taking a naive approach to the concept of probability, but one which does admit a primitive notion of a new independent bit. In principle we can enumerate all the basic events and assign them "atomic probabilities" that add to one. Therefore we will take the following point of view. We will assume whenever we need we can have an independent additional coin flip (which may not be a fair coin). We will operate at a more intuitive level of "probability", as if we had a definite meaning of a physical "randomness". Thus we can say, for example, perform the following random trials independently for certain number of times with certain probability. We take this approach with the understanding that, whenever any potential difficulty should arise, we immediately retreat back to the safe cocoon of Kolmogorov foundation of measure space, and effectively say that: the only meaningful thing is what's happening in the following measure space; any implication to the "outside world" (such as what exactly does it mean by a random step) is not the responsibility of our probability analysis and any conclusions thereof.

3.1.1 Markov's Inequality

The first inequality to consider is Markov's Inequality. It deals with any random variable that takes only nonnegative values and the estimate is in terms of its expectation. Technically we need to assume it has a finite expectation, however, one can apply it with abandon, for when the nonnegative random variable has infinite expectation the estimate is trivially true.

Theorem 3.1. *Let X be a random variable such that $X \geq 0$ and $\mathrm{E}[X] < \infty$. For all $a > 0$,*

$$\Pr[X \geq a] \quad \leq \quad \frac{\mathrm{E}[X]}{a}$$

The expectation $\mathrm{E}[X]$ is defined to be $\int_\Omega X d\mu$. This theorem gives us a bound on the probability that X takes on a value that is *greater* than its expected value by a given amount. We give a quick proof of the theorem:

Proof. Let I be an indicator variable for the event $X \geq a$. That is,

$$I = \begin{cases} 1 & \text{if } X \geq a \\ 0 & \text{if } X < a \end{cases}$$

Note that

$$I = \begin{cases} \leq \frac{X}{a} & \text{if } X \geq a \\ = 0 & \text{if } X < a \end{cases} \leq \frac{X}{a}.$$

Clearly, $\mathrm{E}[I] = \Pr[X \geq a]$. Then,

$$\Pr[X \geq a] = \int_\Omega I d\mu = \int_{X \geq a} 1 d\mu \leq \int_{X \geq a} \frac{X}{a} d\mu \leq \int_\Omega \frac{X}{a} d\mu = \frac{\mathrm{E}[X]}{a}.$$

□

3.1.2 Chebyshev Inequality

Theorem 3.2. *For any random variable X with finite $\mathrm{E}[X]$ and $\mathrm{Var}(X)$, and let $a > 0$,*

$$\Pr[|X - \mathrm{E}[X]| \geq a] \leq \frac{\mathrm{Var}(X)}{a^2}$$

Here the variance $\mathrm{Var}(X)$ is defined as $\mathrm{E}\left[(X - \mathrm{E}[X])^2\right] = \mathrm{E}\left[X^2\right] - \mathrm{E}[X]^2$, and technically we must assume it is finite. However, the above estimate is (trivially) true even if it is infinite.

Proof. Define a random variable $Y = (X - \mathrm{E}[X])^2$. Then,

$$\begin{aligned} \Pr[|X - \mathrm{E}[X]| \geq a] &= \Pr[Y \geq a^2] \\ &\leq \frac{\mathrm{E}[Y]}{a^2} \\ &= \frac{\mathrm{Var}(X)}{a^2} \end{aligned}$$

In the above derivation, the inequality is obtained using the Markov's inequality. □

3.1.3 Chernoff Bound

It may appear that Markov's inequality is pretty weak. However it is very applicable in many situations, mainly because it only assumes that the random variable under consideration is non-negative and does not assume anything about its distribution. Judiciously applied it can yield remarkably sharp bounds. In this section, we use it to prove the Chernoff bound, which is a powerful inequality dealing with sums of independent random variables.

Before we give the statement of the Chernoff bound, we introduce the variables it will use. Let $\{X_1, X_2, \cdots, X_n\}$ be a set of n independent, identically distributed random variables such that each X_i has

$$\Pr[X_i = 1] = \Pr[X_i = -1] = \frac{1}{2}.$$

Let S_n be their sum: $S_n = \sum_{i=1}^n X_i$.

Theorem 3.3. *For all n, and for all $\Delta > 0$,*

$$\Pr[S_n \geq \Delta] \quad < \quad e^{-\Delta^2/2n}$$

Before proving this theorem, we note that it can be restated as

$$\Pr[S_n \geq \epsilon \cdot n] \quad \leq \quad e^{-\frac{1}{2}\epsilon^2 n} = (e^{-\frac{1}{2}\epsilon^2})^n$$

or

$$\Pr[S_n \geq \alpha \cdot \sqrt{n}] \quad \leq \quad e^{-\frac{1}{2}\alpha^2}$$

If we think of ϵ and α as positive constants, then the first inequality says the probability of having an $\Theta(n)$ deviation from expectation (0) is exponentially small, and the second inequality says that this "tail probability" balances out at $\Theta(\sqrt{n})$ away from expectation. This last statement is in accord with the central limit theorem, which states that

$$\lim_{n\to\infty} \Pr[S_n \geq \alpha \cdot \sqrt{n}] = \int_\alpha^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}\, dx.$$

The advantage of the Chernoff Bound is that it is valid for all n and a, and not merely a statement of limit.

Now, we prove the theorem.

Proof. The trick is to consider the exponentiation of S_n, namely the random variable $e^{\lambda S_n}$. Here $\lambda > 0$ is some number to be fixed later. We compute the expectation of $e^{\lambda S_n}$ as follows.

$$\mathrm{E}\left[e^{\lambda S_n}\right] = \mathrm{E}\left[e^{\lambda \sum_{i=1}^n X_i}\right] = \mathrm{E}\left[\prod_{i=1}^n e^{\lambda X_i}\right] = \prod_{i=1}^n \mathrm{E}\left[e^{\lambda X_i}\right]$$

To get the last equality, we use the assumption that the $\{X_i\}$ are independent random variables and hence $\{e^{\lambda X_i}\}$ are also independent. (Recall that, for independent random variables X and Y, the expectation is multiplicative $\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y]$.) For any i, the random variable $e^{\lambda X_i}$ takes values e^λ and $e^{-\lambda}$, each with probability $1/2$. Its expectation is

$$\mathrm{E}\left[e^{\lambda X_i}\right] = \cosh(\lambda) = \frac{e^\lambda + e^{-\lambda}}{2}$$

For $\lambda > 0$, the above quantity is bounded by

$$\frac{e^\lambda + e^{-\lambda}}{2} < e^{\lambda^2/2}.$$

One can prove the above inequality by analyzing the Taylor expansion of LHS and RHS. We do this analysis at the end of this proof. Using this bound, we have

$$\mathrm{E}\left[e^{\lambda S_n}\right] < \prod_{i=1}^n e^{\lambda^2/2} = e^{\lambda^2 n/2}$$

We can now apply Markov's inequality to get:

$$\Pr[S_n \geq \Delta] = \Pr[e^{\lambda S_n} \geq e^{\lambda\Delta}] \leq \frac{\mathrm{E}\left[e^{\lambda S_n}\right]}{e^{\lambda\Delta}} < \frac{e^{\lambda^2 n/2}}{e^{\lambda\Delta}}$$

The above inequality is true for any $\lambda > 0$, so we are now free to choose λ to optimize it. To get the tightest upper bound, we minimize the exponent in the above function. Standard calculus technique shows that a minimum occurs at $\lambda = \Delta/n$. This gives us

$$\Pr[S_n \geq \Delta] \quad < \quad e^{-\Delta^2/2n}$$

Finally, we show that

$$\frac{e^{\lambda} + e^{-\lambda}}{2} < e^{\lambda^2/2}.$$

The Taylor expansion of LHS is

$$\cosh \lambda = 1 + \frac{\lambda^2}{2!} + \frac{\lambda^4}{4!} + \cdots$$

The Taylor expansion for the function e^x shows that

$$e^{\lambda^2/2} = 1 + \frac{\lambda^2}{2} + \frac{(\lambda^2/2)^2}{2!} + \frac{(\lambda^2/2)^3}{3!} + \cdots$$

Compare the two series term by term. The kth term of LHS is $\lambda^{2k}/(2k)!$, while that of $e^{\lambda^2/2}$ is $(\lambda^2/2)^k/k!$. Focusing only on the even factors, we have $(2k!) \geq 2^k \cdot k!$ and strictly so for $k > 1$. Thus,

$$\cosh \lambda < e^{\lambda^2/2}.$$

□

By symmetry,

$$\Pr[|S_n| \geq \Delta] \quad < \quad 2e^{-\Delta^2/2n}$$

There are several different forms of Chernoff Bound which will be useful. They all concern tail probabilities of sums of independent random variables. Theorem 3.3 can be generalized as follows.

Let $\{X_1, X_2, \cdots, X_n\}$ be a set of n independent 0-1 random variables, with

$$\Pr[X_i = 1] = p_i, \quad \Pr[X_i = 0] = 1 - p_i,$$

and let $p = \sum_{i=1}^{n} p_i/n$. We define the centralized random variables $\{Y_1, Y_2, \cdots, Y_n\}$ where $Y_i = X_i - p_i$, then

$$\Pr[Y_i = 1 - p_i] = p_i, \quad \Pr[Y_i = -p_i] = 1 - p_i,$$

and $\mathrm{E}[Y_i] = 0$. Let $S_n = \sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} X_i - pn$.

Theorem 3.4. *For any $\Delta > 0$,*

$$\Pr[S_n \geq \Delta] \leq e^{-2\Delta^2/n}.$$

By symmetry, the bound applies to $-S_n$ as well, and so

$$\Pr[|S_n| \geq \Delta] \leq 2e^{-2\Delta^2/n}.$$

Note that when we take $\Delta = \delta pn$, then

$$\Pr[|\sum_{i=1}^{n} X_i - pn| > \delta pn] < 2e^{-2\delta^2 p^2 n}.$$

Here is another form. Let $\{X_1, X_2, \cdots, X_n\}$ be a set of n independent 0-1 random variables, with $\Pr[X_i = 1] = p$. Then

Theorem 3.5. *For any* $0 < \delta < 1/2$,

$$\Pr[\sum_{i=1}^{n} X_i > (1+\delta)pn] < e^{-\delta^2 pn/4}$$

and

$$\Pr[\sum_{i=1}^{n} X_i < (1-\delta)pn] < e^{-\delta^2 pn/2}.$$

Theorem 3.5 is better than Theorem 3.4 for small p.

For not necessarily 0-1 random variables,

Theorem 3.6. *Let* X_i, $1 \le i \le n$, *be mutually independent with all* $\mathrm{E}[X_i] = 0$ *and all* $|X_i| \le 1$. *Let* $S_n = \sum_{i=1}^{n} X_i$. *Then for all* $\Delta > 0$,

$$\Pr[S_n > \Delta] < e^{-\Delta^2/2n}.$$

The proofs of these versions of Chernoff Bound all follow similar lines.

A version of this type of bound also holds with hypergeometric distribution. Randomly pick n balls without replacement, from N black and white balls, with pN black balls. Let S be the number of black balls among n balls picked. The distribution is

$$\Pr[X = k] = \frac{\binom{pN}{k}}{\binom{(1-p)N}{n-k}\binom{N}{n}}.$$

Then

Theorem 3.7. *For any* $\Delta \ge 0$,

$$\Pr[|S - pn| \ge \Delta] \le 2e^{-2\Delta^2/n}.$$

This is known as the *Hoeffding bound*.

3.1.4 Universal Hashing

Definition 3.8 (Universal family of hash functions)**.** *Let* U *and* T *be finite sets. Let* S *be an index set for a family of functions* $\{h_s : U \to T\}_{s \in S}$. $\{h_s\}_{s \in S}$ *is called a* universal family of hash functions *if* $\forall \alpha, \beta \in T, \forall x, y \in U, x \ne y$,

$$\Pr_{s \in S}[h_s(x) = \alpha \wedge h_s(y) = \beta] = \frac{1}{|T|^2}$$

Notice that the RHS of the equation above $1/|T|^2$ is the probability of getting α and β when we choose two elements independently and uniformly at random from T.

The proper treatment in the Kolmogorov framework will be to define a measure space with the underlying set S, equipped with the uniform distribution. Then for all fixed $x \in U$, the map $Z_x : s \mapsto h_s(x)$ is a random variable.

We have $\forall x \neq y \in U$, $\forall \alpha, \beta \in T$, $\Pr_{s\in S}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] = \frac{1}{|T|^2}$. Hence, $\forall \alpha \in T, \forall x \in U$, take any $y \in U$, and $y \neq x$, (we assume $|U| \geq 2$),

$$\begin{aligned}
\Pr_{s\in S}[Z_x(s) = \alpha] &= \sum_{\beta\in T} \Pr_{s\in S}[Z_x(s) = \alpha \wedge Z_y(s) = \beta] \\
&= \sum_{\beta\in T} \frac{1}{|T|^2} \\
&= \frac{1}{|T|}
\end{aligned}$$

So, Z_x is a uniform random variable on T. It follows that for any $x \neq y \in U$ and $\alpha, \beta \in T$,

$$\Pr_{s\in S}[Z_x(s) = \alpha \ \wedge \ Z_y(s) = \beta] = \Pr_{s\in S}[Z_x(s) = \alpha] \cdot \Pr_{s\in S}[Z_y(s) = \beta].$$

So, for any $x \neq y \in U$, the random variables Z_x and Z_y are independent. The definition of universal hash function is equivalent to asking a set of *pairwise independent* and uniformly distributed random variables, $\{Z_x\}_{x\in U}$. The random variables in this set may be jointly dependent, but any two of them are independent.

However, instead of thinking in terms of pair-wise independent random variables $Z_x(s)$, we rather think of $h_s(x) = Z_x(s)$ as a random map from U to T, by randomly choosing an index $s \in S$. The two views are of course completely equivalent here.

An Example: Let p be a prime number. Then $\mathbb{Z}/p = \{0, 1, \ldots, p-1\}$ with the operations $+$ and $\cdot$ forms a finite field. Consider the map $h_{s=(a,b)} : x \mapsto ax + b$ for $a, b \in \mathbb{Z}/p$. We will verify that $\{h_{(a,b)}\}_{a,b\in\mathbb{Z}/p}$ is a universal family of hash functions.

For all $x, y, \alpha, \beta \in \mathbb{Z}/p, x \neq y$, how many pairs $(a, b) \in \mathbb{Z}/p$ are there satisfying the following equations?

$$\begin{aligned}
ax + b &= \alpha \\
ay + b &= \beta
\end{aligned}$$

(In the above equations, a and b are the unknowns.) The determinant of this 2×2 linear system is

$$\det \begin{pmatrix} x & 1 \\ y & 1 \end{pmatrix} = x - y \neq 0.$$

Therefore, there exists a unique solution such that this equation holds. Thus,

$$\Pr_{s=(a,b)\in(\mathbb{Z}/p)^2}[h_s(x) = \alpha \wedge h_s(y) = \beta] = \frac{1}{p^2}.$$

So, $\{h_{(a,b)}\}_{a,b\in\mathbb{Z}/p}$ is a universal family of hash functions.

This can be generalized to any finite field $\mathrm{GF}(p^n)$. It is known that for any prime p and any $n \geq 1$, there is a finite field of p^n elements. Up to isomorphism such a field is unique. In particular, for any $k \geq 0$, the polynomial $X^{2\cdot 3^k} + X^{3^k} + 1$ is an irreducible polynomial in $\mathbb{Z}_2[X]$, and therefore we have an explicit finite field in the form of

$$\mathbb{Z}_2[X]/(X^{2\cdot 3^k} + X^{3^k} + 1).$$

In the definition of this particular family of universal hash functions via affine linear functions $ax+b$, if it is defined over a finite field $\mathrm{GF}(2^n)$, we can truncate any number of bits from n to make $|T| = 2^k$, for any $0 \leq k \leq n$. Whenever in the following we speak of a family of universal hash functions, unless otherwise stated, we always refer to this family of affine linear functions, and if necessary, over that particular family of finite fields $\mathrm{GF}(2^n)$, with $n = 2 \cdot 3^k$.

The power of universal hash functions comes from the fact that on the one hand they behave more or less like a random function, on the other hand they can be succinctly specified by only $2n$ bits.

4 A Randomized Algorithm for MAXCUT

As a taste for randomized algorithms, we discuss the graph problem MAXCUT. Let $G = (V, E)$ be an undirected graph over n vertices. A cut C of G is a partition of vertices V into disjoint union $V_1 \cup V_2$. We also identify a cut with the set of edges between V_1 and V_2, i.e., $e \in C$ consists of those edges with one of its incident vertices in V_1 and the other in V_2.

Definition 4.1 (MAXCUT)**.** *A* MAXCUT *of a graph $G = (V, E)$ is a cut C such that $|C|$ is maximized over all cuts of G.*

Similar to MAXCUT, MINCUT of G is defined as the minimum $|C|$ over all cuts of G. We know that MINCUT is in P. This is the max-flow min-cut theorem, essentially equivalent to the theorems in Section 2.

Project: Explain the relationship between them.

MINCUT can be solved using maximum network flow between all pairs of vertices.

We know that MAXCUT is NP-complete, therefore we do not expect to solve it efficiently. However, we can look for approximate solutions. To quantify the accuracy of our approximations we will introduce a new term. We want a polynomial-time algorithm that achieves a cut C such that

$$\frac{|C|}{|C^*|} \geq r$$

where C^* is a maximum cut. Such an algorithm is called an r-approximation.

4.1 Deterministic MAXCUT Approximation Algorithm

We can define a greedy algorithm that achieves a 1/2-approximation:
For a graph $G = (V, E)$ with $V = \{1, \ldots, n\}$, define $E_i = \{(k, i) \in E : k < i\}$. Initially, let $V_1 = V_2 = \emptyset$. Then, for each i from 1 to n, add i to either V_1 or V_2 so that the number of edges in E_i that are on the cut is maximized, i.e., put i in V_1 iff $|\{k \in V_2 : (k, i) \in E_i\}| \geq |\{k \in V_1 : (k, i) \in E_i\}|$. We claim that this heuristic achieves 1/2-approximation.

Let C be the cut obtained by the algorithm. The disjoint sets $E_1, E_2, \ldots, E_n$ partition E. So, $|E| = \sum_i E_i$. For each $i \in V$, let $E'_i = E_i \cap C$. Then, $C = \bigcup_i E'_i$. As sets E_i are disjoint, the sets E'_i are also disjoint. Thus, $|C| = \sum_i |E'_i|$. The main observation is that for each $i \in V$, we have $E'_i \geq E_i/2$. We conclude that $|C| \geq |E|/2$. As the size of any maximum cut $|C^*| \leq |E|$, we get $|C| \geq |C^*|/2$.

4.2 Randomized MAXCUT Approximation Algorithm

We present a randomized 1/2-approximation algorithm for MAXCUT. Then we show that it can be derandomized in polynomial time. This example illustrates the ideas of randomization and derandomization in a simple setting.

The randomized algorithm is very simple. Assign a monkey at each vertex and have each monkey throw a dart. If it throws to the left, assign the vertex to V_1, and if it throws to the right, assign it to V_2. More formally, given a graph $G = (V, E)$, we assign each vertex independently with equal probability to either V_1 or V_2. This will give us a cut C of G, and we will show that the expected size of $C \geq |C^*|/2$.

Consider an edge $(i, j) \in E$. $\Pr[(i, j) \in C] = 1/2$. For $e \in E$, define χ_e to be a random variable such that $\chi_e = 1$ if $e \in C$ and $\chi_e = 0$ if $e \notin C$. Then $|C| = \sum_{e \in E} \chi_e$. Thus,

$$\mathrm{E}[|C|] = \sum_{e \in E} \mathrm{E}[\chi_e] = \sum_{e \in E} \Pr[e \in C] = \frac{|E|}{2} \geq \frac{1}{2}|C^*|.$$

The first equality follows from linearity of expectation, $E[X + Y] = E[X] + E[Y]$, for any two random variables X and Y. This formula holds even if X and Y are not independent.

4.3 Derandomizing MAXCUT Approximation Algorithm Using Universal Hash Functions

Let $G = (V, E)$ be a graph with $V = \{0, \ldots, n-1\}$. Set k so that $2^k \geq n > 2^{k-1}$. Choose a and b at random from $\mathrm{GF}(2^k)$. For each $i \in V$, treat i as a member of $\mathrm{GF}[2^k]$ compute $ai + b$. Assign i to either V_1 or V_2 according to the first bit of $ai + b$. We claim that the expected size of the cut obtained is $|E|/2$.

Let $\chi_{(a,b)}(i) =$ the first bit of $ai+b$. We know that $\{ai+b\}_{a,b \in \mathrm{GF}(2^k)}$ is a universal family of hash functions. Thus, $\{\chi_{(a,b)}\}_{a,b \in \mathrm{GF}(2^k)}$ is a universal family of hash functions. Then, a cut C obtained by the above randomized algorithm is given by $C = \{(i, j) | \chi_{(a,b)}(i) \neq \chi_{(a,b)}(j)\}$.

Because $\{\chi_{(a,b)}\}_{a,b \in \mathrm{GF}(2^k)}$ is a universal family of hash functions, $\Pr[\chi_{(a,b)}(i) \neq \chi_{(a,b)}(j)] = 1/2$. Thus, using the analysis from Section 3, we have $\mathrm{E}[|C|] = |E|/2$.

We can derandomize the above algorithm in polynomial time. There are less than $4n^2$ different choices for (a, b). To derandomize, we can examine the cuts created by $\{\chi_{(a,b)}\}$ for all $a, b \in \mathrm{GF}(2^k)$ in polynomial time. One of these cuts is guaranteed to be at least $|C^*|/2$ because $\mathrm{E}[|C|] \geq |C^*|/2$. This gives us a deterministic r-approximation algorithm for MAXCUT.

Although this derandomized algorithm does not give a better approximation ratio than the greedy algorithm, it is a parallel algorithm. For each pair (a, b), the determination of which side of the cut each vertex is on can be determined separately regardless of the other vertices. Thus, this can be executed in parallel. Additionally, the cut produced from each pair (a, b) can be determined in parallel. After all these cuts have been computed, the maximum can be selected. We will say later that this can be computed in NC.

4.4 Goemans-Williamson Algorithm

Goemans and Williamson gave an approximation algorithm for MAXCUT with error ratio of about 12%. Randomization plays an important role here combined with semidefinite optimization.

Typically a subset of $[n]$ can be described by a binary sequence in $\{0,1\}^n$. Although this is not essential, we will find it here more convenient to describe such a subset $V_1 \subseteq V$, which corresponds to a cut $V_1 \cup (V \setminus V_1)$, by a vector $x \in \{-1,1\}^n$, by letting $x_i = -1$ iff $i \in V_1$. Then the cut size is $\sum_{e=\{i,j\}} (x_i - x_j)^2/4$. Therefore the MAXCUT problem seeks to maximize

$$\frac{1}{4} \sum_{e=\{i,j\}} (x_i - x_j)^2$$

subject to the constraint $x \in \{-1,1\}^n$. This constraint can be expressed as

$$x_i^2 = 1, \forall i \in V.$$

Such a problem is called a quadratic programming problem, which, in its generality, is also NP-hard (which is of course no surprise, since we have just reduced an NP-hard problem MAXCUT to it.)

The next trick is to linearize the problem by introducing a set of new variables $y_{ij}, 1 \le i, j \le n$, with the intention that $y_{ij} = x_i x_j$. Under this new set of variables, the objective function becomes

$$\frac{1}{4} \sum_{e=\{i,j\}} (y_{ii} + y_{jj} - 2y_{ij})$$

to be maximized subject to

$$y_{ii} = 1, \forall i \in V.$$

Observe that a "solution" y_{ij} need not correspond to any real solution x_i. In particular, if $y_{ij} = x_i x_j$, then the matrix $Y = (y_{ij})$ is symmetric and positive semi-definite, being the product of X and its transpose X^{T},

$$Y = XX^{\mathrm{T}} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$$

Not only that, since X is $n \times 1$, Y is of rank 1 (assuming $X \neq 0$, which is implied by $y_{ii} = 1$). However, we choose to ignore this rank condition, yet preserve the constraint that Y is symmetric and positive semi-definite. The reason for this is that there is a polynomial time algorithm, based on the ellipsoid method, which solves this "semi-definite" programming problem optimally.

What we have done is called a "relaxation" of the original problem. The requirement of Y being positive semi-definite amounts to a seemingly infinitely many inequalities

$$v^{\mathrm{T}} Y v \ge 0, \quad \forall v \in \mathbf{R}^n.$$

These are linear constraints on y_{ij}. It is a trick of the "semi-definite" programming algorithm which can handle these infinitely many inequalities implicitly. We will not dwell further on this, and simply assume such a polynomial time algorithm is available.

Coming back to MAXCUT, in polynomial time we find a symmetric and positive semi-definite Y^*, with $y^*_{ii} = 1$ which maximizes $\frac{1}{4} \sum_{e=\{i,j\}} (y_{ii} + y_{jj} - 2y_{ij})$ among all Y satisfying these constraints. Let the maximum value be M^*. Since this is a relaxation of the original problem, M^* is an upper bound of the maximum cut size.

It is known that a symmetric and positive semi-definite matrix Y^* can be expressed as a product of UU^{T}, where $U = \begin{pmatrix} u_1 & u_2 & \cdots & u_n \end{pmatrix}^{\mathrm{T}}$, where column vectors $u_i \in \mathbf{R}^n$. Moreover this decomposition can be found in polynomial time.

Thus, $y^*_{ii} = u_i^{\mathrm{T}} u_i = ||u_i||^2 = 1$ says that each vector u_i is a unit vector. Moreover, "formally" retracing the expansion of $\sum_{e=\{i,j\}}(x_i - x_j)^2$ into $\sum_{e=\{i,j\}}(y_{ii} + y_{jj} - 2y_{ij})$, we get

$$M^* = \frac{1}{4} \sum_{e=\{i,j\}} ||u_i - u_j||^2.$$

The next idea of Goemans and Williamson is brilliant. *Randomly* choose a hyperplane Π in $\mathbf{R}^n$. This amounts to choosing a unit vector $v \in \mathbf{R}^n$, uniformly on the unit sphere, as the normal vector to Π. (This can be carried out approximately with exponentially small error; we are ignoring issues of discretizing the process here.) Now partition V according to which side of Π the vector u_i falls. More precisely, assign vertex i to V_1 iff the inner product $\langle u_i, v \rangle > 0$.

If we express the cut size thus formed as a sum of 0-1 random variables, which indicate whether edge $e = \{i, j\}$ belongs to the cut, then the expectation is

$$\sum_{e=\{i,j\}} \Pr[\ \Pi \text{ separates } u_i, u_j].$$

To investigate this probability $\Pr[\ \Pi$ separates $u_i, u_j]$, we only need to think of it in terms of the 2-dimensional space spanned by u_i and u_j. Clearly this probability is θ_{ij}/π where θ_{ij} is the angle between u_i and u_j. So

$$\sum_{e=\{i,j\}} \Pr[\ \Pi \text{ separates } u_i, u_j] = \sum_{e=\{i,j\}} \frac{\theta_{ij}}{\pi}.$$

Meanwhile the length $||u_i - u_j||$ is clearly $2 \sin \frac{\theta_{ij}}{2}$. Hence

$$\sum_{e=\{i,j\}} \Pr[\ \Pi \text{ separates } u_i, u_j] = \frac{1}{\pi} \sum_{e=\{i,j\}} f(\theta_{ij}) \cdot ||u_i - u_j||^2,$$

where $f(\theta) = \frac{\theta}{4 \sin^2(\theta/2)}$.

Simple calculus shows that the function $f(\theta)$ achieves minimum .69002507... at 2.331122370.... It follows that the expectation of the cut

$$\sum_{e=\{i,j\}} \frac{\theta_{ij}}{\pi} \geq \frac{.69002507}{\pi} \sum_{e=\{i,j\}} ||u_i - u_j||^2 \approx 0.878 \cdot M^*.$$

This is a randomized MAXCUT algorithm that achieves 87.8% approximation ratio. There are ways to derandomize this algorithm, but we will not discuss this problem any further.

5 Randomized Complexity Classes

Definition 5.1 (BPP)**.** *BPP stands for bounded error probabilistic polynomial time. A language L is in* BPP*, if there is a boolean predicate $D(\cdot, \cdot)$, computable in deterministic polynomial time such that,*

$$\begin{aligned} x \in L &\implies \Pr_y[D(x,y)=1] \geq 3/4 \\ x \notin L &\implies \Pr_y[D(x,y)=1] \leq 1/4, \end{aligned}$$

where $|y|$, the number of random bits used, is polynomial in length of the input $|x|$.

Definition 5.2 (RP)**.** *RP was the first class defined to capture feasible probabilistic computation, and simply stands for randomized polynomial time. A language L is in* RP*, if there is a boolean predicate $D(\cdot,\cdot)$, computable in deterministic polynomial time such that,*

$$\begin{aligned} x \in L &\implies \Pr_y[D(x,y)=1] \geq 1/2 \\ x \notin L &\implies \Pr_y[D(x,y)=1] = 0, \end{aligned}$$

where $|y|$, the number of random bits used, is polynomial in length of the input $|x|$.

Definition 5.3 (ZPP)**.** *A language L is said to be in* ZPP *if there is a polynomial time computable function $D : \{0,1\}^* \times \{0,1\}^* \mapsto \{0,1,?\}$ such that if $x \in L$, for any y, its output $D(x,y) \in \{1,?\}$, and if $x \notin L$, for any y, its output $D(x,y) \in \{0,?\}$. Moreover, D should have high success probability:*

$$\begin{aligned} x \in L &\implies \Pr_y[D(x,y)=1] \geq 1/2 \\ x \notin L &\implies \Pr_y[D(x,y)=0] \geq 1/2, \end{aligned}$$

where $|y|$, the number of random bits used, is polynomial in length of the input $|x|$.

Exercise: Prove that ZPP is the class of languages with expected polynomial time algorithms that never make any errors.

5.1 Amplification of BPP

In this section, we show that the probability of success of a BPP algorithm can be "amplified" to be as high as exponentially close to 1, with only a polynomial amount of extra work. We use the Chernoff bound in Theorem 3.4.

Let L be a language accepted by a probabilistic polynomial time TM M, in the following sense:

$$\begin{aligned} x \in L &\implies \Pr[M(x;r)=1] \geq \frac{1}{2} + \epsilon \\ x \notin L &\implies \Pr[M(x;r)=1] \leq \frac{1}{2} - \epsilon \end{aligned}$$

for random strings r of some polynomial length in $n = |x|$. Here, ϵ can be as low as $1/p(n)$ for some fixed polynomial $p(\cdot)$. We wish to amplify the success probability of the algorithm. In particular, we want to get exponentially close to 1, meaning, for any fixed polynomial $q(\cdot)$, we want a machine M' with

$$\begin{aligned} x \in L &\implies \Pr[M'(x;r')=1] \geq 1 - e^{-q(n)} \\ x \notin L &\implies \Pr[M'(x;r')=1] \leq e^{-q(n)} \end{aligned}$$

We require that M' run in polynomial time, and hence the length of the random string $|r'|$ used by M' should also be polynomially bounded.

The idea is to run M a large polynomial number of times, and take the majority vote. Given an input x, M' will simply run the machine M on input x some $2m+1$ times, with $m \geq q(n)/(4\epsilon^2)$, say, and accept x iff at least $m+1$ runs of M on x accept. In this process we will need independent and uniformly chosen random $r_1, r_2, \cdots, r_{2m+1}$, with a total length of the random string $|r'| = O(|r|q(n)/\epsilon^2)$.

Let X_i be the 0-1 random variable indicating the i-th run $M(x; r_i)$, $p = \Pr[M(x; r) = 1]$, and $S = \sum_i (X_i - p)$. Suppose $x \notin L$, then $p \leq \frac{1}{2} - \epsilon$. Apply Theorem 3.4 with $\Delta = (2m+1)\epsilon$, we get

$$\Pr[M' \text{ accepts } x] = \Pr[\sum_i X_i \geq m+1] \leq \Pr[S \geq (2m+1)\epsilon] \leq e^{-q(n)}.$$

Similarly, if $x \in L$, then $p \geq \frac{1}{2} + \epsilon$, and

$$\Pr[M' \text{ rejects } x] \leq e^{-q(n)}.$$

When ϵ^{-1} is polynomially bounded, so is $m = O(q(n)/\epsilon^2)$. This achieves exponentially small error probability.

Exercise: What if the threshold is not 1/2?

5.2 Sipser-Lautemann Theorem: BPP $\subseteq$ PH

Let L be a language in BPP. We will show that $L \in \Sigma_2^p$. Without loss of generality, there is a polynomial time computable predicate $D(\cdot, \cdot)$ such that,

$$\begin{aligned} x \in L \quad &\Longrightarrow \quad \Pr_{y \in \{0,1\}^m}[D(x, y) = 1] \geq 1 - 1/m \\ x \notin L \quad &\Longrightarrow \quad \Pr_{y \in \{0,1\}^m}[D(x, y) = 1] \leq 1/m \end{aligned}$$

where the number of random bits used m is polynomially bounded in input length $n = |x|$. For any input $x \in \{0,1\}^n$, define its witness set $W_x = \{y \in \{0,1\}^m | D(x, y) = 1\}$. So, if $x \in L$ then the witness set W_x is "fat", whereas, if $x \notin L$, the witness set is "thin". We will show how to test whether W_x is fat or thin in Σ_2^p and thereby prove that $L \in \Sigma_2^p$. Let us first formalize the notion of fat and thin sets and prove some properties of such sets.

We say that $S \subseteq \{0,1\}^m$ is *fat*, if $|S|/2^m \geq 1 - 1/m$. S is said to be thin, if $|S|/2^m \leq 1/m$. In general a subset S may be neither fat nor thin. But, the witness sets we are interested in are always either fat or thin. For a string $u \in \{0,1\}^m$, define $S \oplus u = \{s \oplus u | s \in S\}$. Here, $s \oplus u$ denotes the m-bit string obtained by bit-wise XOR of s and u: if $s = s_1 s_2 \ldots s_m$ and $u = u_1 u_2 \ldots u_m$, then $s \oplus u = s_1 \oplus u_1 \cdots s_m \oplus u_m$. Think of $\{0,1\}^m$ as a vector space, S as a subset of this space and u to be a vector in it. Then, $S \oplus u$ is nothing but the subset obtained by "shifting" S by u.

We first discuss informally the effect of shifting fat and thin sets. Suppose S is fat. Then, if we choose a suitable number of shift vectors $u_1, u_2, \ldots, u_r$ at random, with high probability, the union of these shifts will "cover" the entire space: $\bigcup_{i=1}^r (S \oplus u_i) = \{0,1\}^m$. On the other hand, if S is a thin, then for any set of vectors $u_1, u_2, \ldots, u_r$, where $r < m$, the shifts will not cover the space: $\bigcup_{i=1}^r (S \oplus u_i) \neq \{0,1\}^m$. We will formally prove these claims. Observe that, with these claims, it is easy to put L in Σ_2^p: the condition "there is a set of vectors such that W_x shifted by these vectors covers the entire space" can be expressed as a Σ_2^p predicate!

Lemma 5.4. *Let $S \subseteq \{0,1\}^m$.*

1. *If S is thin, then for any $r < m$, for any set of r vectors, the shifts cannot cover the entire space:*
$$\Pr_{u_1,u_2,\ldots,u_r \in \{0,1\}^m}\left[\bigcup_{i=1}^{r}(S \oplus u_i) = \{0,1\}^m\right] = 0.$$
2. *If S is fat, then with high probability, randomly chosen shifts will cover the entire space:*
$$\Pr_{u_1,u_2,\ldots,u_r \in \{0,1\}^m}\left[\bigcup_{i=1}^{r}(S \oplus u_i) = \{0,1\}^m\right] \geq 1 - \frac{2^m}{m^r}$$

Proof. The first part is obvious. For any vectors $u_1, u_2, \ldots, u_r$, the union of the shifts has cardinality,
$$\left|\bigcup_{i=1}^{r}(S \oplus u_i)\right| \leq r \cdot |S|$$
Since we assume that $r < m$ and S is thin, $|\bigcup_{i=1}^{r}(S \oplus u_i)| < 2^m$. Thus, the shifts cannot cover the entire space $\{0,1\}^m$ which has cardinality 2^m.

To prove the second part, we will bound the probability of the negation of the event under consideration. A set of vectors $u_1, u_2, \ldots, u_r$ do not cover the entire space means that some vector $y \in \{0,1\}^m$ is not covered by these shifts. So,
$$\Pr_{u_1,u_2,\ldots,u_r}\left[\bigcup_{i=1}^{r}(S \oplus u_i) \neq \{0,1\}^m\right] \leq \sum_{y \in \{0,1\}^m} \Pr_{u_1,u_2,\ldots,u_r}\left[y \notin \bigcup_{i=1}^{r}(S \oplus u_i)\right]$$
Fix any $y \in \{0,1\}^m$. By the properties of $\oplus$ function, we see that $y \notin S \oplus u$ iff $y \oplus u \notin S$ (we use the fact that for any u, $u \oplus u = 0$). Thus, y will not be covered by the r shifts iff $\{y \oplus u_1, y \oplus u_2, \ldots, y \oplus u_r\} \cap S = \emptyset$. For any u chosen uniformly at random from $\{0,1\}^m$, $y \oplus u$ is distributed uniformly in $\{0,1\}^m$. (We use the fact that the function $f_y : u \mapsto y \oplus u$ is 1-1.) So, for a randomly chosen u, $\Pr[y \oplus u \notin S] \leq 1/m$, because S is fat. By independence, it follows that
$$\Pr_{u_1,u_2,\ldots,u_r}[\{y \oplus u_1, y \oplus u_2, \ldots, y \oplus u_r\} \cap S = \emptyset] = \prod_{i=1}^{r} \Pr_{u_i}[y \oplus u_i \notin S] \leq \left(\frac{1}{m}\right)^r.$$
We conclude that,
$$\begin{aligned}\Pr_{u_1,u_2,\ldots,u_r}\left[\bigcup_{i=1}^{r}(S \oplus u_i) \neq \{0,1\}^m\right] &\leq \sum_{y \in \{0,1\}^m} \Pr_{u_1,u_2,\ldots,u_r}\left[y \notin \bigcup_{i=1}^{r}(S \oplus u_i)\right] \\ &= \frac{2^m}{m^r}.\end{aligned}$$
Part 2 of the lemma follows from the above bound. □

Using Lemma 5.4, we can show that BPP $\subseteq \Sigma_2^p$. For a suitable value of r, the lemma will show that if S is fat, then for some set of r vectors, the shifts would cover the entire space, and if S is thin, then for any set of r vectors, the shifts would not cover the entire space. The above property can be tested in Σ_2^p. Formal proof is given below.

Theorem 5.5 (Sipser–Lautemann)**.** BPP $\subseteq \Sigma_2^p$.

Proof. Let $L \in$ BPP. Without loss of generality, there is a polynomial time computable predicate $D(\cdot,\cdot)$ such that,

$$\begin{aligned} x \in L &\implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1] \geq 1-1/m \\ x \notin L &\implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1] \leq 1/m \end{aligned}$$

where the number of random bits used m is polynomially bounded in input length $n = |x|$. For an input $x \in \{0,1\}^n$, define its witness set $W_x = \{y | D(x,y) = 1\}$. If $x \in L$ then the witness set W_x is fat, whereas, if $x \notin L$, the witness set is thin. Choose $r = m/2$. Then, from Lemma 5.4, we have

$$\begin{aligned} x \in L &\implies \Pr_{u_1,u_2,\ldots,u_r}\left[\bigcup_{i=1}^{r}(W_x \oplus u_i) = \{0,1\}^m\right] \geq 1 - \frac{2^m}{m^r} \\ x \notin L &\implies \Pr_{u_1,u_2,\ldots,u_r}\left[\bigcup_{i=1}^{r}(W_x \oplus u_i) = \{0,1\}^m\right] = 0. \end{aligned}$$

For large m ($m > 4$ so that $2^m < m^{m/2}$), the first probability > 0. So, if $x \in L$, there exist r vectors such that the shifts cover the entire space, and if $x \notin L$, for any r vectors, the shifts do not cover the entire space. Observe that, for any $u_1, u_2, \ldots, u_r$,

$$\bigcup_{i=1}^{r}(W_x \oplus u_i) = \{0,1\}^m \iff \forall y \in \{0,1\}^m \left[\vee_{i=1}^{r}(y \oplus u_i \in W_x)\right]$$

It follows that,

$$x \in L \iff \exists u_1, u_2, \ldots, u_r \forall y \in \{0,1\}^m \left[\vee_{i=1}^{r}(y \oplus u_i \in W_x)\right].$$

$y \oplus u_i \in W_x$ simply means that $D(x, y \oplus u_i) = 1$. So we conclude that,

$$x \in L \iff \exists u_1, u_2, \ldots, u_r \forall y \in \{0,1\}^m \left[\vee_{i=1}^{r}(D(x, y \oplus u_i) = 1)\right].$$

The predicate $\vee_{i=1}^{r}(D(x, y \oplus u_i) = 1)$ is testable in polynomial time, as r is polynomial in n and D is a polynomial time predicate. We conclude that $L \in \Sigma_2^p$. □

BPP is closed under complementation. So, we have also shown that BPP $\in \Pi_2^p$. We can also prove this claim directly by exhibiting a Π_2^p predicate. To do that, we first rephrase Lemma 5.4. Observe that, for any set $S \subseteq \{0,1\}^m$ and $u \in \{0,1\}^m$, $(S \oplus u)^c = S^c \oplus u$. Hence, for any $u_1, u_2, \ldots, u_r$,

$$\left[\bigcup_{i=1}^{r}(S \oplus u_i) = \{0,1\}^m\right] \iff \left[\bigcap_{i=1}^{r}(S^c \oplus u_i) = \emptyset\right].$$

Moreover, S is thin iff S^c is fat. So, Lemma 5.4 can be rephrased as follows. Suppose S is fat. Then S^c is thin. Applying Part 1 of the lemma to S^c, we see that for any set of $r < m$ shift vectors, $u_1, u_2, \ldots, u_r$, the intersection of the shifts of S is non-empty. Now suppose S is thin. Then S^c is fat. Applying Part 2 of the lemma to S^c, we see that for randomly chosen r shift vectors $u_1, u_2, \ldots, u_r$, for reasonably large r, with high probability, the intersection of the shifts of S is empty. Formally,

Lemma 5.6. *Let $S \subseteq \{0,1\}^m$.*

1. *If S is fat, then for any $r < m$, for any set of r vectors, the intersections of the shifts is non-empty:*
$$\Pr_{u_1,u_2,\ldots,u_r}\left[\bigcap_{i=1}^{r}(S \oplus u_i) \neq \emptyset\right] = 1.$$
2. *If S is thin, then with high probability, randomly chosen shifts will have an empty intersection:*
$$\Pr_{u_1,u_2,\ldots,u_r}\left[\bigcap_{i=1}^{r}(S \oplus u_i) = \emptyset\right] \geq 1 - \frac{2^m}{m^r}$$

Applying Lemma 5.6 to witness sets, with $r = m/2$, (assuming $m > 4$ so that $2^m < m^{m/2}$), then

$$x \in L \implies \forall u_1, u_2, \ldots, u_r \left[\bigcap_{i=1}^{r}(W_x \oplus u_i) \neq \emptyset\right]$$
$$x \notin L \implies \exists u_1, u_2, \ldots, u_r \left[\bigcap_{i=1}^{r}(W_x \oplus u_i) = \emptyset\right]$$

The above property can be expressed as a Π_2^p predicate:

$$x \in L \iff \forall u_1, u_2, \ldots, u_r \exists y \left[\wedge_{i=1}^{r}(y \oplus u_i) \in W_x)\right]$$

5.3 Isolation Lemma

The isolation lemma provides a mechanism to approximately compute the size of some set $S \subseteq \{0,1\}^m$, such as a BPP witness set. In this section, we state and prove the lemma. In the subsequent sections, we use the lemma to give an alternative proof that BPP $\subseteq \Sigma_2^p$, and also discuss approximate counting.

Let S and T be finite sets and $\mathcal{H}$ be a universal family of hash functions from S to T. Two distinct elements $x, y \in S$ are said to *collide* under a hash function $h \in \mathcal{H}$, if $h(x) = h(y)$. We say that a hash function h *isolates* an element $x \in S$, if no element in S collides with x. One can imagine that h "likes" x and gives it a separate seat in T to sit alone comfortably! A set of hash functions $\{h_1, h_2, \ldots, h_r\}$ is said to isolate an element $x \in S$, if one of these function h_i isolates x. The set of functions is said to *isolate all of* S, if for every element $x \in S$, there is some function h_i in the set that isolates x.

Suppose we choose r hash functions $h_1, h_2, \ldots, h_r$, uniformly and independently at random from $\mathcal{H}$. What is the probability that the set of these functions isolates all of S? The answer depends on the size of S compared to the size of T and the value of r. Intuitively, if S is sufficiently smaller than T and r is large enough, with high probability, the randomly chosen set of functions will isolate all of S. On the other hand, if S is large compared to T and r is small enough, then there is not enough space in T to give "separate seats" for all the elements in S. For suitable choices of $|T|$ and r, we can make the probability of isolation zero.

Lemma 5.7 (Isolation Lemma)**.** *Let $\mathcal{H}$ be a universal family of hash functions from S to T. Let $h_1, h_2, \ldots, h_r$ be chosen uniformly and independently at random from $\mathcal{H}$, where $r > 1$.*

1. *If* $|S| \geq r|T|$, *then*

$$\Pr_{h_1,h_2,\ldots,h_r}[\{h_1, h_2, \ldots, h_r\} \textit{ isolates all of } S] = 0.$$

2. *Suppose* $|S|^{r+1} \leq |T|^r$. *Then,*

$$\Pr_{h_1,h_2,\ldots,h_r}[\{h_1, h_2, \ldots, h_r\} \textit{ isolates all of } S] > 1 - \frac{|S|^{r+1}}{|T|^r}.$$

(In the Lemma, if $r = 1$ then we require $|S| > |T|$, the statement still holds.)

Proof. The first part of lemma is quite obvious. Any hash function $h \in \mathcal{H}$ can isolate at most $|T| - 1$ elements in S (it can assign "separate seats" for some $|T| - 1$ elements in S and then map all the other elements to the last "seat" in T). So, any set of r hash functions can together isolate at most $r(|T| - 1)$ elements. As $|S| \geq r|T|$, no set of r hash functions can isolate all of S.

We next prove the second part of the lemma. Fix any distinct elements $x, y \in S$. From the properties of universal family of hash functions, the probability that x and y collide under h is $1/|T|$. So, for any element $x \in S$,

$$\begin{aligned}\Pr_{h\in\mathcal{H}}[h \text{ does not isolate } x] &\leq \sum_{y\in S-\{x\}} \Pr_{h\in\mathcal{H}}[x \text{ and } y \text{ collide under } h] \\ &< \frac{|S|}{|T|}\end{aligned}$$

It follows that, for any x in S, if we choose $h_1, h_2, \ldots, h_r$ uniformly and independently at random,

$$\Pr_{h_1,h_2,\ldots,h_r\in\mathcal{H}}[\text{none of the } h_i \text{ isolates } x] < \left(\frac{|S|}{|T|}\right)^r.$$

Finally, the probability that randomly chosen r hash functions do not isolate all of S can be bounded by summing up over all possible elements in S:

$$\begin{aligned}&\Pr_{h_1,h_2,\ldots,h_r\in\mathcal{H}}[\{h_1, h_2, \ldots, h_r\} \text{ does not isolates all of } S] \\ \leq\ &\sum_{x\in S} \Pr_{h_1,h_2,\ldots,h_r\in\mathcal{H}}[\text{none of } h_1, h_2, \ldots, h_r \text{ isolates } x] \\ <\ &|S| \times \left(\frac{|S|}{|T|}\right)^r\end{aligned}$$

We conclude that

$$\Pr_{h_1,h_2,\ldots,h_r\in\mathcal{H}}[\{h_1, h_2, \ldots, h_r\} \text{ isolates all of } S\ ] > 1 - \frac{|S|^{r+1}}{|T|^r}$$

□

5.4 BPP $\subseteq \Sigma_2^p$ - Another Proof Using the Isolation Lemma

This section shows how to use the isolation lemma to put BPP in Σ_2^p. We first present the proof idea informally. Let L be a language in BPP, via a randomized algorithm that uses m random bits, where m is polynomial in the input length. We assume that the algorithm has been suitably amplified. We will fix parameters r and size of the target set T suitably in Lemma 5.7. For any input x, let $W_x \subseteq \{0,1\}^m$ be the set of witness strings on which the algorithm accepts the input x. The parameters will be chosen appropriately, so that if $x \in L$, W_x will be large enough that no set of r hash functions $h_1, h_2, \ldots, h_r$ will isolate all of W_x; on the other hand, if $x \notin L$, W_x will be small enough so that, with high probability, randomly chosen r hash functions $h_1, h_2, \ldots, h_r$ will isolate all of W_x, and in particular there will exist some set of r hash functions that isolates all of W_x. Hence, $x \notin L$ iff there exist a set of r hash functions $\{h_1, h_2, \ldots, h_r\}$ that isolates all of W_x. The later condition can be expressed as a Σ_2^p predicate, thereby placing L in Π_2^p. A formal proof follows.

Theorem 5.8. BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$.

Proof. Let $L \in$ BPP. We prove that $L \in \Pi_2^p$. As BPP is closed under complementation, the lemma follows. Without loss of generality, there is a deterministic polynomial time boolean predicate $D(\cdot,\cdot)$ such that for any input $x \in \{0,1\}^n$,

$$\begin{aligned} x \in L \quad &\Longrightarrow \quad \Pr_{y \in \{0,1\}^m}[D(x,y) = 1] \geq \frac{1}{2} \\ x \notin L \quad &\Longrightarrow \quad \Pr_{y \in \{0,1\}^m}[D(x,y) = 1] \leq \frac{1}{4m} \end{aligned}$$

where the number of random bits used m is polynomial in the input length n. Wolog we can assume m is a power of 2. Fix $r = m$, and let T be a set of size $2^m/(2m)$. (As it is usually the case in such proofs, the above choices of parameters such as the extent of amplification, value of r and size of $|T|$ are not crucial. We can fix them in many ways to make the proof work! One such setting is given above). Let $\mathcal{H}$ be a universal family of hash functions from $\{0,1\}^m$ to T. For any input x, let $W_x = \{y | D(x,y) = 1\}$. Suppose $x \in L$. Then,

$$|W_x| \geq 2^{m-1} = r \cdot |T|.$$

So, Lemma 5.7 shows that no set of r hash functions $\{h_1, h_2, \ldots, h_r\}$ from $\mathcal{H}$ isolates all of W_x. On the other hand, suppose $x \notin L$. Then, from our choice of parameters,

$$\frac{|W_x|^{r+1}}{|T|^r} \leq \frac{1}{4m}.$$

(In more detail: $|W_x| \leq \frac{1}{4m} \cdot 2^m$, and so $|W_x| \left(\frac{|W_x|}{|T|}\right)^{r=m} \leq \frac{1}{4m} 2^m (1/2)^m = \frac{1}{4m}$.)

Again, Lemma 5.7 implies that if we choose $h_1, h_2, \ldots, h_r$ uniformly and independently at random from $\mathcal{H}$,

$$\Pr_{h_1, h_2, \ldots, h_r}[\{h_1, h_2, \ldots, h_r\} \text{ isolates } W_x] \geq 1 - \frac{1}{4m}.$$

In particular, it follows that,

$$\begin{aligned} x \in L &\implies \forall h_1, h_2, \ldots, h_r[\{h_1, h_2, \ldots, h_r\} \text{ does not isolate } W_x] \\ x \notin L &\implies \exists h_1, h_2, \ldots, h_r[\{h_1, h_2, \ldots, h_r\} \text{ isolates } W_x] \end{aligned}$$

So, we have,

$$x \notin L \iff \exists h_1, h_2, \ldots, h_r[\{h_1, h_2, \ldots, h_r\} \text{ isolates } W_x].$$

Given a particular set of $\{h_1, h_2, \ldots, h_r\}$, the predicate "$\{h_1, h_2, \ldots, h_r\}$ isolates W_x" can be expressed as

$$(\forall y \in W_x)(\exists 1 \leq i \leq r)(\forall y' \in W_x - \{y\})[h_i(y) \neq h_i(y')].$$

At first glance, it seems we need to write a Π_3^p predicate to express the above condition. But, we can do better and express it as a coNP predicate, because $\exists 1 \leq i \leq r$ is a bounded quantifier and can be eliminated. This is a general principle in logic, but to be totally concrete, we have $\{h_1, h_2, \ldots, h_r\}$ isolates W_x if and only if

$$(\forall y \in W_x)\{(\forall y_1 \in W_x - \{y\})[h_1(y) \neq h_1(y_1)] \vee \ldots \vee (\forall y_r \in W_x - \{y\})[h_r(y) \neq h_r(y_r)]\}$$

which is equivalent to

$$(\forall y \in W_x)(\forall y_1, \ldots, y_r \in W_x - \{y\})[(h_1(y) \neq h_1(y_1)) \vee \cdots \vee (h_r(y) \neq h_r(y_r)].$$

We conclude that, $x \notin L$ if and only if

$$\begin{aligned} &(\exists h_1, \ldots, h_r)(\forall y \in W_x)(\forall y_1, \ldots, y_r \in W_x - \{y\}) \\ &\quad [(h_1(y) \neq h_1(y_1)) \vee (h_2(y) \neq h_2(y_2)) \vee \cdots \vee (h_r(y) \neq h_r(y_r)]. \end{aligned}$$

We have shown that $L \in \Pi_2^p$. □

5.4.1 Approximate Counting Using Isolation Lemma

Using Sipser's Isolation Lemma, Stockmeyer showed how to do approximate counting in $\mathrm{P}^{\Sigma_2^p}$. In fact, his technique shows approximate counting at the level of $\mathrm{RP}^{\mathrm{NP}}$ already. Let us be more precise.

Definition 5.9 (#P)**.** *#P is a function class. A function $f : \{0,1\}^* \to \mathbf{N}$ is in #P, iff there is a p-time NTM M, such that $f(x) =$ # of accepting paths of $M(x)$.*

So typically any NP language has a "counting" version; e.g., #SAT(φ) is the number of satisfying assignments to the formula φ; #HAM(G) is the number of Hamiltonian circuits in G. One can easily develop a notion of polynomial time reduction for these functions. It is no surprise that both functions #SAT and #HAM are #P-complete under this reduction. This follows from the fact that Cook's reduction is parsimonious, i.e., they preserve the number of solutions. (For instance, in Cook's reduction from a generic NP language to SAT, every accepting computation is in a unique way associated with a satisfying assignment.)

Less obvious, yet also #P-complete, is the permanent function: For any n by n matrix $A = (a_{ij})$,

$$\mathrm{per}(A) = \sum_{\sigma \in S_n} a_{1,\sigma 1} a_{2,\sigma 2} \ldots a_{n,\sigma n},$$

where the sum is over all permutations $\sigma : i \mapsto \sigma i$. In other words, the permanent function is defined much as the determinant function, except there are no more minus signs. For a 0-1 matrix, $\mathrm{per}(A)$ counts the number of perfect matchings of the bipartite graph with matrix A.

Valiant showed that the permanent function is also #P-complete (even though the decision problem of graph matching is in P.)

Definition 5.10 (PP)**.** *PP stands for probabilistic polynomial time. A language L is in* PP*, if there is a boolean predicate $D(\cdot,\cdot)$, computable in deterministic polynomial time such that,*

$$\begin{aligned} x \in L &\implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1] \geq 1/2 \\ x \notin L &\implies \Pr_{y\in\{0,1\}^m}[D(x,y)=1] < 1/2, \end{aligned}$$

where the number of random bits used, m, is polynomial in length of the input $|x|$.

Note that, unlike BPP and RP, every polynomial time NTM defines a PP language, whereas not every polynomial time NTM defines a language in BPP or RP. BPP, RP and ZPP are "promise classes" in the sense that a polynomial time predicate (or equivalently a polynomial time NTM) defines a language in BPP, RP or ZPP only if it satisfies some global conditions. Moreover, these conditions are over all input length n, and not decidable in general. Therefore we do not have an enumeration of these classes simply by an enumeration of their acceptors. This implies that we do not have a universal language, nor a complete language by this process. It is an open problem whether such complete languages exist. (Of course if BPP = P, then they indeed exist.) For PP, one can easily enumerate the class, and therefore complete languages exist. For example, the set of Boolean formulae on n variables having at least 2^{n-1} satisfying assignments is such a language.

It is also easy to see that

Theorem 5.11.

$$\mathrm{P}^{\mathrm{PP}} = \mathrm{P}^{\#P}.$$

Next we discuss approximate counting. Given a formula φ over n variables, the goal is to approximately count the number of satisfying truth assignments of φ. Let $S = \{\sigma | \varphi(\sigma) = 1\}$ be the set of all such assignments. We want to compute the first $c \cdot \log n$ bits of $|S|$, where c is any constant. This can be done by a polynomial time algorithm using a Σ_2^p oracle, as shown by Stockmeyer. In fact, his technique can accomplish the same task with just an NP oracle, if we are ready to settle for a randomized polynomial time algorithm.

Our main tool is the isolation lemma, Lemma 5.7. We first rephrase the lemma in a more suitable format:

Lemma 5.12. *Let $S \subseteq \{0,1\}^n$, and let $\mathcal{H}$ be a family of 2-universal hash functions from $\{0,1\}^n$ to $\{0,1\}^k$. For all $m \geq k$, choose $h_1, h_2, \ldots, h_m$ independently at random from $\mathcal{H}$.*

1. *if $|S| \leq 2^{k-1}$ then*

$$\Pr_{h_1,\ldots,h_m}[\forall x \in S \text{ some } h_i \text{ isolates } x\,] \geq 1 - \frac{1}{2^{m-k+1}}$$

2. *if $|S| > m2^k$ then*

$$\Pr_{h_1,\ldots,h_m}[\forall x \in S \text{ some } h_i \text{ isolates } x\,] = 0.$$

Note that in case 1, for $r = m$, $|S|^{r+1} = |S|^{m+1} \leq 2^{mk+k-m-1} < 2^{mk} = |T|^m = |T|^r$, and so $\frac{|S|^{r+1}}{|T|^r} \leq \frac{2^{mk+k-m-1}}{2^{mk}} = \frac{1}{2^{m-k+1}}$.

The idea is to try all values of k from 1 to n, and attempt to find a k such that $|S| \approx 2^k$. (We may assume our $S \neq \emptyset$; at any rate with one query to SAT we can verify this.) For any $\emptyset \neq S \subseteq \{0,1\}^n$, there is some k_S, where $1 \leq k_S \leq n$, such that $2^{k_S-1} \leq |S| \leq 2^{k_S}$. If we take every k in the range $1 \leq k \leq n+1$, and randomly pick $m = 2n$ hash functions $h_1, \ldots, h_m : \{0,1\}^n \to \{0,1\}^k$, then for each $k \geq k_S + 1$, we would get *isolation* with probability $\geq 1 - \frac{1}{2^n}$. For each k we ask the SAT oracle, whether the chosen set of $h_1, \ldots, h_m$ has the property that "$\forall x \in S$, one of h_i isolates x". Since there are only $m = 2n$ hash functions this is a SAT query. We pick the least k_0 such that the oracle confirms *isolation*. Then $k_0 \leq k_S + 1$, with probability $\geq 1 - \frac{1}{2^n}$. (We abort if for no k the chosen hash functions achieve *isolation*; this happens with exponentially small probability.) Also by the second part of Lemma 5.12, we know definitely $|S| \leq 2n2^{k_0}$. Denote by $U = 2n2^{k_0}$, then with high probability,

$$\frac{U}{8n} \leq |S| \leq U.$$

($\frac{U}{8n} = 2^{k_0-2} \leq 2^{k_S-1} \leq |S|$.)

This give us a randomized polynomial time algorithm using a SAT oracle to approximate $|S|$ within $O(n)$. We'd like to do better. To do this, we use a little trick to amplify the accuracy.

First we build a set S' such that

$$S' = \underbrace{S \times S \times \cdots \times S}_{m \text{ times}} \quad \subseteq \{0,1\}^{nm}$$

where m is polynomial in n. Then we run the previous algorithm on the set S' to get an estimate U' for S' such that

$$\frac{U'}{8nm} \leq |S'| \leq U'.$$

Now we set $e(S) = (U')^{1/m}$. It follows that

$$\frac{e(S)}{(8nm)^{1/m}} \leq |S| \leq e(S).$$

By choosing m to be a sufficiently large polynomial in n, we can get

$$e(S) \cdot \left(1 - \frac{1}{n^c}\right) \leq |S| \leq e(S),$$

for any constant c.

5.5 Unique Satisfiability: Valiant–Vazirani Theorem

Assuming NP $\neq$ P, SAT cannot be solved in polynomial time. One may suspect that it is difficult to design a polynomial time algorithm for SAT because the input formula may have myriad truth assignments and it is hard to get hands on one of them. This suspicion leads to the following interesting problem, called USAT (Unique SAT). We are given a formula φ which is guaranteed to be either unsatisfiable or has exactly one satisfying truth assignment. Is USAT solvable in polynomial time? Here we show some evidence that it is unlikely. We prove that if USAT is solvable in polynomial time, then NP = RP.

Theorem 5.13. *Suppose there is a polynomial time algorithm A, which for a given boolean formula φ, answers*

$$A(\varphi) = \begin{cases} \text{No} & \text{if } \varphi \text{ has no satisfying assignments;} \\ \text{Yes} & \text{if } \varphi \text{ has exactly one satisfying assignment;} \\ \text{Yes/No} & \text{if } \varphi \text{ has more than one satisfying assignments.} \end{cases}$$

Then NP = RP.

Proof. Note that the algorithm A may output anything (YES or NO) if it is given a formula with more than one satisfying truth assignment. Assuming A runs in polynomial time, we design a RP algorithm for SAT. The idea is to use some coin tosses and convert φ into a more constrained formula φ', so that if φ is unsatisfiable, φ' will also be unsatisfiable. And if φ is satisfiable, with non-trivial probability, exactly one of the satisfying assignments of φ will satisfy φ'. Once we are successful in obtaining such a φ', we can apply the algorithm A to it.

To start with assume that φ is satisfiable and let $\#\varphi$ be the number of satisfying truth assignments of φ. In order to convert φ to φ', the algorithm needs an estimate for $\#\varphi$. It is computationally hard to estimate $\#\varphi$. So, we simply pick a k with $1 \leq k \leq n$ at random! The hope is that $2^{k-1} \leq \#\varphi \leq 2^k$. The probability that k satisfies the above condition is at least $1/n$, as long as $\#\varphi \neq 0$. Assume that we are lucky and k indeed satisfies the above condition. Next we choose a set T of size 2^{k+1}. If $2^{k-1} \leq \#\varphi \leq 2^k$, then $2(\#\varphi) \leq |T| \leq 4(\#\varphi)$, meaning $|T|$ is neither too big nor too small compared to $\#\varphi$. We then setup a universal family of hash functions H from the set of all assignments $\{0,1\}^n$ to T, and we randomly pick a hash function h from H and an element α from T. We will show that, with non-trivial probability, there will be a *unique* satisfying truth assignment x such that $h(x) = \alpha$. So, we consider the question: "Is there a truth assignment t such that $\varphi(t) = 1$ and $h(t) = \alpha$". By Cook's theorem this can be converted to a SAT question φ', and it will have a unique satisfying assignment iff there is a unique t satisfying φ, and is mapped to α by h. If we are lucky in choosing the "correct" k, h and α, φ' will have exactly one satisfying truth assignment.

Algorithm for SAT:

Input: A formula φ over n variables.

1. Choose a number $1 \leq k \leq n$ uniformly at random.
2. Let $T = \{0,1\}^{k+1}$, let H be a universal family of hash functions from $\{0,1\}^n$ to T.
3. Choose $h \in H$ and $\alpha \in T$ uniformly at random.
4. Let φ' be the Boolean formula from Cook's theorem, encoding the NP predicate "$(\exists t \in \{0,1\}^n)[(\varphi(t) = 1) \wedge (h(t) = \alpha)]$".
5. Run the procedure A for USAT on φ'. *Accept* φ iff $A(\varphi')$ = accept and the truth assignment extracted using A via self-reducibility indeed satisfies φ.

It is clear that, if φ is unsatisfiable then our algorithm will reject it with probability one. So, assume that φ is satisfiable. Then over the random choices of k, h and α, we show that our algorithm accepts φ with probability $\Omega(1/n)$. This probability can then be easily amplified, making it an RP algorithm.

First of all suppose $2^{k-1} \leq \#\varphi \leq 2^k$. The probability that k satisfies the above condition is at least $1/n$. Under this assumption, we have $2(\#\varphi) \leq |T| \leq 4(\#\varphi)$. We say that two *satisfying* distinct truth assignments a and b (with $a \neq b$) collide under h, if $h(a) = h(b)$. Over the random choices h, let C be the random variable that counts the number of collisions. We first show that the expected number of collisions $\mathrm{E}[C]$ is small. For any fixed $a \neq b$, the probability that a and b

collide is $1/|T|$ (this follows from the definition of universal family of hash functions). For any pair of satisfying truth assignments $a \neq b$, let $X_{a,b}$ be a 0-1 random variable such that $X_{a,b} = 1$, if a and b collide under h and $X_{a,b} = 0$, if they don't collide. Its expectation is $\mathrm{E}[X_{a,b}] = 1/|T|$. The number of collisions C is the sum of over all $X_{a,b}$. So, expectation of C is

$$\mathrm{E}[C] = \sum_{(a,b)} \mathrm{E}[X_{a,b}].$$

The summation above ranges over all distinct pairs of satisfying truth assignments. The number of such pairs is $\binom{\#\varphi}{2}$. Hence,

$$\mathrm{E}[C] = \binom{\#\varphi}{2} \frac{1}{|T|} \leq \frac{\#\varphi}{4}.$$

Using Markov's inequality, we have

$$\Pr\left[C \geq \frac{\#\varphi}{3}\right] \leq \frac{E[C]}{\#\varphi/3} \leq \frac{3}{4}.$$

So, with probability at least $1/4$, the number of collisions C is at most $\#\varphi/3$. In general, if there are c collisions, at most $2c$ satisfying truth assignments can participate in collisions. (This can be shown easily by the inductive argument: Take any such x and pick any $y \neq x$ such that $h(x) = h(y)$. Now remove x and y. There can be at most $c-1$ collisions left among the remaining points, and thus at most $2c-2$ points.) Thus, in our case, at most $2/3 \cdot \#\varphi$ satisfying truth assignments can participate in some collision. It follows that at least $\#\varphi/3$ satisfying truth assignments are mapped to a unique image. Call these $\geq \#\varphi/3$ images in T *good*: i.e., there is a unique satisfying assignment a such that $h(a) = t$. Recall that $|T| \leq 4(\#\varphi)$. So, at least $1/12$ fraction of elements in T are good. Therefore, with probability at least $1/12$, the randomly picked element α is good.

We can now lower bound the probability that φ' has a unique satisfying truth assignment. Assume that φ is satisfiable. Then, with probability $1/n$, the number k chosen by the algorithm satisfies $2^{k-1} \leq \#\varphi \leq 2^k$. Assuming k satisfies the above condition, with probability at least $1/4$, the number of collision C is at most $\#\varphi/3$. Assuming $C \leq \#\varphi/3$, with probability at least $1/12$, the randomly chosen α is good. If α is good, then φ' has exactly one satisfying truth assignment. Putting together,

$$\Pr_{k,h,\alpha}[\varphi' \text{ has a unique satisfying truth assignment}] \geq \frac{1}{n} \cdot \frac{1}{4} \cdot \frac{1}{12} = \frac{1}{48n}$$

As we noted already, if φ is unsatisfiable, our algorithm has zero probability of accepting it. If φ is satisfiable, our algorithm accepts it with probability at least $1/(48n)$. So our algorithm is an RP algorithm with success probability $1/(48n)$. We can amplify this success probability, as usual, by running the algorithm multiple times. For example, by running the algorithm $96n$ times, the success probability can be amplified to $1/2$. □

6 Random Walks

Suppose $G = (V, E)$ is a connected graph of n vertices. We would like to take a random walk on G starting from some arbitrary vertex. How long do we need to walk before we can expect to visit every vertex?

Of course this question depends on what graph, and where we start. The surprising answer is that, *for every connected graph* and *for every starting vertex*, after about $O(n^3)$ steps we can be quite sure that the random walk has visited every vertex.

By a random walk, we mean the following random process. Repeat the following step: At any vertex, randomly and uniformly select a neighbor and move there.

Definition 6.1. *A random walk on a graph $G = (V, E)$, starting at s, is a random sequence $v_0 = s, v_1, v_2, \ldots, v_k, \ldots$, of vertices such that for all $k \geq 0$,*

$$\Pr[v_{k+1} = w \mid v_k = u] = \begin{cases} \frac{1}{\deg(u)} & \text{if } \{u, w\} \in E \\ 0 & \text{otherwise} \end{cases}$$

As an example, consider a graph consisting of an $n/2$-clique (i.e., $n/2$ vertices with all possible edges present) connected at one of its vertices v to one end of an $n/2$-chain. Now start at a vertex s in the clique, and let t be at the opposite end of the chain from v. Note that, even when the walk reaches v, the probability that the walk escapes the clique and enters the chain is only $2/n$. Even if the walk does escape the clique and move a few vertices toward t, it is much more likely to fall back into the clique than it is to reach t. Incredibly, even in this pathological case, the random walk is likely to visit every vertex in no more than $O(n^3)$ steps.

Exercise: What is the expected number of steps for a random walk to reach t from s? What about for an n-chain with s and t at opposite ends?

6.1 Random Commute

Definition 6.2. *A commute from i to j is a path that starts at i and ends the first time it returns to i after having visited j. A random commute is a random walk that forms a commute.*

Consider the example of a chain of 9 vertices, labeled 1 to 9 from left to right.

A commute from 2 to 3 on this chain might look like $2, 1, 2, 1, 2, 3, 4, 5, 4, 5, 4, 3, 2$. Note that it may visit i multiple times before encountering j, and then j multiple times before its return to i.

Definition 6.3. *For all i, j, u, v, with $\{u, v\} \in E$, let c_{ijuv} be a random variable that is the number of times a random commute from i to j crosses the edge (u, v) (in the direction from u to v). Let*

$$\theta_{ijuv} = \mathbb{E}(c_{ijuv}),$$

be the expectation of c_{ijuv}.

Example: For the chain graph of 9 nodes, $\theta_{2323} = 1$. This clear because starting at 2 we can ignore all steps before it crosses $(2, 3)$ (which happens with probability one eventually) in the direction from 2 to 3. And then we can ignore all successive steps until it moves from 3 to 2 (which also happens with probability one eventually). At that point the random commute ends.

Also $\theta_{2343} = 1$. This can be seen as follows: Wait at 3 until the random walk reaches 3 for the first time (which happens with probability one eventually). Then $\Pr[(3, 4) \text{ is crossed}] = 1/2$, and if this happens, then eventually random walk crosses $(4, 3)$ back to 3. If the random walk goes back from 3 to 2 instead, we wait for it to appear again at 3.

So this process is probabilistically exactly the following process: Keep tossing a fair coin, count how many consecutive Heads appear before the first Tail appears. Thus $\theta_{2343} = 1$, Since this is $\theta_{2343} = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot [1 + \theta_{2343}]$.

It will be shown that, surprisingly $\theta_{2389} = 1$ also.

We present a theorem due to Göbel and Jagers. First, a simple lemma.

Lemma 6.4. *For any vertices i, j, and u, θ_{ijuv} is the same for all neighbors v of u.*

Imagine extending the random commute into an infinite process. Let $y_0, y_1, y_2, \ldots$ be an infinite sequence of random vertices, where we let the walk stay at i after the random commute has ended at i. More precisely, we define a $y_0, y_1, y_2, \ldots$ where a (unique) prefix $y_0, y_1, y_2, \ldots, y_m$ is a random commute from i to j, and for which $y_k = i$ for all $k > m$.

Define

$$X_{kuv} = \begin{cases} 1 & \text{if } y_k = u \text{ and } y_{k+1} = v \\ 0 & \text{otherwise} \end{cases}$$

This is like a sentry sitting at the kth step and records whether it is the edge (u, v). Clearly

$$c_{ijuv} = \sum_{k=0}^{\infty} X_{kuv}.$$

Then

$$\begin{aligned} \theta_{ijuv} &= \mathbb{E}[c_{ijuv}] \\ &= \sum_{k=0}^{\infty} \Pr[X_{kuv} = 1] \\ &= \sum_{k=0}^{\infty} \Pr[y_k = u \wedge y_{k+1} = v] \\ &= \sum_{k=0}^{\infty} \Pr[y_k = u] \Pr[y_{k+1} = v \mid y_k = u] \\ &= \sum_{k=0}^{\infty} \Pr[y_k = u] \frac{1}{\deg(u)} \end{aligned}$$

This last quantity is independent of v, i.e., independent of which neighbor of u it is.

The next lemma is the key step.

Lemma 6.5. *For any vertices i, j, u, and v,*

$$\theta_{ijuv} = \theta_{ijvu}.$$

We define a 1-1 correspondence of the set C_{ij} of all commutes from i to j to itself C_{ij}, such that all corresponding pairs have the same value for c_{ijuv} and c_{ijvu} respectively, and with the same probability weight.

For any commute C from i to j, We define C^R as in the figure. We observe that

1. C^R is a commute from i to j.
2. $(C^R)^R = C$.
3. The probability weight of C^R as a random commute is the same as that of C. This is because the weight is just the product of all reciprocals of the degrees of the vertices appearing on this circular path.

4. Let $N(C, u, v)$ be the number of times C crosses the edge (u, v). Then $N(C, u, v) = N(C^R, v, u)$. Hence, let R_{ij} be a random commute,

$$\begin{aligned}
\theta_{ijuv} &= \sum_C \Pr[R_{ij} = C] N(C, u, v) \\
&= \sum_C \Pr[R_{ij} = C^R] N(C^R, v, u) \\
&= \sum_{C^R} \Pr[R_{ij} = C^R] N(C^R, v, u) \\
&= \theta_{ijvu}.
\end{aligned}$$

Theorem 6.6. *For any connected, undirected graph $G = (V, E)$, and for any two vertices $i, j \in V$, θ_{ijuv} is independent of u and v (i.e., it is the same for all edges $\{u, v\}$ and each of the two directions could be crossed).*

Proof. Let $\{u, v\}$ and $\{u', v'\}$ be two edges, and let $P = (u, v, w, \ldots, u', v')$ be a path containing both. Such a path exists because G is connected. As $\{u, v\}$ and $\{u', v'\}$ are listed as unordered pairs as edges, we may assume such a path has this form, i.e., first u then v, and finally u' and then v'. Applying the lemmas above to P alternately, we have the following:

$$\theta_{ijuv} = \theta_{ijvu} = \theta_{ijvw} = \theta_{ijwv} = \ldots = \theta_{iju'v'}.$$

The following theorem is due to Aleliunas, Karp, Lipton, Lovász, and Rackoff.

Theorem 6.7. *Let $G = (V, E)$ be a connected, undirected graph with $|V| = n$ and $|E| = m$. For any starting vertex, the expected number of steps that a random walk needs in order to visit every vertex in V is at most $4nm$.*

Proof. Consider an infix traversal of any spanning tree of G. (Such a spanning tree exists because G is connected.) We show that the expected time to visit the vertices of G in the order given by this traversal is no more than $4nm$.

Let T_{ij} be the expected number of steps for a random walk to get from vertex i to vertex j, where i and j are adjacent in the spanning tree. Certainly $T_{ij} \leq$ the expected number of steps in a random commute from i to j, which is equal to the expected total number of crossings (in both directions) of all edges in a random commute from i to j. That is,

$$\begin{aligned}
T_{ij} &\leq \sum_{(u,v) \in V \times V, \{u,v\} \in E} \theta_{ijuv} \\
&\leq \sum_{(u,v) \in V \times V, \{u,v\} \in E} \theta_{ijij} \\
&\leq \sum_{(u,v) \in V \times V, \{u,v\} \in E} 1 \\
&\leq 2m.
\end{aligned}$$

Note that $\theta_{ijij} \leq 1$ because it is impossible for a random commute to traverse the edge (i, j) twice! (If it is about to traverse the edge (i, j) the second time, it should have ended.)

There are $n-1$ edges in a spanning tree, and each is traversed twice in an infix walk, so the expected number of steps in the entire walk is at most $2(n-1)2m < 4nm$.

Now we can use this random walk idea to decide in probabilistic logarithmic space whether a given undirected graph is connected or not.

Exercise: Carry out the proof details of the above claim.

7 Expanders

Let $G = (V, E)$ be a d-regular graph on n vertices. Let $p \in \mathbb{R}^n_{\geq 0}$ be a probability distribution on V. We assume each $v \in V$ has a self-loop. A random walk repeatedly takes a neighbor chosen uniformly at random (u.a.r.) and moves to that neighbor (it also stays put with probability $1/d$). Let A be the random-walk matrix, which is $1/d$ of the symmetric adjacency matrix, i.e., $A_{i,j} = 1/d$ if $\{i, j\}$ is an edge, and 0 otherwise. If we start with the distribution p and take one step of the random walk, then the probability distribution is Ap.

Definition 7.1. *Let $\mathbf{1} = \frac{1}{n}(1, 1, \ldots, 1)^\intercal \in \mathbb{R}^n$ be the uniform distribution on V. Denote by $\mathbf{1}^\perp = \{v \in \mathbb{R}^n \mid \langle v, \mathbf{1}\rangle = 0\}$ be the set of vectors orthogonal to $\mathbf{1}$, where $\langle u, v\rangle = \sum_{i=1}^n u_i v_i$. Then*

$$\lambda(G) = \lambda(A) = \max_{0 \neq p \in \mathbf{1}^\perp} \frac{||Ap||}{||p||},$$

where $||\cdot||$ is the 2-norm $||p|| = \sqrt{\sum_{i=1}^n |p_i|^2}$.

As $||cp|| = |c| \cdot ||p||$ for any scalar c, the definition is the same if we restrict to $||p|| = 1$, or to $||p||_1 = \sum_{i=1}^n |p_i| = 1$.

This quantity $\lambda(G)$ is called the second largest eigenvalue (of the random-walk matrix) of the graph G. As A is real symmetric, it has n eigenvalues counting multiplicity: $\lambda_1, \lambda_2, \ldots, \lambda_n$, with n orthogonal eigenvectors $v_1, v_2, \ldots, v_n$. We may assume they are ordered such that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|.$$

Then $\lambda(G) = |\lambda_2|$. It can be shown that the maximum eigenvalue in absolute value is $|\lambda_1| = 1$, with a corresponding eigenvector $\mathbf{1}$. G is bipartite iff $\lambda_2 = -1$.

Exercise: Prove that For any graph G, he maximum $|\lambda_1| = 1$, with a corresponding eigenvector $\mathbf{1}$. And G is bipartite iff $\lambda_2 = -1$.

The gap $1 - \lambda(G)$ controls how fast a random walk mixes, or converges to the uniform distribution.

Lemma 7.2. *For any distribution p on V,*

$$||A^k - \mathbf{1}|| \leq \lambda(G)^k.$$

Proof. For any $v \in \mathbf{1}^\perp$, $||Av|| \leq \lambda(G)||v||$. Note that $\mathbf{1}^\perp$ is an invariant subspace of A (spanned by the subset of the orthogonal basis $\{v_2, \ldots, v_n\}$.) This is because $\langle Av, \mathbf{1}\rangle = (Av)^\intercal \mathbf{1} = v^\intercal A\mathbf{1} = v^\intercal \mathbf{1} = 0$, where we used the fact that A is symmetric and $A\mathbf{1} = \mathbf{1}$ since A is a random walk matrix (so the row sum is 1). As the eigenvalues of A^k are just $\lambda_1^k, \lambda_2^k, \ldots, \lambda_n^k$, clearly $\lambda(A^k) = |\lambda_2|^k = \lambda(A)^k$.

Write any distribution p as a decomposition along $\mathbf{1}$ and orthogonal to $\mathbf{1}$,

$$p = \alpha\mathbf{1} + p'.$$

Being a distribution, the sum of the coordinates $\sum_{i=1}^n p_i = 1$. On the RHS, it is $\alpha + \sum_{i=1}^n p'_i = \alpha$, because $\sum_{i=1}^n p'_i = 0$. So we have $p = \mathbf{1} + p'$. then $A^k p = \mathbf{1} + A^k p'$, and

$$||A^k p - \mathbf{1}|| \leq \lambda(A)^k ||p'||.$$

Note that p' is an orthogonal projection of p, we have $||p||^2 = ||\mathbf{1}||^2 + ||p'||^2 = \frac{1}{n} + ||p'||^2 > ||p'||^2$. Also $||p||^2 = \sum_{i=1}^n |p_i|^2 \leq (\sum_{i=1}^n |p_i|)^2 = ||p||_1^2 = 1$, we get $||p'|| \leq 1$. Hence

$$||A^k p - \mathbf{1}|| \leq \lambda(G)^k.$$

□

Lemma 7.3. *If G is a d-regular connected graph with a self-loop at every vertex, then $\lambda(G) \leq 1 - \Theta(\frac{1}{n^2})$.*

Proof. Take $\epsilon = \frac{1}{6n^2}$. We want to prove that for any unit vector $u \in \mathbf{1}^\perp$, $1 - ||Au||^2 \geq \epsilon$. If so, then $||Au|| \leq 1-\epsilon/2$, and the lemma is proved. To wit: if $||Au|| > 1-\epsilon/2$, then $||Au||^2 > (1-\epsilon/2)^2 > 1-\epsilon$.

Write $v = Au$. As $||u|| = 1$, we have $1 - ||Au||^2 = ||u||^2 - ||v||^2$. Consider

$$\sum_{i,j=1}^n A_{ij}(u_i - v_j)^2 = \sum_{i,j=1}^n A_{ij}(u_i^2 - 2u_i v_j + v_j^2) = \sum_i u_i^2 - 2\sum_{i,j} A_{ij} u_i v_j + \sum_j v_j^2.$$

The sum $\sum_{i,j} A_{ij} u_i v_j$ is $\langle Au, v\rangle = ||v||^2$. Hence

$$\sum_{i,j=1}^n A_{ij}(u_i - v_j)^2 = ||u||^2 - ||v||^2.$$

So we want to show this sum is $\geq \epsilon$.

Since u is a unit vector with all entries sum to 0, some entries are positive and some are negative, with at least one $|u_k| \geq \frac{1}{\sqrt{n}}$. (Otherwise $||u||^2 = \sum_i u_i^2 < 1$.) So there is a pair of indices i and j such that $u_i - u_j \geq \frac{1}{\sqrt{n}}$. Since G is connected, there is a path from i to j. Suppose the path has length D, and we rename the vertices so that the path is $1, 2, \ldots, D+1$ (with $i = 1$ and $j = D+1$). Consider the telescoping sum

$$\frac{1}{\sqrt{n}} \leq u_1 - u_{D+1} = (u_1 - v_1) + (v_1 - u_2) + (u_2 - v_2) + (v_2 - u_3) + \ldots + (v_D - u_{D+1}).$$

(We first telescope with $u_1, u_2, \ldots, u_D$, then insert v_i between $u_i - u_{i+1}$.) This sum has $2D$ terms. So

$$\frac{1}{\sqrt{n}} \leq |u_1 - v_1| + |v_1 - u_2| + |u_2 - v_2| + |v_2 - u_3| + \ldots |v_D - u_{D+1}| \tag{7.1}$$

$$\leq \sqrt{|u_1 - v_1|^2 + |v_1 - u_2|^2 + |u_2 - v_2|^2 + |v_2 - u_3|^2 + \ldots |v_D - u_{D+1}|^2}\sqrt{2D} \tag{7.2}$$

Since the graph has self-loops, and every pair $\{i, i+1\}$ $(1 \le i \le D)$ is an edge, in the sum $\sum_{i,j=1}^{n} A_{ij}(u_i - v_j)^2$ every term of the last sum in (7.2) appears with coefficient $A_{ij} = 1/d$. All other terms in $\sum_{i,j=1}^{n} A_{ij}(u_i - v_j)^2$ are nonnegative, so

$$\sum_{i,j=1}^{n} A_{ij}(u_i - v_j)^2 \ge \frac{1}{2Ddn}.$$

This already gives an inverse polynomial $\Omega(1/n^3)$ gap.

For d-regular graphs without parallel edges, one can do better to get an $\Omega(1/n^2)$ gap. We observe that $D \le \frac{3n}{d}$, which completes the proof. To see that $D \le \frac{3n}{d}$, consider any pair (i, j) achieving the longest distance among any two vertices (called the diameter of G.) Thus we have a path $i = i_0, i_1, \ldots, i_D = j$, which is the shortest path between them, of length D. Each of $i_0, i_3, i_6, \ldots$ (there are at least $D/3$ of them) has $d-3$ neighbors that form pairwise disjoint subsets among the vertices not on the path: $[n] - \{i_0, i_1, \ldots, i_D\}$, ($d-3$ is counting the self-loop and the two neighbors; the end point(s) have $d-2$ such neighbors). So

$$\frac{D}{3}(d-3) \le n - D.$$

This gives $D \le \frac{3n}{d}$.

7.1 Algebraic and Combinatorial Notions of Expanders

Definition 7.4. *An (n, d, λ)-expander is a d-regular graph $G = (V, E)$ on n vertices, such that $\lambda(G) \le \lambda$.*

We typically look for a graph family $\{G_n\}$, for some fixed d and fixed $\lambda < 1$, with $n \to \infty$.

Definition 7.5. *An n-vertex d-regular graph $G = (V, E)$ is called an (n, d, ρ)-(combinatorial-edge)-expander if for every subset $S \subset V$ of $|S| \le n/2$,*

$$|E(S, \overline{S})| \ge \rho d|S|,$$

where $E(S, T) = E \cap (S \times T)$ is the set of edges of the form (s, t) for $s \in S$ and $t \in T$, and $\overline{S}$ is the complement of S.

Using the probabilistic method, one can prove the existence of expanders.

Exercise: Pick d permutations $\sigma_1, \ldots, \sigma_d$ from S_n u.a.r. Define a graph on $[n]$ such that (i, j) is an edge iff some σ_k maps i to j or j to i. Prove that, for constant $d \ge 2$ and some constant $\rho > 0$, this defines a $2d$-regular expander graph family as $n \to \infty$, with high probability.

Theorem 7.6. *If G is an (n, d, λ)-expander, then it is an (n, d, ρ)-(combinatorial-edge)-expander, with $\rho = (1 - \lambda)/2$.*

If G is an (n, d, ρ)-(combinatorial-edge)-expander, then its second largest eigenvalue (without taking absolute value) is at most $1 - \frac{\rho^2}{2}$. If furthermore G has all self-loops, then it is an $(n, d, 1-\epsilon)$-expander, where $\epsilon = \min\{2/d, \rho^2/2\}$.

We prove these two statements of the theorem by the following two lemmas.

Lemma 7.7. *If G is an (n, d, λ)-expander, then for any $S \subseteq V$,*

$$|E(S, \overline{S})| \geq (1 - \lambda)\frac{d|S||\overline{S}|}{n}.$$

Proof. Define a vector $v \in \mathbb{R}^n$ where

$$v_i \quad = \quad \begin{cases} |\overline{S}| & \text{if } i \in S \\ -|S| & \text{if } i \notin S. \end{cases}$$

Then $||v||^2 = |S| \cdot |\overline{S}|^2 + |\overline{S}| \cdot |S|^2 = |S||\overline{S}|n$. Also $v \in \mathbf{1}^\perp$.

Let $Z = \sum_{i,j=1}^n A_{ij}(v_i - v_j)^2$. Only those pairs (i, j) between S and $\overline{S}$ give a nonzero value for $(v_i - v_j)^2$, which is $(|S| + |\overline{S}|)^2 = n^2$. For every edge between S and $\overline{S}$, A_{ij} contributes $2/d$ (it occurs twice in the sum). So, $Z = \frac{2}{d}|E(S, \overline{S})|n^2$.

On the other hand, if we expand out Z, we get

$$Z = \sum_{i,j}[A_{ij}v_i^2 - 2A_{ij}v_iv_j + A_{ij}v_j^2] = 2||v||^2 - 2\langle Av, v\rangle.$$

(We used the fact that the sum of each row and column of A is 1.) Since $x \in \mathbf{1}^\perp$, we have $||Av|| \leq \lambda||v||$, and $|\langle Av, v\rangle| \leq ||Av||||v|| \leq \lambda||v||^2$. So

$$\frac{1}{d}|E(S, \overline{S})|n^2 \geq (1 - \lambda)||v||^2 = (1 - \lambda)|S||\overline{S}|n.$$

This proves the lemma.

The quantity $\frac{1}{2}Z = \frac{1}{2}\sum_{i,j=1}^n A_{ij}(x_i - x_j)^2 = ||x||^2 - \langle Ax, x\rangle$ as a quadratic form, is also called the *Laplacian*, and can be expressed as $\langle x, (I - A)x\rangle$. The matrix $I - A$ is also called the *Laplacian*. It plays an important role.

Lemma 7.8. *If G is an (n, d, ρ)-(combinatorial-edge)-expander, then its second largest eigenvalue (without taking absolute value) is at most $1 - \frac{\rho^2}{2}$.*

Proof. Let A be the random walk matrix of G, and let λ' be the second largest eigenvalue without taking absolute value. So there is some unit vector $u \perp \mathbf{1}$ such that $Au = \lambda' u$. Write $u = v + w$ where v is the positive part and w is the negative part of u, i.e., $v_i = u_i$ if $u_i \geq 0$ and 0 otherwise, and similarly for w_i. Both v and $w \neq 0$. By exchanging u with $-u$ we may assume at most $n/2$ entries of v are nonzero. Let $Z = \sum_{i,j} A_{ij}|v_i^2 - v_j^2|$.

Claim 1: $Z \geq 2\rho||v||^2$.

To prove Claim 1: Sort v_i so that $v_1 \geq v_2 \geq \ldots \geq v_n$, where $v_i = 0$ for $i > n/2$. Then $v_i^2 - v_j^2 = \sum_{k=i}^{j-1}(v_k^2 - v_{k+1}^2)$,

$$Z = \sum_{i,j} A_{ij}|v_i^2 - v_j^2| = 2\sum_{i<j} A_{ij}\sum_{k=i}^{j-1}(v_k^2 - v_{k+1}^2).$$

Every term $(v_k^2 - v_{k+1}^2)$ appears in this sum once (with weight $2/d$) for each edge $\{i, j\}$ such that $i \leq k < j$. Since $v_k = 0$ for $k > n/2$, we have, by edge expansion,

$$Z = \frac{2}{d}\sum_{1\leq k\leq n/2} |E(\{1, \ldots, k\}), (\{k + 1, \ldots, n\})|(v_k^2 - v_{k+1}^2) \geq \frac{2}{d}\sum_{1\leq k\leq n/2} \rho dk(v_k^2 - v_{k+1}^2).$$

By telescoping,

$$Z \geq \frac{2}{d}\rho d \sum_{1\leq k\leq n/2} k(v_k^2 - v_{k+1}^2) = 2\rho \sum_{1\leq k\leq n} (kv_k^2 - (k-1)v_k^2) = 2\rho \sum_{1\leq k\leq n} v_k^2 = 2\rho||v||^2.$$

Claim 2: $Z \leq \sqrt{8(1-\lambda')}||v||^2$.

To prove Claim 2: Since $Au = \lambda' u$, and since $\langle v, w\rangle = 0$,

$$\langle Av, v\rangle + \langle Aw, v\rangle = \langle A(v+w), v\rangle = \langle Au, v\rangle = \lambda'\langle v+w, v\rangle = \lambda'||v||^2.$$

Since all entries of Aw are all nonpositive, and all entries of v are nonnegative, $\langle Aw, v\rangle \leq 0$. And so $\lambda'||v||^2 \geq \langle Av, v\rangle$. Hence

$$1 - \lambda' \geq 1 - \frac{\langle Av, v\rangle}{||v||^2} = \frac{||v||^2 - \langle Av, v\rangle}{||v||^2} = \frac{\sum_{i,j} A_{ij}(v_i - v_j)^2}{2||v||^2},$$

the last equality is by expanding the sum $\sum_{i,j} A_{ij}(v_i - v_j)^2$.

Multiply both the numerator and denominator by $\sum_{i,j} A_{ij}(v_i + v_j)^2$, the numerator is

$$\left(\sum_{i,j} A_{ij}(v_i - v_j)^2\right)\left(\sum_{i,j} A_{ij}(v_i + v_j)^2\right) \geq \left(\sum_{i,j} A_{ij}|v_i - v_j| \cdot |v_i + v_j|\right)^2$$

by applying Cauchy-Schwarz to $\sqrt{A_{ij}}|v_i - v_j|$ and $\sqrt{A_{ij}}|v_i + v_j|$. Hence, using $(a+b)(a-b) = a^2 - b^2$,

$$1 - \lambda' \geq \frac{\left(\sum_{i,j} A_{ij}|v_i^2 - v_j^2|\right)^2}{2||v||^2 \sum_{i,j} A_{ij}(v_i + v_j)^2} = \frac{Z^2}{2||v||^2\left(\sum_{i,j} A_{ij}v_i^2 + 2\sum_{i,j} A_{ij}v_i v_j + \sum_{i,j} A_{ij}v_j^2\right)}.$$

Using the fact that the sum of each row and column of A is 1, we get the denominator

$$2||v||^2(2||v||^2 + 2\langle Av, v\rangle) \leq 8||v||^4.$$

Here we used the fact that, for any v, $||Av|| \leq ||v||$, since the maximum eigenvalue of A is 1. Combining these we have proved Claim 2.

Claim 1 and 2 prove the Lemma 7.8.

The "furthermore" part of the theorem is obtained by adding the self-loops to a $(d-1)$-regular graph with the random walk matrix A. The modified random walk matrix becomes $\frac{d-1}{d}A + \frac{1}{d}I$, which has the same set of eigenvectors as A, and the eigenvalues are $\frac{d-1}{d}\lambda_i(A) + \frac{1}{d}$. The minimum eigenvalue (without absolute value) of A is ≥ -1, the new minimum eigenvalue is $\geq -\frac{d-1}{d} + \frac{1}{d} = -1 + \frac{2}{d}$. This proves that (after taking absolute value) the second largest eigenvalue $\lambda(G) \leq 1 - \min\{2/d, \rho^2/2\}$. This finishes the proof of Theorem 7.6.

8 Random Walk on an Expander

Theorem 8.1. *Let G be an (N, d, λ)-expander, and let $B \subset [N]$ be any subset with $|B| \leq \beta N$, for some $\beta < 1$. Let $X_0, X_1, \ldots, X_k$ be random variables denoting a k-step random walk, starting with a vertex X_0 chosen u.a.r. from $[N]$. Then*

$$\Pr\left[\bigwedge_{i=0}^{k}(X_i \in B)\right] \leq (\sqrt{\beta} + \lambda)^k.$$

Note that for small $\beta < 1$ and $\lambda < 1$, the quantity $\sqrt{\beta} + \lambda$ is a small constant < 1. With a little more care we can replace $\sqrt{\beta} + \lambda$ by $(1-\lambda)\sqrt{\beta} + \lambda$. For $\beta < 1$ and $0 \leq \lambda < 1$, the quantity $(1-\lambda)\sqrt{\beta} + \lambda$ is a weighted average of $\sqrt{\beta}$ (< 1) and 1, and hence $(1-\lambda)\sqrt{\beta} + \lambda < 1$.

Proof. For $0 \leq i \leq k$, let B_i be the event $[X_i \in B]$. Then

$$\Pr\left[\bigwedge_{i=0}^{k}(X_i \in B)\right] = \Pr[B_0] \cdot \Pr[B_1|B_0] \cdot \Pr[B_2|B_0 \wedge B_1] \cdots \Pr[B_k| \bigwedge_{0 \leq i \leq k-1} (X_i \in B)].$$

Let $M = M_B$ be the projection matrix on B, namely, $M_{ij} = 1$ for all $i = j \in B$, and 0 otherwise. We make two simple Observations:

1. For any probability distribution $p \in \mathbb{R}^N_{\geq 0}$, Mp records the probability on each $i \in B$, and $\frac{Mp}{||Mp||_1}$ is the conditional probability distribution under the condition that it is in B, since $||Mp||_1$ is the probability to be in B.
2. In particular, if we start with p and take one step in the random walk, and denote the location after this step by the random variable Y, then the conditional probability distribution of Y under the condition that $Y \in B$ is given by $\frac{MAp}{||MAp||_1}$.

Thus, Observation 1 and 2 are to change p to the $||\cdot||_1$-normalized versions of Mp and MAp respectively for the resulting distributions. Note that the formulas above are invariant if we replaced the vector p by a positive scalar multiple cp, for any $c > 0$.

Now we prove the following inductively for $i \geq 1$:

The conditional probability

$$\Pr[B_i|B_0 \wedge \ldots \wedge B_{i-1}] = \frac{||(MA)^i M\mathbf{1}||_1}{||(MA)^{i-1}M\mathbf{1}||_1},$$

and the conditional distribution of X_i, under the condition $\bigwedge_{0 \leq j \leq i} B_j$, is

$$\frac{(MA)^i M\mathbf{1}}{||(MA)^i M\mathbf{1}||_1}.$$

To prove this inductively, we observe that the distribution of X_0 is $\mathbf{1}$, and the conditional distribution of X_0 under the condition B_0 is $\frac{M\mathbf{1}}{||M\mathbf{1}||_1}$ (by Observation 1), which is a positive scalar multiple of $M\mathbf{1}$. The distribution of X_1 (conditioned on B_0) is

$$\frac{AM\mathbf{1}}{||M\mathbf{1}||_1}.$$

Then $\Pr[B_1|B_0] = \frac{||MAM\mathbf{1}||_1}{||M\mathbf{1}||_1}$. By Observation 2, the conditional distribution of X_1 conditioned on B_1 (and on B_0) is the $||\cdot||_1$-normalized $MA(M\mathbf{1})$, which is $\frac{MAM\mathbf{1}}{||MAM\mathbf{1}||_1}$.

For general $i > 1$, start with the distribution of X_i (conditioned on $B_0 \wedge \ldots \wedge B_i$) which is a positive multiple of $(MA)^i M\mathbf{1}$, by Observation 2, the conditional distribution of X_{i+1} under the additional condition B_{i+1} (as well as $B_0 \wedge \ldots \wedge B_i$) is

$$\frac{(MA)^{i+1} M\mathbf{1}}{||(MA)^{i+1} M\mathbf{1}||_1}.$$

The conditional distribution of X_{i+1} under the condition $B_0 \wedge \ldots \wedge B_i$ is

$$\frac{A(MA)^i M\mathbf{1}}{||(MA)^i M\mathbf{1}||_1}.$$

Then, the probability of B_{i+1} under the condition $B_0 \wedge \ldots \wedge B_i$ is

$$\Pr[B_{i+1}|B_0 \wedge \ldots \wedge B_i] = \frac{||(MA)^{i+1} M\mathbf{1}||_1}{||(MA)^i M\mathbf{1}||_1},$$

completing the induction.

Hence,

$$\begin{aligned}
\Pr\left[\bigwedge_{i=0}^{k}(X_i \in B)\right] &= \Pr[B_0] \cdot \Pr[B_1|B_0] \cdot \Pr[B_2|B_0 \wedge B_1] \cdots \Pr[B_k| \bigwedge_{0\le i\le k-1} (X_i \in B)] \\
&= ||M\mathbf{1}||_1 \cdot \frac{||(MA)M\mathbf{1}||_1}{||M\mathbf{1}||_1} \cdots \frac{||(MA)^i M\mathbf{1}||_1}{||(MA)^{i-1} M\mathbf{1}||_1} \cdots \frac{||(MA)^k M\mathbf{1}||_1}{||(MA)^{k-1} M\mathbf{1}||_1} \\
&= ||(MA)^k M\mathbf{1}||_1
\end{aligned}$$

By Cauchy-Schwarz we have

$$||(MA)^k M\mathbf{1}||_1 \le ||(MA)^k M\mathbf{1}||\sqrt{N} \qquad \text{(This latter norm is the 2-norm.)}.$$

Hence,

$$||(MA)^k M\mathbf{1}||_1 \le ||MA||^k \cdot ||M|| \cdot ||\mathbf{1}|| \cdot \sqrt{N} = ||MA||^k \cdot ||M||,$$

as $||\mathbf{1}|| = \frac{1}{\sqrt{N}}$.

For any $x \in \mathbb{R}^N$, Mx just truncates x to its $|B|$ entries in B, thus clearly $||Mx|| \le ||x||$, and so $||M|| \le 1$.

The purpose of transferring the estimate from 1-norm to 2-norm is to use the orthogonal spectral decomposition. Let $v_1 = \frac{1}{\sqrt{N}}(1, 1, \ldots, 1)^\intercal, v_2, \ldots, v_N$ be a set of orthonormal eigenvectors of A, with eigenvalues $\lambda_1 = 1, \lambda_1, \ldots, \lambda_N$ where except $\lambda_1 = 1$, all other $|\lambda_i| \le \lambda(G)$. Take any $x \in \mathbb{R}^N$. Then $x = \sum_{i=1}^N \alpha_i v_i$, where $\alpha_i = \langle x, v_i\rangle$, and they satisfy $\sum_{i=1}^N \alpha_i^2 = ||x||^2$.

$$MAx = MA\alpha_1 v_1 + M\sum_{i=2}^{N} \alpha_i A v_i = \alpha_1 M v_1 + M\sum_{i=2}^{N} \alpha_i \lambda_i v_i.$$

Note that $||Mv_1||^2 = \frac{|B|}{N} \le \beta$. Also $|\alpha_1| \le ||x||$. For the second part, the components $\alpha_i\lambda_i v_i$ are mutually orthogonal, thus

$$\left|\left|\sum_{i=2}^{N} \alpha_i \lambda_i v_i\right|\right|^2 = \sum_{i=2}^{N} |\alpha_i|^2 |\lambda_i|^2 \le \lambda(G)^2 \sum_{i=2}^{N} |\alpha_i|^2 \le \lambda(G)^2 \sum_{i=1}^{N} |\alpha_i|^2 = \lambda(G)^2 ||x||^2.$$

So

$$||MAx|| \le (\beta + \lambda(G)) \cdot ||x||.$$

This proves the theorem. □

If $L \in \text{RP}$ where we can test membership $w \in L$ by using a random string $s \in \{0,1\}^n$, with error probability $\leq 1/2$. If we repeat this probabilistic process by running k independent runs, using a total of kn random bits, we can achieve error probability $\leq 1/2^k$. Instead of running on k independent batches of n random bits each, we can use n bits to select a point in $\{0,1\}^n$, and then impose an expander structure on $V = \{0,1\}^n$, with constant degree, and $\lambda < 1$. Then, to take successive probabilistic tests for membership $w \in L$ we take a random walk on the expander. Each point is a potential "random" test string $s' \in V = \{0,1\}^n$, and each successive step only takes $O(1)$ bits. Thus we use a total of $n + O(k)$ bits. Theorem 8.1 says that this expander walk achieves error probability $\leq 2^{-\Theta(k)}$. Thus we can achieve error probability $\leq 1/2^k$ using only $O(n+k)$ bits.

9 Explicit Construction of Expanders

We take two graphs G and H, where G is an (n, d, λ)-expander and H is an (n', d', λ')-expander. Let A and B be their random walk matrices respectively. If we form the product BA, then this represents a directed graph of out-degree dd'. Each edge corresponds to taking a step in G followed by a step in H. To make it undirected (symmetric), we can take BAB or ABA. It can be easily verified that $\lambda(ABA) \leq \lambda(A)\lambda(B)\lambda(A)$, because they share an eigenvector $\mathbf{1}$, and $\mathbf{1}^\perp$ are invariant under both A and B.
Exercise: Prove that $\lambda(ABA) \leq \lambda(A)\lambda(B)\lambda(A)$.

A more interesting product is the tensor product $\otimes$. To make it even more useful (in low level complexity setting, such as logspace computation) we define a rotation map for a graph G as follows: We consider at each vertex $v \in [n]$, its incident edges are labeled by $[d]$, and the rotation map

$$\text{Rot}_G : [n] \times [d] \to [n] \times [d]$$

gives the neighboring information more explicitly, including the local labeling of the edges:

$$\text{Rot}_G(v, i) = (u, j),$$

where the ith edge of G at v leads to the vertex u, and that edge is labeled as the jth edge at u. Thus in the reverse direction, $\text{Rot}_G(u, j) = (v, i)$.

Now the tensor product graph $G \otimes H$ is defined as follows: The vertex set is $[n] \times [n']$. Its edges are labeled by $[d] \times [d']$, and

$$\text{Rot}_{G\otimes H}((v, v'), (i, i')) = ((u, u'), (j, j')),$$

where $\text{Rot}_G(v, i) = (u, j)$ and $\text{Rot}_G(v', i') = (u', j')$.

In terms of random walk matrices, the random walk matrix of $G \otimes H$ is just the tensor product of the random walk matrices A and B.

$$\begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \dots & a_{nn}B \end{pmatrix}$$

Lemma 9.1.

$$\lambda(G \otimes H) \leq \max\{\lambda(G), \lambda(H)\}.$$

Exercise: Prove that $\lambda(ABA) \leq \lambda(A)\lambda(B)\lambda(A)$.

Now we come to the third product: Replacement product $\textcircled{r}$.

Suppose G is a D-regular graph on n vertices. Let H be a d-regular graph on D vertices. We will replace each vertex v of G by a copy H_v of H. Each vertex of H_v will be placed in 1-1 correspondence with each edge at v. Within each copy H_v the edges are as they exist in H. Between different copies of H_v, we connect them using the name of the vertex of H_v in the rotation map of G. We weight these edges by a multiplicity of d so that a random step of the the random walk in $G\textcircled{r}H$ has probability $1/2$ to stay within the copy H_v or to go to another copy H_u. Formally,

1. For every vertex $v \in G$, the graph $G\textcircled{r}H$ has a copy H_v of H (including both edges and vertices. Thus $V(G\textcircled{r}H) = [n] \times [D]$.
2. If $\mathrm{Rot}_G(v,i) = (u,j)$, where $u, v \in [n]$ and $i, j \in [D]$, then we place d parallel edges between (v,i) and (u,j).

The graph $G\textcircled{r}H$ has nD vertices and is $2d$-regular (d edges within H_v and d edges going to another copy H_u).

In terms of the rotation map for $G\textcircled{r}H$, for all $(v,i) \in V(G\textcircled{r}H) = [n] \times [D]$ and for all $(\sigma, b) \in [d] \times \{0,1\}$ (edge labels),

$$\mathrm{Rot}_{G\textcircled{r}H}((v,i),(\sigma,b)) = \begin{cases} ((v,j),(\tau,0)) & \text{if } b = 0, \text{ and } \mathrm{Rot}_H(i,\sigma) = (j,\tau) \\ ((u,j),(\sigma,1)) & \text{if } b = 1, \text{ and } \mathrm{Rot}_G(v,i) = (u,j). \end{cases}$$

Note that the first line (for $b = 0$) is for the move inside H_v. In this case, only Rot_H is involved, but not Rot_G. The second line (for $b = 1$) is for the move between copies of H. In this case, only Rot_G is involved, but not Rot_H, and there are d parallel edges corresponding to the multiplicity for all $\sigma \in [d]$.

We can define a permutation matrix $P \in \mathbb{R}^{nD \times nD}$, indexed by $\{(v,i) \in [n] \times [D]\}$. The column indexed by (v,i) has a single 1 at (u,j) if $\mathrm{Rot}_G(v,i) = (u,j)$, and 0 elsewhere. Note that by the definition of the rotation map, this matrix P is symmetric (thus an involution: $P = P^{\intercal} = P^{-1}$). Then the random walk matrix of $G\textcircled{r}H$ is

$$A\textcircled{r}B = \frac{1}{2}P + \frac{1}{2}(I_n \otimes B).$$

Note that $P(e_v^{(n)} \otimes e_i^{(D)}) = e_u^{(n)} \otimes e_j^{(D)}$, where $\mathrm{Rot}_G(v,i) = (u,j)$, and $e_v^{(n)}$ and $e_i^{(D)}$ denote the unit vectors in the respective dimensions.

Lemma 9.2. *If $\lambda(G) \leq 1 - \epsilon$, and $\lambda(H) \leq 1 - \delta$, then*

$$\lambda(G\textcircled{r}H) \leq 1 - \frac{\epsilon\delta^2}{24}.$$

Thus while the ordinary product improves the spectral gap (at the cost of increasing degrees), the replacement product reduces degrees with not too much of a loss of the spectral gap. If we imagine G is a D-regular graph on n vertices with a good expansion, but D is too large (perhaps because we obtained G by ordinary product), using the replacement product can reduce the degree and not reducing the expansion too badly. This process can then be repeated. (You should connect this idea with the idea of taking random walks on $\{0,1\}^n$ as witnesses for an RP language membership test.)

Proof. We prove that $\lambda((G\textcircled{r}H)^3) \leq 1-\frac{\epsilon\delta^2}{8}$. This is sufficient, because $(1-x)^3 = 1-3x+x^2(3-x) \geq 1-3x$, for $0 \leq x \leq 3$, (we take $x = \frac{\epsilon\delta^2}{8}$), and $\lambda((G\textcircled{r}H)^3) = [\lambda(G\textcircled{r}H)]^3$.

The random walk matrix of $(G\textcircled{r}H)^3$ is

$$W = \left(\frac{1}{2}P + \frac{1}{2}(I_n \otimes B)\right)^3.$$

Claim: We can write $B = (1-\delta)B' + \delta J_D$, where B' has 2-norm $||B'|| \leq 1$, and J_D is $\frac{1}{D}$ times the $D \times D$ all 1's matrix.

We will prove this Claim later. For now assume this Claim, then we can substitute B by this expression and get

$$W = \left(\frac{1}{2}P + \frac{1-\delta}{2}(I_n \otimes B') + \frac{\delta}{2}(I_n \otimes J_D)\right)^3.$$

We expand the product and get a horrible looking sum of 27 terms. Other than the term

$$\frac{\delta^2}{8}(I_n \otimes J_D) \cdot P \cdot (I_n \otimes J_D),$$

we get 26 terms and each product is a matrix with norm (2-norm) at most 1, with some weight, and the combined coefficient weight sums to $(1 - \frac{\delta^2}{8})$. Thus we have the expression

$$W = (1 - \frac{\delta^2}{8})W' + \frac{\delta^2}{8}(I_n \otimes J_D) \cdot P \cdot (I_n \otimes J_D).$$

(Note that we are not claiming that for matrices M and M' each of norm ≤ 1, we get to "take out the coefficient" $cM+c'M'$ directly, (for $c, c' > 0$). But, instead, we simply let $M'' = \frac{1}{c+c'}(cM+c'M')$, and then $cM + c'M' = (c+c')M''$, and $||M''|| \leq \frac{1}{c+c'}(c||M|| + c'||M||) \leq 1$.)

Now we verify that

- $(I_n \otimes J_D) \cdot P \cdot (I_n \otimes J_D) = A \otimes J_D$.

  Take $e_v = e_v^{(n)}$ and $f_i = e_i^{(D)}$. Consider a basis vector $e_v \otimes f_i$. First

$$(I_n \otimes J_D)(e_v \otimes f_i) = e_v \otimes \frac{1}{D}(1,1,\ldots,1)^\intercal = \frac{1}{D}\sum_{k=1}^{D} e_v \otimes f_k.$$

Then

$$P\left(\frac{1}{D}\sum_{k=1}^{D} e_v \otimes f_k\right) = \frac{1}{D}\sum_{1\leq k\leq D, \mathrm{Rot}_G(v,k)=(u,j)} e_u \otimes f_j.$$

Finally,

$$\begin{aligned}(I_n \otimes J_D)\left(\frac{1}{D}\sum_{1\leq k\leq D, \mathrm{Rot}_G(v,k)=(u,j)} e_u \otimes f_j\right) &= \frac{1}{D}\sum_{1\leq k\leq D, \mathrm{Rot}_G(v,k)=(u,j)} e_u \otimes (\frac{1}{D}(1,1,\ldots,1)^\intercal) \\ &= \left(\frac{1}{D}\sum_{1\leq k\leq D, \mathrm{Rot}_G(v,k)=(u,j)} e_u\right) \otimes (\frac{1}{D}(1,1,\ldots,1)^\intercal).\end{aligned}$$

This is exactly $(A \otimes J_D)(e_v \otimes f_i)$.

Comment: All this algebra perhaps obscures the underlying simple reasoning. What we did is to separate out the component of W into the part along $(A \otimes J_D)$ with a non-trivial weight $\frac{\delta^2}{8}$, from the rest, expressed as a "noise" with weight $1 - \frac{\delta^2}{8}$. The "noise" part is some norm 1 matrix. But the matrix $(A \otimes J_D)$ represents a random step along A (on G, which has a non-trivial diffusion with parameter ϵ), coupled with a "totally" random step represented by J_D on H. All this algebra is to spell out this reasoning.

As W and $A \otimes J_D$ are both symmetric, so is W' (you don't need to multiply out W to see this for W') and they both preserve the eigenspace $\mathbf{1} \in \mathbb{R}^{nD}$ and $\mathbf{1}^\perp$. Hence

$$\lambda(W) \leq (1 - \frac{\delta^2}{8})\lambda(W') + \frac{\delta^2}{8}\lambda(A \otimes J_D).$$

Since $\lambda(W') \leq ||W'|| \leq 1$, and $\lambda(J_D) = 0$, we have $\lambda(A \otimes J_D) = \lambda(A) \leq 1 - \epsilon$, we have

$$\lambda(W) \leq (1 - \frac{\delta^2}{8}) + \frac{\delta^2}{8}(1 - \epsilon) = 1 - \frac{\epsilon\delta^2}{8}.$$

Now we finish the proof by taking care of that Claim.

We prove that every random walk matrix $B \in \mathbb{R}^{D \times D}$ with a spectral gap $\delta = 1 - \lambda$ can be written as $\lambda M + (1 - \lambda)J_D$, for some M with $||M|| \leq 1$. First if B is linearly dependent with J_D, then by the row sum being 1, we have $B = J_D$, and we are done with $M = 0$. So suppose B and J_D are linearly independent. Then we claim $\lambda \neq 0$. Otherwise B has eigenvalues $1, 0, \ldots, 0$, with orthonormal eigenvectors $v_1 = \sqrt{D}\mathbf{1} = \frac{1}{\sqrt{D}}(1, 1, \ldots, 1)^\intercal$, and $v_2, \ldots, v_D$. Then by the spectral decomposition

$$B = (v_1, v_2, \ldots, v_D) \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 \end{pmatrix} \begin{pmatrix} v_1^\intercal \\ v_2^\intercal \\ \vdots \\ v_D^\intercal \end{pmatrix} = v_1 v_1^\intercal = J_D.$$

So we have $\lambda \neq 0$.

Now we let $M = \frac{1}{\lambda}(B - (1 - \lambda)J_D)$. We prove that $||M|| \leq 1$. Let x be any unit vector. Decompose x along $\mathbf{1}$ and $\mathbf{1}^\perp$, we write $x = y + z$, and $||x||^2 = ||y||^2 + ||z||^2$. $Bx = By + Bz = y + Bz$, an since $z \in \mathbf{1}^\perp$, $||Bz|| \leq \lambda||z||$. Then

$$Mx = \frac{y + Bz - (1 - \lambda)J_D z}{\lambda} = \frac{y + Bz - (1 - \lambda)y}{\lambda} = y + \frac{Bz}{\lambda},$$

because $J_D x = J_D y + J_D z = y + 0$. Hence

$$\langle x, Mx \rangle = \langle x, y \rangle + \frac{1}{\lambda}\langle x, Bz \rangle = ||y||^2 + \frac{1}{\lambda}\langle x, Bz \rangle,$$

and so $|\langle x, Mx \rangle| \leq ||y||^2 + ||z||^2 = ||x||^2 = 1$, we conclude that $||M|| \leq 1$. This proves the Claim. □

Exercise: Use this Claim to improve the the constant $\sqrt{\beta} + \lambda$ in Theorem 8.1 to $(1 - \lambda)\sqrt{\beta} + \lambda$, which is a wighted sum of $\sqrt{\beta}$ and 1, thus also < 1.

A Construction

Theorem 9.3. *One can explicitly construct a (n, d, λ)-expander family of d-regular graphs, for some constants d and $\lambda < 1$.*

Proof. The basic idea is simple: we will iterate

$$G' \leftarrow (G \otimes G)^{\text{const}} ⓡ H.$$

The tensor product is to increase the size rapidly, the powering to a constant is to shrink the λ (thus to increase the gap $1-\lambda$) but at the cost of increasing degrees, and ⓡ is to reduce the degree.

Now the details. Start with a $((2d)^c, 2d, 1/2)$-expander G_1, and an $((2d)^{2c}, 2d, 1/2)$-expander G_2 for some constants c, d. Such expanders can be found by an exhaustive search, and the existence is guaranteed by the probabilistic proof. We can take $c = 100$, say. Also we can find an $((2d)^c, 2d, 0.01)$-expander H, for a sufficiently large d, again by an exhaustive search. Let $D = (2d)^c$. Then G_1, G_2, and H have D, D^2, and D vertices, respectively.

For $k > 2$, define

$$G_k = \left(G_{\lfloor \frac{k-1}{2} \rfloor} \otimes G_{\lceil \frac{k-1}{2} \rceil}\right)^{50} ⓡ H.$$

Claim: G_k is a $(D^k, 2d, 49/50)$-expander.

This is an easy induction. The base cases $k = 1, 2$ are clear. If inductively G_ℓ has $n_\ell = D^\ell$ vertices for $\ell < k$, then $G_{\lfloor \frac{k-1}{2} \rfloor} \otimes G_{\lceil \frac{k-1}{2} \rceil}$ has D^{k-1} vertices, and so G_k has D^k vertices. If G is $2d$-regular, then $G \otimes G$ is $(2d)^2$-regular, and $(G \otimes G)^{50}$ is D-regular, and therefore $(G \otimes G)^{50} ⓡ H$ makes sense, and is back to $2d$-regular.

Finally, by Lemma 9.2,

$$\lambda(G_k) \le 1 - \frac{\frac{1}{2}(0.99)^2}{24} \le \frac{49}{50}.$$

□

10 Reingold's Theorem

Using these ideas Reingold proved that Connectivity problem for undirected graphs can be decided in deterministic logspace. Thus, not only a random walk (which takes randomized logspace) can solve this problem, it can also be done deterministically.

Theorem 10.1. *Connectivity for undirected graph is in deterministic logspace.*

Suppose we are given a pair of vertices s and t in an undirected graph G. The task is to decide whether there is a path (of length $\le n-1$) rom s to t. (This is called the s-t connectivity problem. One can cycle through all t in deterministic logspace to decide connectivity for the whole graph. Thus the s-t connectivity problem is logspace equivalent to the connectivity problem.)

Suppose G is an expander, then for any pair of connected vertices s and t, there is a path of length $O(\log n)$ that connects them. This is because, if we consider the set $B_\ell(s)$ of reachable vertices from s by paths of length $\le \ell$, then as along is $|B_\ell(s)| \le n/2$, the size keeps expanding by a constant factor $1 + \epsilon$. Thus within $O(\log n)$ steps we reach $|B_\ell(s)| > n/2$. Similarly the set $|B_\ell(t)|$ also reach $n/2$ for ℓ within $O(\log n)$, and so they have a nonempty intersection. Thus there is a path of the $O(\log n)$ connecting s and t.

The main idea of Reingold's proof is to replace G by a sequence of operations so that we end up with a new graph G' what is only polynomially larger, but every connected component of G has become an expander.

For preprocessing, we can replace each vertex v of G by a cycle of length $\deg(v)$. We add self-loops at every vertex so that it is d^{50}-regular, where d is a constant to be chosen shortly. The connected components of G remain connected components.

For every connected component of G there is already a spectral gap of $1 - 1/\text{poly}(n)$. Let $G_0 = G$. For $k \geq 1$, let $G_k = (G_{k-1} \textcircled{r} H)^{50}$, where H is a $(d^{50}, d/2, 0.01)$-expander. Note that if G_{k-1} is d^{50}-regular, then $G_{k-1} \textcircled{r} H$ is d-regular again, and so G_k is d^{50}-regular.

Claim: For $k = O(\log n)$, every connected component of G_k is an $(d^{50k}n, d^{50}, \lambda)$-expander, where $\lambda \leq 19/20$.

By Lemma 9.2, if $\lambda(G_{k-1}) \leq 1 - \epsilon$, then $\lambda(G_{k-1} \textcircled{r} H) \leq 1 - \frac{(0.99)^2\epsilon}{24} < 1 - \frac{\epsilon}{25}$. The initial ϵ for G_0 is $\Theta(1/n^2)$. Hence

$$\lambda(G_k) \leq \left(1 - \frac{\epsilon}{25}\right)^{50} < e^{-2\epsilon} \approx 1 - 2\epsilon.^{\ddagger}$$

Thus if we take $k = O(\log n)$, then G_k has size $n^{O(1)}$, and $\lambda(G_k) \leq 19/20$.

So the s-t-connectivity algorithm is to run recursively using a stack of $O(\log n)$ levels, a walk of $O(\log n)$ steps. An logspace algorithm does not have space to write down all the intermediate graphs. Instead it recursively constructs the necessary neighborhood edge relationships "on the fly" as needed, all in logspace.

11 Coupling and Path Coupling

I will use the excellent notes by Mark Jerrum, found at

http://www.maths.qmul.ac.uk/~mj/ETHbook/chapter4.pdf

12 Interactive Proofs

If time permit, I will further discuss:

1. Interactive Proof System for graph non-isomorphism.
2. Permanent has Interactive Proof Systems.
3. IP = PSPACE by commutative diagrams.

13 Graph Isomorphism

Permutation groups and its application to graph isomorphism.

$^{\ddagger}(1 - 1/x)^x < e^{-1}$, when $x \to \infty$, as $x \log_e(1 - 1/x) = -\left[1 + \frac{1}{2x} + \frac{1}{3x^2} + \ldots\right] < -1$.