Chapter 16

Random and Pseudo-Random Walks on Graphs

Random walks on graphs have turned out to be a powerful tool in the design of algorithms and other applications. In particular, *expander graphs*, which are graphs on which random walks have particularly good properties, are extremely useful in complexity and other areas of computer science. In this chapter we study random walks on general regular graphs, leading to a the randomized logspace algorithm for undirected connectivity alluded to in Chapter 7. We then show the definition and constructions of expander graphs, and their application for randomness-efficient error reduction of probabilistic algorithms. Finally we use the ideas behind that construction to show a *deterministic* logspace algorithm for undirected connectivity.

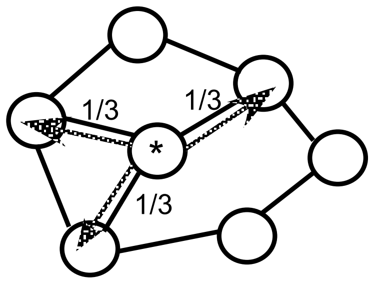


Figure 16.1: A *random walk* on a graph can be thought of as placing a token on some vertex, and at each step move the token to random neighbor of its current location.

The idea of a random walk is simple (see Figure 16.1): let G be a graph (in this chapter we restrict ourselves to *undirected* graphs) and let v be a vertex in G. A *T-step random walk from v in G* is the sequence of dependent random variables $X_0, \ldots, X_T$ defined as follows: $X_0 = v$ with probability one, and for $i \in [T]$, X_i is chosen at random from $\Gamma(X_{i-1})$, where for any vertex u, $\Gamma(u)$ denotes the set of neighbors of u in the graph G. That is, a random walk involves starting at a vertex and then at each step going to a random neighbor of this vertex. A probabilistic process of this form, where there is a fixed distribution specifying the dependence of X_i on X_{i-1}, is often called a *Markov chain*.

16.1 Undirected connectivity in randomized logspace

Recall the language PATH of the triplets $\langle G, s, t\rangle$ where G is a (directed) graph and s and t are vertices in G with a path from s to t. In Chapter 3 we showed that PATH is **NL**-complete. Consider the problem UPATH where G is restricted to be *undirected* (or equivalently, we place the condition that there is an edge from i to j in G iff there's an edge from j to i: the adjacency matrix is symmetric). It turns out that UPATH can be solved by a logspace probabilistic TM.

Theorem 16.1 ([**?**])
UPATH $\in$ **RL**.

Theorem 16.1 is proven by a simple algorithm. Given an input G, s, t to UPATH, we first use an implicitly logspace computable reduction to transform the graph into a degree 4-regular graph G' such that s is connected to t in G' if and only if they are connected in G (see Claim 16.1.1).[1] We then take a random walk of length $T = 100n^3 \log n$ from s on the graph G' (where n is the number of vertices in G'). We accept if at the end of the walk we reach the vertex t. Otherwise, reject. The algorithm can be implemented in $O(\log n)$ space since it only requires space to store the current and next vertex in the walk, and a counter. Clearly, it never accepts if t is not connected to s. We show in the next section that if t is connected to s, then the algorithm accepts with probability $\frac{1}{\Omega(n)}$, which can be easily amplified using the standard error reduction techniques (see Section 7.4.1).

[1]It can be shown, via a slightly different analysis, that the algorithm will work even without this step, see Exercise 9 of Chapter 9.

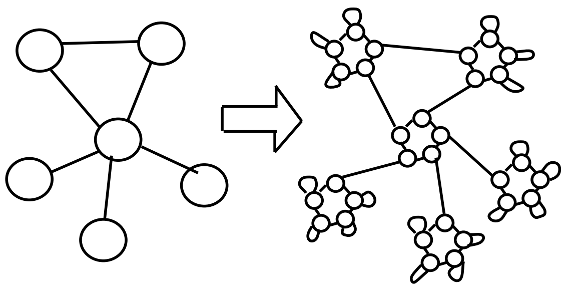


Figure 16.2: To reduce a graph G to a degree 3 regular graph that has the same connectivity properties, we replace each vertex with a cycle, making every vertex have degree 2 or 3. We then add self loops on some vertices to make the graph 3 regular. To ensure every vertex has a self loop, we add such a loop for every vertex, making the graph 4-regular.

Reducing to the regular constant-degree case. As mentioned above, we start by reducing to the case that every vertex in G has degree 4 (i.e., G is 4-regular) and that every vertex has a self-loop (i.e., an edge to itself). This claim not strictly necessary for this algorithm but will somewhat simplify the analysis and be useful for us later on. We note that here and throughout this chapter all graphs may have parallel edges (i.e., more than one edge between the same pair of vertices i, j).

Claim 16.1.1
There exists an implicitly logspace computable function f that maps every triple $\langle G, s, t\rangle$ to a triple $\langle G', s', t'\rangle$ such that:

1. *G' is a 4-regular graph with a self loop at each vertex.*
2. *s is connected to t in G iff s' is connected to t' in G'.*

Proof sketch: We sketch the proof, leaving verifying the details as an exercise. We transform G to G' as shown in Figure 16.2: for every vertex i in G, the graph G' will have n vertices arranged in a cycle. For every two neighbors i, j in G, we connect in an edge the j^{th} vertex from the cycle corresponding to i, and the i^{th} vertex from the cycle corresponding to j. Thus, every vertex in G' has either degree two (if it's only connected to its neighbors on the cycle) or three (if it also has a neighbor in a different cycle). We add to each vertex either one or two self loops to make the degree four. It can be easily seen that determining the value of an entry in the adjacency matrix of G' can be computed in log space using read-only access to the adjacency matrix of G. ■

16.2 Random walk on graphs

In this section we study random walks on (undirected regular) graphs. As a corollary we obtain the proof of correctness for the above algorithm for UPATH. We will see that we can use elementary linear algebra to relate parameters of the graph's adjacency matrix to the behavior of the random walk on that graph. The following definitions and notations will be widely used in this and later sections of this chapter:

16.2.1 Distributions as vectors and the parameter $\lambda(G)$.

Let G be a d-regular n-vertex graph. Let $\mathbf{p}$ be some probability distribution over the vertices of G. We can think of $\mathbf{p}$ as a (column) vector in $\mathbb{R}^n$ where $\mathbf{p}_i$ is the probability that vertex i is obtained by the distribution. Note that the L_1-norm of $\mathbf{p}$ (see Note 16.2), defined as $|\mathbf{p}|_1 = \sum_{i=1}^n |\mathbf{p}_i|$, is equal to 1. (In this case the absolute value is redundant since $\mathbf{p}_i$ is always between 0 and 1.)

Now let $\mathbf{q}$ represent the distribution of the following random variable: choose a vertex i in G according to $\mathbf{p}$, then take a random neighbor of i in G. We can compute $\mathbf{q}$ as a function of $\mathbf{p}$: the probability $\mathbf{q}_j$ that j is chosen is equal to the sum over all j's neighbors i of the probability $\mathbf{p}_i$ that i is chosen times $1/d$ (where $1/d$ is the probability that, conditioned on i being chosen, the walk moves to $\mathbf{q}$). Thus $\mathbf{q} = A\mathbf{p}$, where $A = A(G)$ which is the *normalized adjacency matrix* of G. That is, for every two vertices i, j, $A_{i,j}$ is equal to the number of edges between i and j divided by d. Note that A is a symmetric matrix,[2] where each entry is between 0 and 1, and the sum of entries in each row and column is exactly one (such a matrix is called a symmetric *stochastic* matrix).

Let $\{\mathbf{e}^i\}_{i=1}^n$ be the *standard basis* of $\mathbb{R}^n$ (i.e. $\mathbf{e}^i$ has 1 in the i^{th} coordinate and zero everywhere else). Then, $A^T\mathbf{e}^s$ represents the distribution X_T of taking a T-step random walk from the vertex s. This already suggests that considering the adjacency matrix of a graph G could be very useful in analyzing random walks on G.

[2] A matrix A is *symmetric* if $A = A^\dagger$, where $A^\dagger$ denotes the *transpose* of A. That is, $(A^\dagger)_{i,j} = A_{j,i}$ for every i, j.

NOTE 16.2 (L_p NORMS)
A *norm* is a function mapping a vector $\mathbf{v}$ into a real number $\|\mathbf{v}\|$ satisfying **(1)** $\|\mathbf{v}\| \geq 0$ with $\|\mathbf{v}\| = 0$ if and only $\mathbf{v}$ is the all zero vector, **(2)** $\|\alpha\mathbf{v}\| = |\alpha| \cdot \|\mathbf{v}\|$ for every $\alpha \in \mathbb{R}$, and **(3)** $\|\mathbf{v}+\mathbf{u}\| \leq \|\mathbf{v}\| + \|\mathbf{u}\|$ for every vector $\mathbf{u}$. The third inequality implies that for every norm, if we define the *distance* between two vectors $\mathbf{u}$,$\mathbf{v}$ as $\|\mathbf{u}-\mathbf{v}\|$ then this notion of distance satisfies the triangle inequality.

For every $\mathbf{v} \in \mathbb{R}^n$ and number $p \geq 1$, the L_p *norm* of $\mathbf{v}$, denoted $\|\mathbf{v}\|_p$, is equal to $(\sum_{i=1}^n |\mathbf{v}_i|^p)^{1/p}$. One particularly interesting case is $p = 2$, the so-called *Euclidean norm*, in which $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^n \mathbf{v}_i^2} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}$. Another interesting case is $p = 1$, where we use the single bar notation and denote $|\mathbf{v}|_1 = \sum_{i=1}^n |\mathbf{v}_i|$. Another case is $p = \infty$, where we denote $\|\mathbf{v}\|_\infty = \lim_{p\to\infty} \|\mathbf{v}\|_p = \max_{i\in[n]} |\mathbf{v}_i|$.

The *Hölder inequality* says that for every p, q with $\frac{1}{p} + \frac{1}{q} = 1$, $\|\mathbf{u}\|_p \|\mathbf{v}\|_q \geq \sum_{i=1}^n |\mathbf{u}_i\mathbf{v}_i|$. To prove it, note that by simple scaling, it suffices to consider norm one vectors, and so it enough to show that if $\|\mathbf{u}\|_p = \|\mathbf{v}\|_q = 1$ then $\sum_{i=1}^n |\mathbf{u}_i||\mathbf{v}_i| \leq 1$. But $\sum_{i=1}^n |\mathbf{u}_i||\mathbf{v}_i| = \sum_{i=1}^n |\mathbf{u}_i|^{p(1/p)}|\mathbf{v}_i|^{q(1/q)} \leq \sum_{i=1}^n \frac{1}{p}|\mathbf{u}_i|^p + \frac{1}{q}|\mathbf{v}_i|^q = \frac{1}{p} + \frac{1}{q} = 1$, where the last inequality uses the fact that for every $a, b > 0$ and $\alpha \in [0,1]$, $a^\alpha b^{1-\alpha} \leq \alpha a + (1-\alpha)b$. This fact is due to the log function being concave— having negative second derivative, implying that $\alpha \log a + (1-\alpha)\log b \leq \log(\alpha a + (1-\alpha)b)$.
Setting $p = 1$ and $q = \infty$, the Hölder inequality implies that

$$\|\mathbf{v}\|_2 \leq |\mathbf{v}|_1 \|\mathbf{v}\|_\infty$$

Setting $p = q = 2$, the Hölder inequality becomes the *Cauchy-Schwartz Inequality* stating that $\sum_{i=1}^n |\mathbf{u}_i\mathbf{v}_i| \leq \|\mathbf{u}\|_2\|\mathbf{v}\|_2$. Setting $\mathbf{u} = (1/\sqrt{n}, 1/\sqrt{n}, \ldots, 1/\sqrt{n})$, we get that

$$|\mathbf{v}|_1/\sqrt{n} = \sum_{i=1}^n \tfrac{1}{\sqrt{n}}|\mathbf{v}_i| \leq \|\mathbf{v}\|_2$$

Definition 16.3 (The parameter $\lambda(G)$.)
Denote by $\mathbf{1}$ the vector $(1/n, 1/n, \ldots, 1/n)$ corresponding to the uniform distribution. Denote by $\mathbf{1}^\perp$ the set of vectors perpendicular to $\mathbf{1}$ (i.e., $\mathbf{v} \in \mathbf{1}^\perp$ if $\langle \mathbf{v}, \mathbf{1} \rangle = (1/n)\sum_i \mathbf{v}_i = 0$).
The parameter $\lambda(A)$, denoted also as $\lambda(G)$, is the maximum value of $\|A\mathbf{v}\|_2$ over all vectors $\mathbf{v} \in \mathbf{1}^\perp$ with $\|\mathbf{v}\|_2 = 1$.

Remark 16.4
The value $\lambda(G)$ is often called the *second largest eigenvalue* of G. The reason is that since A is a symmetric matrix, we can find an orthogonal basis of eigenvectors $\mathbf{v}^1, \ldots, \mathbf{v}^n$ with corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$ which we can sort to ensure $|\lambda_1| \geq |\lambda_2| \ldots \geq |\lambda_n|$. Note that $A\mathbf{1} = \mathbf{1}$. Indeed, for every i, $(A\mathbf{1})_i$ is equal to the inner product of the i^{th} row of A and the vector $\mathbf{1}$ which (since the sum of entries in the row is one) is equal to $1/n$. Thus, $\mathbf{1}$ is an *eigenvector* of A with the corresponding eigenvalue equal to 1. One can show that a symmetric stochastic matrix has all eigenvalues with absolute value at most 1 (see Exercise 1) and hence we can assume $\lambda_1 = 1$ and $\mathbf{v}^1 = \mathbf{1}$. Also, because $\mathbf{1}^\perp = \mathrm{Span}\{\mathbf{v}^2, \ldots, \mathbf{v}^n\}$, the value λ above will be maximized by (the normalized version of) $\mathbf{v}^2$, and hence $\lambda(G) = |\lambda_2|$. The quantity $1 - \lambda(G)$ is called the *spectral gap* of the graph. We note that some texts use *un-normalized* adjacency matrices, in which case $\lambda(G)$ is a number between 0 and d and the spectral gap is defined to be $d - \lambda(G)$.

One reason that $\lambda(G)$ is an important parameter is the following lemma:

Lemma 16.5
For every regular n vertex graph $G = (V, E)$ let $\mathbf{p}$ be any probability distribution over V, then

$$\|A^T\mathbf{p} - \mathbf{1}\|_2 \leq \lambda^T$$

Proof: By the definition of $\lambda(G)$, $\|A\mathbf{v}\|_2 \leq \lambda\|\mathbf{v}\|_2$ for every $\mathbf{v} \perp \mathbf{1}$. Note that if $\mathbf{v} \perp \mathbf{1}$ then $A\mathbf{v} \perp \mathbf{1}$ since $\langle \mathbf{1}, A\mathbf{v} \rangle = \langle A^\dagger \mathbf{1}, \mathbf{v} \rangle = \langle \mathbf{1}, \mathbf{v} \rangle = 0$ (as $A = A^\dagger$ and $A\mathbf{1} = \mathbf{1}$). Thus A maps the space $\mathbf{1}^\perp$ to itself and since it shrinks any member of this space by at least λ, $\lambda(A^T) \leq \lambda(A)^T$. (In fact, using the eigenvalue definition of λ, it can be shown that $\lambda(A^T) = \lambda(A)$.)

Let $\mathbf{p}$ be some vector. We can break $\mathbf{p}$ into its components in the spaces parallel and orthogonal to $\mathbf{1}$ and express it as $\mathbf{p} = \alpha\mathbf{1} + \mathbf{p}'$ where $\mathbf{p}' \perp \mathbf{1}$

and α is some number. If $\mathbf{p}$ is a probability distribution then $\alpha = 1$ since the sum of coordinates in $\mathbf{p}'$ is zero. Therefore,

$$A^T\mathbf{p} = A^T(\mathbf{1} + \mathbf{p}') = \mathbf{1} + A^T\mathbf{p}'$$

Since $\mathbf{1}$ and $\mathbf{p}'$ are orthogonal, $\|\mathbf{p}\|_2^2 = \|\mathbf{1}\|_2^2 + \|\mathbf{p}'\|_2^2$ and in particular $\|\mathbf{p}'\|_2 \leq \|\mathbf{p}\|_2$. Since $\mathbf{p}$ is a probability vector, $\|\mathbf{p}\|_2 \leq |\mathbf{p}|_1 \cdot 1 \leq 1$ (see Note 16.2). Hence $\|\mathbf{p}'\|_2 \leq 1$ and

$$\|A^T\mathbf{p} - \mathbf{1}\|_2 = \|A^T\mathbf{p}'\|_2 \leq \lambda^T$$

■

It turns out that every connected graph has a noticeable spectral gap:

Lemma 16.6
For every d-regular connected G with self-loops at each vertex, $\lambda(G) \leq 1 - \frac{1}{8dn^3}$.

Proof: Let $\mathbf{u} \perp \mathbf{1}$ be a unit vector and let $\mathbf{v} = A\mathbf{u}$. We'll show that $1 - \|\mathbf{v}\|_2^2 \geq \frac{1}{d4n^3}$ which implies $\|\mathbf{v}\|_2^2 \leq 1 - \frac{1}{d4n^3}$ and hence $\|\mathbf{v}\|_2 \leq 1 - \frac{1}{d8n^3}$.

Since $\|\mathbf{u}\|_2 = 1$, $1 - \|\mathbf{v}\|_2^2 = \|\mathbf{u}\|_2^2 - \|\mathbf{v}\|_2^2$. We claim that this is equal to $\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2$ where i, j range from 1 to n. Indeed,

$$\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 = \sum_{i,j} A_{i,j}\mathbf{u}_i^2 - 2\sum_{i,j} A_{i,j}\mathbf{u}_i\mathbf{v}_j + \sum_{i,j} A_{i,j}\mathbf{v}_j^2 =$$
$$\|\mathbf{u}\|_2^2 - 2\langle A\mathbf{u}, \mathbf{v}\rangle + \|\mathbf{v}\|_2^2 = \|\mathbf{u}\|_2^2 - 2\|\mathbf{v}\|_2^2 + \|\mathbf{v}\|_2^2,$$

where these equalities are due to the sum of each row and column in A equalling one, and because $\|\mathbf{v}\|_2^2 = \langle \mathbf{v}, \mathbf{v}\rangle = \langle A\mathbf{u}, \mathbf{v}\rangle = \sum_{i,j} A_{i,j}\mathbf{u}_i\mathbf{v}_j$.

Thus it suffices to show $\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \frac{1}{d4n^3}$. This is a sum of non-negative terms so it suffices to show that for *some* i, j, $A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \frac{1}{d4n^3}$. First, because we have all the self-loops, $A_{i,i} \geq 1/d$ for all i, and so we can assume $|\mathbf{u}_i - \mathbf{v}_i| < \frac{1}{2n^{1.5}}$ for every $i \in [n]$, as otherwise we'd be done.

Now sort the coordinates of $\mathbf{u}$ from the largest to the smallest, ensuring that $\mathbf{u}_1 \geq \mathbf{u}_2 \geq \cdots \mathbf{u}_n$. Since $\sum_i \mathbf{u}_i = 0$ it must hold that $\mathbf{u}_1 \geq 0 \geq \mathbf{u}_n$. In fact, since $\mathbf{u}$ is a unit vector, either $\mathbf{u}_1 \geq 1/\sqrt{n}$ or $\mathbf{u}_n \leq 1/\sqrt{n}$ and so $\mathbf{u}_1 - \mathbf{u}_n \geq 1/\sqrt{n}$. One of the $n - 1$ differences between consecutive coordinates $\mathbf{u}_i - \mathbf{u}_{i+1}$ must be at least $1/n^{1.5}$ and so there must be an i_0 such that if we let $S = \{1, \ldots, i_0\}$ and $\overline{S} = [n] \setminus S_i$, then for every $i \in S$ and $j \in \overline{S}$, $\mathbf{u}_i - \mathbf{u}_j \geq 1/n^{1.5}$. Since G is connected there exists an edge (i, j) between S and $\overline{S}$. Since $|\mathbf{v}_j - \mathbf{u}_j| \leq \frac{1}{2n^{1.5}}$, for this choice of i, j, $|\mathbf{u}_i - \mathbf{v}_j| \geq |\mathbf{u}_i - \mathbf{u}_j| - \frac{1}{2n^{1.5}} \geq \frac{1}{2n^{1.5}}$. Thus $A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \frac{1}{d}\frac{1}{4n^3}$. ■

Remark 16.7
The proof can be strengthened to show a similar result for every connected non-bipartite graph (not just those with self-loops at every vertex). Note that this condition is essential: if A is the adjacency matrix of a bipartite graph then one can find a vector $\mathbf{v}$ such that $A\mathbf{v} = -\mathbf{v}$.

16.2.2 Analysis of the randomized algorithm for undirected connectivity.

Together, Lemmas 16.5 and 16.6 imply that our algorithm for UPATH outputs "accept" with probability $1/\Omega(n)$ if s is connected to t in the graph:

Corollary 16.8
Let G be a d-regular n-vertex graph with all vertices having a self-loop. Let s be a vertex in G. Let $T > 10dn^3 \log n$ and let X_T denote the distribution of the vertex of the T^{th} step in a random walk from s. Then, for every j connected to s, $\Pr[X_T = j] > \frac{1}{2n}$.

Proof: By these Lemmas, if we consider the restriction of an n-vertex graph G to the connected component of s, then for every probability vector $\mathbf{p}$ over this component and $T \geq 10dn^3 \log n$, $\|A^T\mathbf{p} - \mathbf{1}\|_2 < \frac{1}{2n^{1.5}}$ (where $\mathbf{1}$ here is the uniform distribution over this component). Using the relations between the L_1 and L_2 norms (see Note 16.2), $|A^T\mathbf{p} - \mathbf{1}|_1 < \frac{1}{2n}$ and hence every element in the connected component appears in $A^T\mathbf{p}$ with at least $1/n - 1/(2n) \geq 1/(2n)$ probability. ■

16.3 Expander graphs and some applications.

Expander graphs have played a crucial role in numerous computer science applications, including routing networks, error correcting codes, hardness of approximation and the PCP theorem, derandomization, and more. In this chapter we will see their definition, constructions, and two applications, including a *deterministic logspace* algorithm for the problem UPATH of undirected connectivity.

Expanders can be defined in several roughly equivalent ways. One is that these are graphs where every set of vertices has a very large boundary. That is, for every subset S of vertices, the number of S's neighbors outside S is (up to a constant factor) roughly equal to the number of vertices inside S. (Of course this condition cannot hold if S is too big and already contains most of the vertices in the graph.) For example, the n by n grid (where a vertex is a pair (i, j) and is connected to the four neighbors $(i \pm 1, j \pm 1))$ is *not* an expander, as any k by k square (which is a set of size k^2) in this graph only has a boundary of size $O(k)$ (see Figure 16.3). Another way to define expanders is as graphs where the random walks rapidly converges to the uniform distribution. That is, unlike in the general case that (in regular graphs) the random walk may take a polynomial number of steps to converge to the uniform distribution, in an n-vertex regular expander this will only take $O(\log n)$ steps.

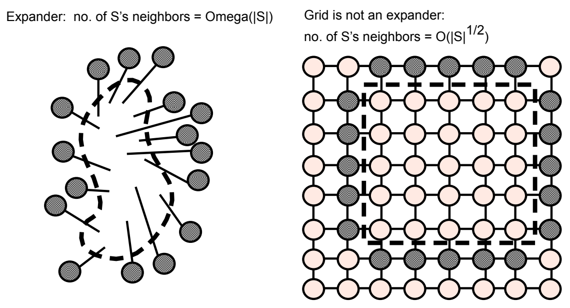

Expander: no. of S's neighbors = Omega(|S|)
Grid is not an expander:
no. of S's neighbors = O(|S|^{1/2})

Figure 16.3: In a *combinatorial expander*, every subset S of the vertices that is not too big has at least $\Omega(|S|)$ neighbors outside the set. The grid (and every other planar graph) is not a combinatorial expander as a $k \times k$ square in the grid has only $O(k)$ neighbors outside it.

Given the previous section, it is not surprising that we can define expanders also in an *algebraic* way, based on the parameter $\lambda(G)$ of Definition 16.9. That is, we will say that G is an expander if $\lambda(G)$ is bounded away from 1. By Lemma 16.5, this does indeed imply that the random walk on G converges to the uniform distribution (in the sense that regardless of the starting distribution, every vertex will be obtained with probability between $\frac{1}{2n}$ and $\frac{3}{2n}$) within $O(\log n)$ steps. We will also see later (Theorem 16.18) the relation between the parameter $\lambda(G)$ and the combinatorial definition of set expansion mentioned above.

Definition 16.9 ((n, d, λ)-graphs.)
If G is an n-vertex d-regular G with $\lambda(G) \leq \lambda$ for some number $\lambda < 1$ then we say that G is an (n, d, λ)-graph.

Expander families. For every d and $\lambda < 1$ a (d, λ)-*expander graph family* is a sequence $\{G_n\}_{n \in I}$ of graphs such that G_n is an (n, d, λ)-graph and I is an infinite set. We sometimes drop the qualifiers d, λ and simply call such a family an *expander graph family*, referring to a particular graph in the sequence as an *expander graph*.

Explicit constructions. We say that an expander family $\{G_n\}_{n \in I}$ is *explicit* if **(1)** the set I is polynomially dense in the sense that there exists a polynomial time algorithm A and a polynomial p such that for every $m \in \mathbb{N}$, $A(m)$ outputs a number $n \in I$ such that $m \leq n \leq p(m)$ and **(2)** there is a polynomial-time algorithm that on input 1^n with $n \in I$ outputs the adjacency matrix of G_n. We say that the family is *strongly explicit* if **(1)** I is polynomially dense and **(2)** there is a polynomial-time algorithm that for every $n \in I$ on inputs $\langle n, v, i \rangle$ where $1 \leq v \leq n'$ and $1 \leq i \leq d$ outputs the i^{th} neighbor of v. (Note that the algorithm runs in time polynomial in the its input length which is polylogarithmic in n.)

As we will see below it is not hard to show that expander families exist using the probabilistic method. But this does not yield *explicit* (or very explicit) constructions of such graphs. In fact, there are also several explicit and strongly explicit constructions of expander graphs known. The smallest λ can be for a d-regular n-vertex graph is $\Omega(\frac{1}{\sqrt{d}})$ and there are constructions meeting this bound (specifically the bound is $(1 - o(1))\frac{2\sqrt{d-1}}{d}$ where by $o(1)$ we mean a function that tends to 0 as the number of vertices grows; graphs meeting this bound are called *Ramanujan graphs*). However, for most applications in Computer Science, any family with constant d and $\lambda < 1$ will suffice (see also Remark 16.10 below). Some of these constructions are very simple and efficient, but their analysis is highly non-trivial and uses relatively deep mathematics.[3] We show in Section 16.5 a strongly explicit construction of expanders with elementary analysis. This construction also introduces a tool that is useful to derandomize the algorithm for UPATH.

REMARK 16.10
One reason that the particular constants of an expander family are not extremely crucial is that we can improve the constant λ (make it arbitrarily smaller) at the expense of increasing the degree: this follows from the fact, observed above in the proof of Lemma 16.5, that $\lambda(G^T) = \lambda(G)^T$, where G^T denotes the graph obtained by taking the adjacency matrix to the T^{th}

[3] An example for such an expander is the following 3-regular graph: the vertices are the numbers 1 to $p - 1$ for some prime p, and each number x is connected to $x + 1$,$x - 1$ and $x^{-1} \pmod{p}$.

Note 16.11 (Explicit construction of pseudorandom objects)
Expanders are one instance of a recurring theme in complexity theory (and other areas of math and computer science): it is often the case that a random object can be easily proven to satisfy some nice property, but the applications require an *explicit* object satisfying this property. In our case, a random d-regular graph is an expander, but to use it for, say, reducing the error of probabilistic algorithms, we need an *explicit* construction of an expander family, with an efficient deterministic algorithm to compute the neighborhood relations. Such explicit constructions can be sometimes hard to come by, but are often surprisingly useful. For example, in our case the explicit construction of expander graphs turns out to yield a deterministic logspace algorithm for undirected connectivity.
We will see another instance of this theme in Chapter 18, which discusses *error correcting codes*.

power, or equivalently, having an edge for every length-T path in G. Thus, we can transform an (n, d, λ) graph into an (n, d^T, λ^T)-graph for every $T \geq 1$. Later we will see a different transformation called the *replacement product* to decrease the degree at the expense of increasing λ somewhat (and also increasing the number of vertices).

16.3.1 Using expanders to reduce error in probabilistic algorithms

Before constructing expanders, let us see one application for them in the area of probabilistic algorithms. Recall that in Section 7.4.1 we saw that we can reduce the error of a probabilistic algorithm from, say, $1/3$ to $2^{-\Omega(k)}$ by executing it k times independently and taking the majority value. If the algorithm utilized m random coins, this procedure will use $m \cdot k$ random coins, and intuitively it seems hard to think of a way to save on randomness. Nonetheless, we will show that using expanders we can obtain such error reduction using only $m + O(k)$ random coins. The idea is simple: take an expander graph G from a very explicit family that is an $(N = 2^m, d, 1/10)$-graph for some constant d.[4] Choose a vertex v_1 at random, and take a

[4]In our definition of an expander family, we did not require that there is an N-vertex graph in the family for every N, however typical constructions can be tweaked to ensure

length $k-1$ long random walk on G to obtain vertices $v_2, \ldots, v_k$ (note that choosing a random neighbor of a vertex requires $O(\log d) = O(1)$ random bits). Invoke the algorithm k times using $v_1, \ldots, v_k$ as random coins (we identify the set $[N]$ of vertices with the set $\{0,1\}^m$ of possible random coins for the algorithm) and output the majority answer.

The analysis is also not too difficult, but to make it even simpler, we analyze here only the case of algorithms with one-sided error. For example, consider an **RP** algorithm that will never output "accept" if the input is not in the language, and for inputs in the language will output "accept" with probability 2/3 (the case of a **coRP** algorithm is analogous). For such an algorithm the procedure will output "accept" if the algorithm accepts even on a single set of coins v_i. If the input is not in the language, the procedure will never accept. If the input is in the language, then let $B \subseteq [N]$ denote the "bad" set of coins on which the algorithms rejects. We know that $|B| \leq \frac{N}{3}$. To show the procedure outputs "reject" with at most $2^{-\Omega(k)}$ probability, we prove the following theorem:

Theorem 16.12 (Expander walks)
Let G be an (N, d, λ) graph, and let $B \subseteq [N]$ be a set with $|B| \leq \beta N$. Let $X_1, \ldots, X_k$ be random variables denoting a $k-1$-step random walk from X_1, where X_1 is chosen uniformly in $[N]$. Then,

$$\underbrace{\Pr[\forall_{1 \leq i \leq k} X_i \in B]}_{(*)} \leq ((1-\lambda)\sqrt{\beta} + \lambda)^{k-1}$$

Note that if λ and β are both constants smaller than 1 then so is the expression $(1-\lambda)\sqrt{\beta} + \lambda$.

Proof: For $1 \leq i \leq k$, let B_i be the event that $X_i \in B$. Note that the probability $(*)$ we're trying to bound is $\Pr[B_1]\Pr[B_2|B_1]\cdots\Pr[B_k|B_1, \ldots, B_{k-1}]$. Let $\mathbf{p}^i \in \mathbb{R}^N$ be the vector representing the distribution of X_i, conditioned on the events $B_1, \ldots, B_i$. Denote by $\hat{B}$ the following linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$: for every $\mathbf{u} \in \mathbb{R}^N$, and $j \in [N]$, $(\hat{B}\mathbf{u})_j = \mathbf{u}_j$ if $j \in B$ and $(\hat{B}\mathbf{u})_j = 0$ otherwise. It's not hard to verify that $\mathbf{p}^1 = \frac{1}{\Pr[B_1]}\hat{B}\mathbf{1}$ (recall that $\mathbf{1} = (1/N, \ldots, 1/N)$ is the vector representing the uniform distribution over $[N]$). Similarly, $\mathbf{p}^2 = \frac{1}{\Pr[B_2|B_1]}\frac{1}{\Pr[B_1]}\hat{B}A\hat{B}\mathbf{1}$ where $A = A(G)$ is the adjacency

this (see Theorem 16.24 below) and so we ignore this issue and assume we have such a 2^m-vertex graph in the family. Note that we can use powering improve the parameter λ of the family to be smaller than 1/10 (see Remark 16.10).

matrix of G. Since every probability vector $\mathbf{p}$ satisfies $|\mathbf{p}|_1 = 1$,

$$(*) = |(\hat{B}A)^{k-1}\hat{B}\mathbf{1}|_1$$

We bound this norm by showing that

$$\|(\hat{B}A)^{k-1}\hat{B}\mathbf{1}\|_2 \leq \frac{((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}}{\sqrt{N}} \tag{1}$$

which suffices since for every $\mathbf{v} \in \mathbb{R}^N$, $|\mathbf{v}|_1 \leq \sqrt{N}\|\mathbf{v}\|_2$ (see Note 16.2).

To prove (1), we use the following definition and lemma:

Definition 16.13 (Matrix Norm)
If A is an m by n matrix, then $\|A\|$ is the maximum number α such that $\|A\mathbf{v}\|_2 \leq \alpha\|\mathbf{v}\|_2$ for every $\mathbf{v} \in \mathbb{R}^n$.

Note that if A is a normalized adjacency matrix then $\|A\| = 1$ (as $A\mathbf{1} = \mathbf{1}$ and $\|A\mathbf{v}\|_2 \leq \|\mathbf{v}\|_2$ for every $\mathbf{v}$). Also note that the matrix norm satisfies that for every two n by n matrices A, B, $\|A+B\| \leq \|A\|+\|B\|$ and $\|AB\| \leq \|A\|\|B\|$.

Lemma 16.14
Let A be a normalized adjacency matrix of an (n, d, λ)-graph G. Let J be the adjacency matrix of the n-clique with self loops (i.e., $J_{i,j} = 1/n$ for every i, j). Then

$$A = (1-\lambda)J + \lambda C \tag{2}$$

where $\|C\| \leq 1$.

Note that for every probability vector $\mathbf{p}$, $J\mathbf{p}$ is the uniform distribution, and so this lemma tells us that in some sense, we can think of a step on a (n, d, λ)-graph as going to the uniform distribution with probability $1-\lambda$, and to a different distribution with probability λ. This is of course not completely accurate, as a step on a d-regular graph will only go the one of the d neighbors of the current vertex, but we'll see that for the purposes of our analysis, the condition (2) will be just as good.[5]

Proof of Lemma 16.14: Indeed, simply define $C = \frac{1}{\lambda}(A - (1-\lambda)J)$. We need to prove $\|C\mathbf{v}\|_2 \leq \|\mathbf{v}\|_2$ for very $\mathbf{v}$. Decompose $\mathbf{v}$ as $\mathbf{v} = \mathbf{u} + \mathbf{w}$ where u is $\alpha\mathbf{1}$ for some α and $\mathbf{w} \perp \mathbf{1}$, and $\|\mathbf{v}\|_2^2 = \|\mathbf{u}\|_2^2 + \|\mathbf{w}\|_2^2$. Since $A\mathbf{1} = \mathbf{1}$ and $J\mathbf{1} = \mathbf{1}$ we get that $C\mathbf{u} = \frac{1}{\lambda}(\mathbf{u} - (1-\lambda)\mathbf{u}) = \mathbf{u}$. Now, let $\mathbf{w}' = A\mathbf{w}$.

[5]Algebraically, the reason (2) is not equivalent to going to the uniform distribution in each step with probability $1-\lambda$ is that C is not necessarily a stochastic matrix, and may have negative entries.

Then $\|\mathbf{w}'\|_2 \leq \lambda\|\mathbf{w}\|_2$ and, as we saw in the proof of Lemma 16.5, $\mathbf{w}' \perp \mathbf{1}$. Furthermore, since the sum of the coordinates of $\mathbf{w}$ is zero, $J\mathbf{w} = \mathbf{0}$. We get that $C\mathbf{w} = \frac{1}{\lambda}\mathbf{w}'$. Since $\mathbf{w}' \perp \mathbf{u}$, $\|C\mathbf{w}\|_2^2 = \|\mathbf{u} + \frac{1}{\lambda}\mathbf{w}'\|_2^2 = \|\mathbf{u}\|_2^2 + \|\frac{1}{\lambda}\mathbf{w}'\|_2^2 \leq \|\mathbf{u}\|_2^2 + \|\mathbf{w}\|_2^2 = \|\mathbf{w}\|_2^2$. ■

Returning to the proof of Theorem 16.12, we can write $\hat{B}A = \hat{B}\big((1-\lambda)J + \lambda C\big)$, and hence $\|\hat{B}A\| \leq (1-\lambda)\|\hat{B}J\| + \lambda\|\hat{B}C\|$. Since J's output is always a vector of the form $\alpha\mathbf{1}$, $\|\hat{B}J\| \leq \sqrt{\beta}$. Also, because $\hat{B}$ is an operation that merely zeros out some parts of its input, $\|\hat{B}\| \leq 1$ implying $\|\hat{B}C\| \leq 1$. Thus, $\|\hat{B}A\| \leq (1-\lambda)\sqrt{\beta}+\lambda$. Since $B\mathbf{1}$ has the value $1/N$ in $|B|$ places, $\|B\mathbf{1}\|_2 = \frac{\sqrt{\beta}}{\sqrt{N}}$, and hence $\|(\hat{B}A)^{k-1}\hat{B}\mathbf{1}\|_2 \leq ((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}\frac{\sqrt{\beta}}{\sqrt{N}}$, establishing (1). ■

The analysis of the error reduction procedure for algorithms with *two-sided errors* uses the following theorem, whose proof we omit:

Theorem 16.15 (Expander Chernoff Bound [**?**])
Let G be an (N,d,λ)-graph and $B \subseteq [N]$ with $|B| = \beta N$. Let $X_1,\ldots,X_k$ be random variables denoting a $k-1$-step random walk in G (where X_1 is chosen uniformly). For every $i \in [k]$, define B_i to be 1 if $X_i \in B$ and 0 otherwise. Then, for every $\delta > 0$,

$$\Pr\left[\left|\frac{\sum_{i=1}^k B_i}{k} - \beta\right| > \delta\right] < 2e^{(1-\lambda)\delta^2 k/60}$$

16.3.2 Combinatorial expansion and existence of expanders.

We describe now a combinatorial criteria that is roughly equivalent to Definition 16.9. One advantage of this criteria is that it makes it easy to prove that a non-explicit expander family exists using the probabilistic method. It is also quite useful in several applications.[6]

Definition 16.16 (Combinatorial (edge) expansion)
An n-vertex d-regular graph $G = (V,E)$ is called an (n,d,ρ)-*combinatorial expander* if for every subset $S \subseteq V$ with $|S| \leq n/2$, $|E(S,\overline{S})| \geq \rho d|S|$, where

[6]In our informal discussion above we defined combinatorial expansion by counting the number of *neighboring vertices* of a set S of vertices that are outside the set (this is known as *vertex expansion*). In contrast, Definition 16.16 below counts the number of *edges* that lie between S and its complement (this is known as *edge expansion*). However, these two numbers are clearly related by a factor of up to the degree d, which is not significant for our purposes.

for subsets S, T of V, $E(S,T)$ denotes the set of edges (s,t) with $s \in S$ and $t \in T$.

Note that in this case the bigger ρ is the better the expander. We'll loosely use the term expander for any (n,d,ρ)-combinatorial expander with c a positive constant. Using the probabilistic method, one can prove the following theorem: (Exercise 3 asks you to prove a slightly weaker version)

Theorem 16.17 (Existence of expanders)
Let $\epsilon > 0$ be some constant. Then there exists $d = d(\epsilon)$ and $N \in \mathbb{N}$ such that for every $n > N$ there exists an $(n,d,1-\epsilon)$-combinatorial expander.

The following theorem related combinatorial expansion with our previous Definition 16.9

Theorem 16.18 (Combinatorial and algebraic expansion)

1. *If G is an (n,d,λ)-graph then it is an $(n,d,(1-\lambda)/2)$-combinatorial expander.*
2. *If G is an (n,d,ρ)-combinatorial expander then it is an $(n,d,1-\frac{\rho^2}{2})$-graph.*

The first part of Theorem 16.18 follows by plugging $T = \overline{S}$ into the following lemma:

Lemma 16.19 (Expander Mixing Lemma)
Let $G = (V,E)$ be an (n,d,λ)-graph. Let $S, T \subseteq V$, then

$$\left| |E(S,T)| - \frac{d}{n}|S||T| \right| \leq \lambda d \sqrt{|S||T|}$$

Proof: Let $\mathbf{s}$ denote the vector such that $\mathbf{s}_i$ is equal to 1 if $i \in S$ and equal to 0 otherwise, and let $\mathbf{t}$ denote the corresponding vector for the set S. Thinking of $\mathbf{s}$ as a row vector and of $\mathbf{t}$ as a column vector, the Lemma's statement is equivalent to

$$\left| \mathbf{s}A\mathbf{t} - \tfrac{|S||T|}{n} \right| \leq \lambda \sqrt{|S||T|}\,, \tag{3}$$

where A is G's normalized adjacency matrix. Yet by Lemma 16.14, we can write A as $(1-\lambda)J + \lambda C$, where J is the matrix with all entries equal to $1/n$ and C has norm at most one. Hence,

$$\mathbf{s}A\mathbf{t} = (1-\lambda)\mathbf{s}J\mathbf{t} + \lambda \mathbf{s}C\mathbf{t} \leq \tfrac{|S||T|}{n} + \lambda\sqrt{|S||T|}\,,$$

where the last inequality follows from $\mathbf{s}J\mathbf{t} = |S||T|/n$ and $\mathbf{s}C\mathbf{t} = \langle \mathbf{s}, C\mathbf{t}\rangle \leq \|\mathbf{s}\|_2 \|\mathbf{t}\|_2 = \sqrt{|S||T|}$. ■

Proof of second part of Theorem 16.18.: We prove a slightly relaxed version, replacing the constant 2 with 8. Let $G = (V, E)$ be an n-vertex d-regular graph such that for every subset $S \subseteq V$ with $|S| \leq n/2$, there are $\rho|S|$ edges between S and $\overline{S} = V \setminus S$, and let A be G's normalized adjacency matrix.

Let $\lambda = \lambda(G)$. We need to prove that $\lambda \leq 1 - \rho^2/8$. Using the fact that λ is the second eigenvalue of A, there exists a vector $\mathbf{u} \perp \mathbf{1}$ such that $A\mathbf{u} = \lambda\mathbf{u}$. Write $\mathbf{u} = \mathbf{v} + \mathbf{w}$ where $\mathbf{v}$ is equal to $\mathbf{u}$ on the coordinates on which $\mathbf{u}$ is positive and equal to 0 otherwise, and $\mathbf{w}$ is equal to $\mathbf{u}$ on the coordinates on which $\mathbf{u}$ is negative, and equal to 0 otherwise. Note that, since $\mathbf{u} \perp \mathbf{1}$, both $\mathbf{v}$ and $\mathbf{w}$ are nonzero. We can assume that $\mathbf{u}$ is nonzero on at most $n/2$ of its coordinates (as otherwise we can take $-\mathbf{u}$ instead of $\mathbf{u}$).

Since $A\mathbf{u} = \lambda\mathbf{u}$ and $\langle \mathbf{v}, \mathbf{w}\rangle = 0$,

$$\langle A\mathbf{v}, \mathbf{v}\rangle + \langle A\mathbf{w}, \mathbf{v}\rangle = \langle A(\mathbf{v}+\mathbf{w}), \mathbf{v}\rangle = \langle A\mathbf{u}, \mathbf{v}\rangle = \langle \lambda(\mathbf{v}+\mathbf{w}), \mathbf{v}\rangle = \lambda\|\mathbf{v}\|_2^2 .$$

Since $\langle A\mathbf{w}, \mathbf{v}\rangle$ is negative, we get that $\langle A\mathbf{v}, \mathbf{v}\rangle / \|\mathbf{v}\|_2^2 \geq \lambda$ or

$$1 - \lambda \geq 1 - \frac{\langle A\mathbf{v}, \mathbf{v}\rangle}{\|\mathbf{v}\|_2^2} = \frac{\|\mathbf{v}\|_2^2 - \langle A\mathbf{v}, \mathbf{v}\rangle}{\|\mathbf{v}\|_2^2} = \frac{\sum_{i,j} A_{i,j}(\mathbf{v}_i - \mathbf{v}_j)^2}{2\|\mathbf{v}\|_2^2} ,$$

where the last equality is due to $\sum_{i,j} A_{i,j}(\mathbf{v}_i - \mathbf{v}_j)^2 = \sum_{i,j} A_{i,j}\mathbf{v}_i^2 - 2\sum_{i,j} A_{i,j}\mathbf{v}_i\mathbf{v}_j + \sum_{i,j} A_{i,j}\mathbf{v}_j^2 = 2\|\mathbf{v}\|_2^2 - 2\langle A\mathbf{v}, \mathbf{v}\rangle$. (We use here the fact that each row and column of A sums to one.) Multiply both numerator and denominator by $\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 + \mathbf{v}_j^2)$. By the Cauchy-Schwartz inequality,[7] we can bound the new numerator as follows:

$$\left(\sum_{i,j} A_{i,j}(\mathbf{v}_i - \mathbf{v}_j)^2\right)\left(\sum_{i,j} A_{i,j}(\mathbf{v}_i + \mathbf{v}_j)^2\right) \leq \left(\sum_{i,j} A_{i,j}(\mathbf{v}_i - \mathbf{v}_j)(\mathbf{v}_i + \mathbf{v}_j)\right)^2 .$$

[7]The Cauchy-Schwartz inequality is typically stated as saying that for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\sum_i \mathbf{x}_i\mathbf{y}_i \leq \sqrt{(\sum_i \mathbf{x}_i^2)(\sum_i \mathbf{y}_i^2)}$. However, it is easily generalized to show that for every non-negative $\mu_1, \ldots, \mu_n$, $\sum_i \mu_i\mathbf{x}_i\mathbf{y}_i \leq \sqrt{(\sum_i \mu_i\mathbf{x}_i^2)(\sum_i \mu_i\mathbf{y}_i^2)}$ (this can be proven from the standard Cauchy-Schwartz by multiplying each coordinate of $\mathbf{x}$ and $\mathbf{y}$ by $\sqrt{\mu_i}$. It is this variant that we use here with the $A_{i,j}$'s playing the role of $\mu_1, \ldots, \mu_n$.

Hence, using $(a-b)(a+b) = a^2 - b^2$,

$$1-\lambda \geq \frac{\left(\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2)\right)^2}{2\|\mathbf{v}\|_2^2 \sum_{i,j} A_{i,j}(\mathbf{v}_i + \mathbf{v}_j)^2} = \frac{\left(\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2)\right)^2}{2\|\mathbf{v}\|_2^2 \left(\sum_{i,j} A_{i,j}\mathbf{v}_i^2 + 2\sum_{i,j} A_{i,j}\mathbf{v}_i\mathbf{v}_j + \sum_{i,j} A_{i,j}\mathbf{v}_j^2\right)} =$$

$$\frac{\left(\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2)\right)^2}{2\|\mathbf{v}\|_2^2 \left(2\|\mathbf{v}\|_2^2 + 2\langle A\mathbf{v}, \mathbf{v}\rangle\right)} \geq \frac{\left(\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2)\right)^2}{8\|\mathbf{v}\|_2^4},$$

where the last inequality is due to A having matrix norm at most 1, implying $\langle A\mathbf{v}, \mathbf{v}\rangle \leq \|\mathbf{v}\|_2^2$. We conclude the proof by showing that

$$\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2) \geq \rho\|\mathbf{v}\|_2^2, \tag{4}$$

which indeed implies that $1-\lambda \geq \frac{\rho^2\|\mathbf{v}\|_2^4}{8\|\mathbf{v}\|_2^4} = \frac{\rho^2}{8}$.

To prove (4) sort the coordinates of $\mathbf{v}$ so that $\mathbf{v}_1 \geq \mathbf{v}_2 \geq \cdots \geq \mathbf{v}_n$ (with $\mathbf{v}_i = 0$ for $i > n/2$). Then

$$\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2) \geq \sum_{i=1}^{n/2}\sum_{j=i+1}^{n} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_{i+1}^2) = \sum_{i=1}^{n/2} c_i(\mathbf{v}_i^2 - \mathbf{v}_{i+1}^2),$$

where c_i denotes $\sum_{j>i} A_{i,j}$. But c_i is equal to the number of edges in G from the set $\{k : k \leq i\}$ to its complement, divided by d. Hence, by the expansion of G, $c_i \geq \rho i$, implying (using the fact that $\mathbf{v}_i = 0$ for $i \geq n/2$):

$$\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2) \geq \sum_{i=1}^{n/2} \rho i(\mathbf{v}_i^2 - \mathbf{v}_{i+1}^2) = \sum_{i=1}^{n/2}(\rho i\mathbf{v}_i^2 - \rho\cdot(i-1)\mathbf{v}_i^2) = \rho\|\mathbf{v}\|_2^2,$$

establishing (4). ■

16.4 Graph products and expansion

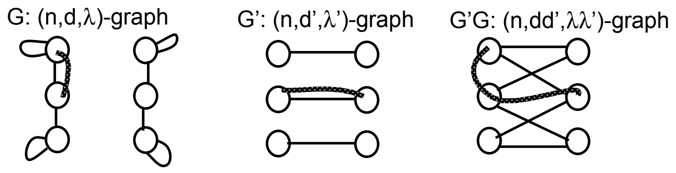
A *graph product* is an operation that takes two graphs G, G' and outputs a graph H. Typically we're interested in the relation between properties of the graphs G, G' to the properties of the resulting graph H. In this section we will mainly be interested in three parameters: the number of vertices (denoted n), the degree (denoted d), and the 2^{nd} largest eigenvalue of the normalized adjacency matrix (denoted λ), and study how different products affect these parameters. In the next sections we will use these products for two applications: **(1)** A construction of a strongly explicit expander graph family and **(2)** A *deterministic* logspace algorithm for undirected connectivity.

16.4.1 Rotation maps.

In addition to the adjacency matrix representation, we can also represent an n-vertex degree-d graph G as a function $\hat{G}$ from $[n] \times [d]$ to $[n]$ that given a pair $\langle v, i \rangle$ outputs u where the i^{th} neighbor of v in G. In fact, it will be convenient for us to have $\hat{G}$ output an additional value $j \in [d]$ where j is the index of v as a neighbor of u. Given this definition of $\hat{G}$ it is clear that we can invert it by applying it again, and so it is a permutation on $[n] \times [d]$. We call $\hat{G}$ the *rotation map* of G. For starters, one may think of the case that $\hat{G}(u, i) = (v, i)$ (i.e., v is the i^{th} neighbor of u iff u is the i^{th} neighbor of v). In this case we can think of $\hat{G}$ as operating only on the vertex. However, we will need the more general notion of a rotation map later on.

We can describe a graph product in the language of graphs, adjacency matrices, or rotation maps. Whenever you see the description of a product in one of this forms (e.g., as a way to map two graphs into one), it is a useful exercise to work out the equivalent descriptions in the other forms (e.g., in terms of adjacency matrices and rotation maps).

16.4.2 The matrix/path product

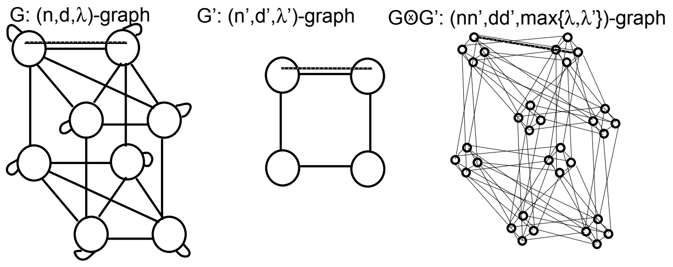


For every two n vertex graphs G, G' with degrees d, d' and adjacency matrices A, A', the graph $G'G$ is the graph described by the adjacency matrix $A'A$. That is, $G'G$ has an edge (u, v) for every length 2-path from u to v where the first step in the path is taken on en edge of G and the second is on an edge of G'. Note that G has n vertices and degree dd'. Typically, we are interested in the case $G = G'$, where it is called *graph squaring*. More generally, we denote by G^k the graph $G \cdot G \cdots G$ (k times). We already encountered this case before in Lemma 16.5, and similar analysis yields the following lemma (whose proof we leave as exercise):

Lemma 16.20 (Matrix product improves expansion)
$\lambda(G'G) \leq \lambda(G')\lambda(G')$

It is also not hard to compute the rotation map of $G'G$ from the rotation maps of G and G'. Again, we leave verifying this to the reader.

16.4.3 The tensor product



Let G and G' be two graphs with n (resp n') vertices and d (resp. d') degree, and let $\hat{G} : [n] \times [d] \to [n] \times [d]$ and $\hat{G}' : [n'] \times [d'] \to [n'] \times [d']$ denote their respective *rotation maps*. The *tensor product* of G and G', denoted $G \otimes G'$, is the graph over nn' vertices and degree dd' whose rotation map $G \hat{\otimes} G'$ is the permutation over $([n] \times [n']) \times ([d] \times [d'])$ defined as follows

$$G \hat{\otimes} G'(\langle u, v\rangle, \langle i, j\rangle) = \langle u', v'\rangle, \langle i', j'\rangle\,,$$

where $(u', i') = \hat{G}(u, i)$ and $(v', j') = \hat{G}'(v, j)$. That is, the vertex set of $G \otimes G'$ is pairs of vertices, one from G and the other from G', and taking a the step $\langle i, j\rangle$ on $G \otimes G'$ from the vertex $\langle u, v\rangle$ is akin to taking two independent steps: move to the pair $\langle u', v'\rangle$ where u' is the i^{th} neighbor of u in G and v' is the i^{th} neighbor of v in G'.

In terms of adjacency matrices, the tensor product is also quite easy to describe. If $A = (a_{i,j})$ is the $n \times n$ adjacency matrix of G and $A' = (a'_{i',j'})$ is the $n' \times n'$ adjacency matrix of G', then the adjacency matrix of $G \otimes G'$, denoted as $A \otimes A'$, will be an $nn' \times nn'$ matrix that in the $\langle i, i'\rangle^{th}$ row and the $\langle j, j'\rangle$ column has the value $a_{i,j} \cdot a'_{i',j'}$. That is, $A \otimes A'$ consists of n^2

copies of A', with the $(i,j)^{th}$ copy scaled by $a_{i,j}$:

$$A \otimes A' = \begin{pmatrix} a_{1,1}A' & a_{2,1}A' & \dots & a_{1,n}A' \\ a_{2,1}A' & a_{2,2}A' & \dots & a_{2,n}A' \\ \vdots & & & \vdots \\ a_{n,1}A' & a_{n,2}A' & \dots & a_{n,n}A' \end{pmatrix}$$

The tensor product can also be described in the language of graphs as having a cluster of n' vertices in $G \otimes G'$ for every vertex of G. Now if, u and v are two neighboring vertices in G, we will put a bipartite version of G' between the cluster corresponding to u and the cluster corresponding to v in G. That is, if (i,j) is an edge in G' then there is an edge between the i^{th} vertex in the cluster corresponding to u and the j^{th} vertex in the cluster corresponding to v.

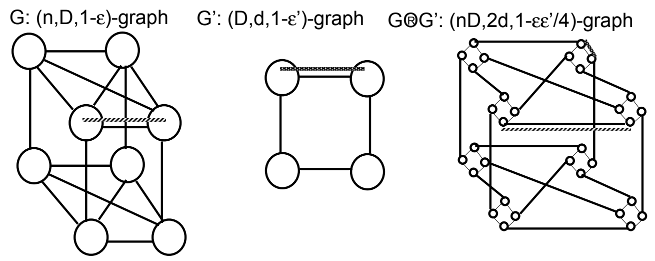
Lemma 16.21 (Tensor product preserves expansion)
Let $\lambda = \lambda(G)$ and $\lambda' = \lambda(G')$ then $\lambda(G \otimes G') \leq \max\{\lambda, \lambda'\}$.

One intuition for this bound is the following: taking a T step random walk on the graph $G \otimes G'$ is akin to taking two independent random walks on the graphs G and G'. Hence, if both walks converge to the uniform distribution within T steps, then so will the walk on $G \otimes G'$.

Proof: Given some basic facts about tensor products and eigenvalues this is immediate since if $\lambda_1, \dots, \lambda_n$ are the eigenvalues of A (where A is the adjacency matrix of G) and $\lambda'_1, \dots, \lambda'_{n'}$ are the eigenvalues of A (where A' is the adjacency matrix of G'), then the eigenvalues of $A \otimes A'$ are all numbers of the form $\lambda_i \cdot \lambda'_j$, and hence the largest ones apart from 1 are of the form $1 \cdot \lambda(G')$ or $\lambda(G) \cdot 1$ (see also Exercise 4). ■

We note that one can show that $\lambda(G \otimes G') \leq \lambda(G) + \lambda(G')$ without relying on any knowledge of eigenvalues (see Exercise 5). This weaker bound suffices for our applications.

16.4.4 The replacement product



In both the matric and tensor products, the degree of the resulting graph is larger than the degree of the input graphs. The following product will enable us to reduce the degree of one of the graphs. Let G, G' be two graphs such that G has n vertices and degree D, and G' has D vertices and degree d. The *balanced replacement product* (below we use simply *replacement product* for short) of G and G' is denoted by $G \circledR G'$ is the nn'-vertex $2d$-degree graph obtained as follows:

1. For every vertex u of G, the graph $G \circledR G'$ has a copy of G' (including both edges and vertices).

2. If u, v are two neighboring vertices in G then we place d parallel edges between the i^{th} vertex in the copy of G' corresponding to u and the j^{th} vertex in the copy of G' corresponding to v, where i is the index of v as a neighbor of u and j is the index of u as a neighbor of v in G. (That is, taking the i^{th} edge out of u leads to v and taking the j^{th} edge out of v leads to u.)

Note that we essentially already encountered this product in the proof of Claim 16.1.1 (see also Figure 16.2), where we reduced the degree of an arbitrary graph by taking its replacement product with a cycle (although there we did not use parallel edges).[8] The replacement product also has a simple description in terms of rotation maps: since $G \circledR G'$ has nD vertices and $2d$ degree, its rotation map $G \hat{\circledR} G'$ is a permutation over $([n] \times [D]) \times ([d] \times \{0,1\})$ and so can be thought of as taking four inputs u, v, i, b where $u \in [n]$, $v \in [D]$, $i \in [d]$ and $b \in \{0,1\}$. If $b = 0$ then it outputs $u, \hat{G}'(v,i), b$ and if $b = 1$ then it outputs $\hat{G}(u,v), i, b$. That is, depending on whether b is equal to 0 or 1, the rotation map either treats v as a vertex of G' or as an edge label of G.

In the language of adjacency matrices the replacement product can be easily seen to be described as follows: $A \circledR A' = 1/2(A \otimes I_D) + 1/2(I_n \otimes A')$, where A, A' are the adjacency matrices of the graphs G and G' respectively, and I_k is the $k \times k$ identity matrix.

If $D \gg d$ then the replacement product's degree will be significantly smaller than G's degree. The following Lemma shows that this dramatic degree reduction does not cause too much of a deterioration in the graph's expansion:

[8]The addition of parallel edges ensures that a random step from a vertex v in $G \circledR G'$ will move to a neighbor within the same cluster and a neighbor outside the cluster with the same probability. For this reason, we call this product the *balanced* replacement product.

Lemma 16.22 (Expansion of replacement product)
If $\lambda(G) \leq 1-\epsilon$ and $\lambda(G') \leq 1-\epsilon'$ then $\lambda(G \circledR G') \leq 1-\epsilon\epsilon'/4$.

The intuition behind Lemma 16.22 is the following: Think of the input graph G as a good expander whose only drawback is that it has a too high degree D. This means that a k step random walk on G' requires $O(k \log D)$ random bits. However, as we saw in Section 16.3.1, sometimes we can use fewer random bits if we use an expander. So a natural idea is to generate the edge labels for the walk by taking a walk using a smaller expander G' that has D vertices and degree $d \ll D$. The definition of $G \circledR G'$ is motivated by this intuition: a random walk on $G \circledR G'$ is roughly equivalent to using an expander walk on G' to generate labels for a walk on G. In particular, each step a walk over $G \circledR G'$ can be thought of as tossing a coin and then, based on its outcome, either taking a a random step on G', or using the current vertex of G' as an edge label to take a step on G. Another way to gain intuition on the replacement product is to solve Exercise 6, that analyzes the *combinatorial* (edge) expansion of the resulting graph as a function of the edge expansion of the input graphs.

Proof of Lemma 16.22: Let A (resp. A') denote the $n \times n$ (resp. $D \times D$) adjacency matrix of G (resp. G') and let $\lambda(A) = 1-\epsilon$ and $\lambda(A') = 1-\epsilon'$. Then by Lemma 16.14, $A = (1-\epsilon)C + J_n$ and $A' = (1-\epsilon')C' + J_D$, where J_k is the $k \times k$ matrix with all entries equal to $1/k$.

The adjacency matrix of $G \circledR G'$ is equal to

$$\tfrac{1}{2}(A \otimes I_D) + \tfrac{1}{2}(I_n \otimes A') = \tfrac{1-\epsilon}{2} C \otimes I_D + \tfrac{\epsilon}{2} J_n \otimes I_D + \tfrac{1-\epsilon'}{2} I_n \otimes C' + \tfrac{\epsilon'}{2} I_n \otimes J_D\,,$$

where I_k is the $k \times k$ identity matrix.

Thus, the adjacency matrix of $(G \circledR G')^2$ is equal to

$$\left(\tfrac{1-\epsilon}{2} C \otimes I_D + \tfrac{\epsilon}{2} J_n \otimes I_D + \tfrac{1-\epsilon'}{2} I_n \otimes C' + \tfrac{\epsilon'}{2} I_n \otimes J_D\right)^2 = \\ \tfrac{\epsilon\epsilon'}{4}(J_n \otimes I_D)(I_n \otimes J_D) + \tfrac{\epsilon'\epsilon}{4}(I_n \otimes J_D)(J_n \otimes I_D) + (1 - \tfrac{\epsilon\epsilon'}{2})F\,,$$

where F is some $nD \times nD$ matrix of norm at most 1 (obtained by collecting together all the other terms in the expression). But

$$(J_n \otimes I_D)(I_n \otimes J_D) = (I_n \otimes J_D)(J_n \otimes I_D) = J_n \otimes J_D = J_{nD}\,.$$

(This can be verified by either direct calculation or by going through the graphical representation or the rotation map representation of the tensor and matrix products.)

Since every vector $\mathbf{v} \in \mathbb{R}^{nD}$ that is orthogonal to $\mathbf{1}$ satisfies $J_{nD}\mathbf{v} = \mathbf{0}$, we get that

$$\left(\lambda(G \circledR G')\right)^2 = \lambda\left((G \circledR G')^2\right) = \lambda\left((1 - \tfrac{\epsilon\epsilon'}{2})F + \tfrac{\epsilon\epsilon'}{2}J_{nD}\right) \leq 1 - \tfrac{\epsilon\epsilon'}{2}\,,$$

and hence

$$\lambda(G \circledR G') \leq 1 - \tfrac{\epsilon\epsilon'}{4}\,.$$

■

16.5 Explicit construction of expander graphs.

We now use the three graph products of Section 16.4 to show a strongly explicit construction of an expander graph family. Recall This is an infinite family $\{G_k\}$ of graphs (with efficient way to compute neighbors) that has a constant degree and an expansion parameter λ. The construction is recursive: we start by a finite size graph G_1 (which we can find using brute force search), and construct the graph G_k from the graph G_{k-1}. On a high level the construction is as follows: each of the three product will serve a different purpose in the construction. The *Tensor product* allows us to take G_{k-1} and increase its number of vertices, at the expense of increasing the degree and possibly some deterioration in the expansion. The *replacement product* allows us to dramatically reduce the degree at the expense of additional deterioration in the expansion. Finally, we use the *Matrix/Path product* to regain the loss in the expansion at the expense of a mild increase in the degree.

THEOREM 16.23 (EXPLICIT CONSTRUCTION OF EXPANDERS)
There exists a strongly-explicit λ, d-expander family for some constants d and $\lambda < 1$.

PROOF: Our expander family will be the following family $\{G_k\}_{k\in\mathbb{N}}$ of graphs:

- Let H be a $(D = d^{40}, d, 0.01)$-graph, which we can find using brute force search. (We choose d to be a large enough constant that such a graph exists)
- Let G_1 be a $(D, d^{20}, 1/2)$-graph, which we can find using brute force search.

- For $k > 1$, let $G_k = ((G_{k-1} \otimes G_{k-1}) \textcircled{z} H)^{20}$.

The proof follows by noting the following points:

1. For every k, G_k has at least 2^{2^k} vertices.

   Indeed, if n_k denotes the number of vertices of G_k, then $n_k = (n_{k-1})^2 D$. If $n_{k-1} \geq 2^{2^{k-1}}$ then $n_k \geq \left(2^{2^{k-1}}\right)^2 = 2^{2^k}$.

2. For every k, the degree of G_k is d^{20}.

   Indeed, taking a replacement produce with H reduces the degree to d, which is then increased to d^{20} by taking the 20^{th} power of the graph (using the matrix/path product).

3. There is a $2^{O(k)}$-time algorithm that given a label of a vertex u in G_k and an index $i \in [d^{20}]$, outputs the i^{th} neighbor of u in G_k. (Note that this is polylogarithmic in the number of vertices.)

   Indeed, such a recursive algorithm can be directly obtained from the definition of G_k. To compute G_k's neighborhood function, the algorithm will make 40 recursive calls to G_{k-1}'s neighborhood function, resulting in $2^{O(k)}$ running time.

4. For every k, $\lambda(G_k) \leq 1/3$.

   Indeed, by Lemmas 16.20, 16.21, and 16.22 If $\lambda(G_{k-1}) \leq 1/3$ then $\lambda(G_{k-1} \otimes G_{k-1}) \leq 2/3$ and hence $\lambda((G_{k-1} \otimes G_{k-1}) \textcircled{R} H) \leq 1 - \frac{0.99}{12} \leq 1 - 1/13$. Thus, $\lambda(G_k) \leq (1 - 1/13)^{20} \sim e^{-20/13} \leq 1/3$.

■

Using graph powering we can obtain such a construction for *every* constant $\lambda \in (0, 1)$, at the expense of a larger degree. There is a variant of the above construction supplying a *denser* family of graphs that contains an n-vertex graph for every n that is a power of c, for some constant c. Since one can transform an (n, d, λ)-graph to an (n', cd', λ)-graph for any $n/c \leq n' \leq n$ by making a single "mega-vertex" out of a set of at most c vertices, the following theorem is also known:

Theorem 16.24
There exist constants $d \in \mathbb{N}$, $\lambda < 1$ and a strongly-explicit graph family $\{G_n\}_{n \in \mathbb{N}}$ such that G_n is an (n, d, λ)-graph for every $n \in \mathbb{N}$.

Remark 16.25
As mentioned above, there are known constructions of expanders (typically based on number theory) that are more efficient in terms of computation time and relation between degree and the parameter λ than the product-based construction above. However, the proofs for these constructions are more complicated and require deeper mathematical tools. Also, the replacement product (and its close cousin, the zig-zag product) have found applications beyond the constructions of expander graphs. One such application is the deterministic logspace algorithm for undirected connectivity described in the next section. Another application is a construction of combinatorial expanders with greater expansion that what is implied by the parameter λ. (Note that even for for the impossible to achieve value of $\lambda = 0$, Theorem 16.1.1 implies combinatorial expansion only $1/2$ while it can be shown that a random graph has combinatorial expansion close to 1.)

16.6 Deterministic logspace algorithm for undirected connectivity.

The replacement product has a surprising consequence: a *deterministic* algorithm to determine whether two vertices are connected in a graph using only logarithmic space.

Theorem 16.26 (Reingold's theorem [?])
$\mathsf{UPATH} \in \mathbf{L}$.

The underlying intuition behind the logspace algorithm for UPATH is that checking connectivity in *expander* graphs is easy. More accurately, if every connected component in G is an expander, then there is a number $\ell = O(\log n)$ such that if s and t are connected then they are connected within a path of length at most ℓ. (Indeed, in this case an ℓ-step random walk from s will reach t with reasonable probability.) We can enumerate over all ℓ-step random walks of a d-degree graph in $O(d\ell)$ space by enumerating over all sequences of indices $i_1, \ldots, i_\ell \in [d]$. Thus, in a constant-degree graph where all connected components are expanders we can check connectivity in logarithmic space. The idea behind the algorithm will be to transform the graph G (in an implicitly computable in logspace way) to a graph G' such that every connected component in G becomes an expander in G', but two vertices that were not connected will stay unconnected. This transformation is reminiscent of the expander construction of the previous section.

Proof of Theorem 16.26: Let G be the input graph and s, t two vertices in G. Using the transformation of Claim 16.1.1, we may assume that G is a regular degree-4 graphs with self loops on all vertices. By adding more self-loops we may assume that the graph is of degree d^{20} for some constant d that is sufficiently large so that there exists a $(d^{20}, d, 0.01)$-graph H.

- Let H be a $(d^{20}, d, 0.01)$-graph, which we can find using brute force search.
- Let $G_0 = G$.
- For $k \geq 1$, we define $G_k = (G_{k-1} \circledR H)^{20}$.

Note that these operations do not connect vertices that were disconnected in G. Thus, we can analyze their effect separately on each connected component of G. By Lemmas 16.20 and 16.22, for every $\epsilon < 1/20$ and D-degree graph F, if $\lambda(F) \leq 1 - \epsilon$ then $\lambda(F \circledR H) \leq 1 - \epsilon/5$ and hence $\lambda\left((F \circledR H)^{20}\right) \leq 1 - 2\epsilon$.

By Lemma 16.6, every connected component of G has expansion parameter at most $1 - 1/(8Dn^3)$, where n denotes the number of G's vertices which is at least as large as the number of vertices in the connect component. It follows that for $k = 10 \log D \log N$, in the graph G_k every connected component has expansion parameter at most $\max\{1 - 1/20, 2^k/(8Dn^3)\} = 1 - 1/20$.

The space required to enumerate over ℓ length walks from some vertex s in G_k is $O(\ell)$ bits to store ℓ indices and the space to compute the rotation map of G_k. To finish the proof, we will show that we can compute this map in $O(k + \log n)$ space. This map's input length is $O(k + \log n)$ and hence we can assume it is placed on a read/write tape, and will compute the rotation map "in-place" changing the input to the output. Let s_k be the additional space (beyond the input) required to compute the rotation map of G_k. Note that $s_0 = O(\log n)$. We show a recursive algorithm to compute G_k satisfying the equation $s_k = s_{k-1} + O(1)$. In fact, the algorithm will be a pretty straightforward implementation of the definitions of the replacement and matrix products.

The input to $\hat{G}_k$ is a vertex in $(G_{k-1} \circledR H)$ and 20 labels of edges in this graph. If we can compute the rotation map of $G_{k-1} \circledR H$ in $s_{k-1} + O(1)$ space then we can do so for $\hat{G}_k$, since we can simply make 20 consecutive calls to this procedure, each time reusing the space.[9] Now, to compute the rotation

[9] One has to be slightly careful while making recursive calls, since we don't want to

map of $(G_{k-1} \circledR H)$ we simply follow the definition of the replacement product. Given an input of the form u, v, i, b (which we think of as read/write variables), if $b = 0$ then we apply the rotation map of H to (v, i) (can be done in constant space), while if $b = 1$ then we apply the rotation map of G_{k-1} to (u, v) using a recursive call at the cost of s_{k-1} space (note that u, v are conveniently located consecutively at the beginning of the input tape). ■

Chapter notes and history

STILL A LOT MISSING

Expanders were well-studied for a variety of reasons in the 1970s but their application to pseudorandomness was first described by Ajtai, Komlos, and Szemeredi [?]. Then Cohen-Wigderson [?] and Impagliazzo-Zuckerman (1989) showed how to use them to "recycle" random bits as described in Section 16.3.1. The upcoming book by Hoory, Linial and Wigderson (draft available from their web pages) provides an excellent introduction to expander graphs and their applications.

The explicit construction of expanders is due to Reingold, Vadhan and Wigderson [?], although we chose to present it using the replacement product as opposed to the closely related zig-zag product used there. The deterministic logspace algorithm for undirected connectivity is due to Reingold [?].

Exercises

§1 Let A be a symmetric stochastic matrix: $A = A^\dagger$ and every row and column of A has non-negative entries summing up to one. Prove that $\|A\| \leq 1$.

Hint: first show that $\|A\|$ is at most say n^2. Then, prove that for every $k \geq 1$, A^k is also stochastic and $\|A^{2k}\mathbf{v}\|_2 \geq \|A^k\mathbf{v}\|_2^2$ using the equality $\langle \mathbf{w}, B\mathbf{z} \rangle = \langle B^\dagger \mathbf{w}, \mathbf{z} \rangle$ and the inequality $\langle \mathbf{w}, \mathbf{z} \rangle \leq \|\mathbf{w}\|_2 \|\mathbf{z}\|_2$.

§2 Let A, B be two symmetric stochastic matrices. Prove that $\lambda(A+B) \leq \lambda(A) + \lambda(B)$.

lose even the $O(\log\log n)$ bits of writing down k and keeping an index to the location in the input we're working on. However, this can be done by keeping k in global read/write storage and since storing the identity of the current step among the 50 calls we're making only requires $O(1)$ space.

§3 Let a n, d random graph be an n-vertex graph chosen as follows: choose d random permutations $\pi_1, ldots, \pi_d$ from $[n]$ to $[n]$. Let the the graph G contains an edge (u, v) for every pair u, v such that $v = \pi_i(u)$ for some $1 \le i \le d$. Prove that a random n, d graph is an $(n, 2d, \frac{2}{3}d)$ combinatorial expander with probability $1 - o(1)$ (i.e., tending to one with n).

Hint: for every set $S \subseteq n$ with $|S| \le n/2$ and set $T \subseteq [n]$ with $|T| \le (1 + \frac{2}{3}d)|S|$, try to bound the probability that $\pi_i(S) \subseteq T$ for every i.

§4 Let A be an $n \times n$ matrix with eigenvectors $\mathbf{u}^1, \ldots, \mathbf{u}^n$ and corresponding values $\lambda_1, \ldots, \lambda_n$. Let B be an $m \times m$ matrix with eigenvectors $\mathbf{v}^1, \ldots, \mathbf{v}^m$ and corresponding values $\alpha_1, \ldots, \alpha_m$. Prove that the matrix $A \otimes B$ has eigenvectors $\mathbf{u}^i \otimes \mathbf{v}^j$ and corresponding values $\lambda_i \cdot \alpha_j$.

§5 Prove that for every two graphs G, G', $\lambda(G \otimes G') \le \lambda(G) + \lambda(G')$ without using the fact that every symmetric matrix is diagonalizable.

Hint: Use Lemma 16.14.

§6 Let G be an n-vertex D-degree graph with ρ combinatorial edge expansion for some $\rho > 0$. (That is, for every a subset S of G's vertices of size at most $n/2$, the number of edges between S and its complement is at least $\rho d|S|$.) Let G' be a D-vertex d-degree graph with ρ' combinatorial edge expansion for some $\rho' > 0$. Prove that $G \circledR G'$ has at least $\rho^2\rho'/1000$ edge expansion.

Hint: Every subset of $G \circledR G'$ can be thought of as n subsets of the individual clusters. Treat differently the subsets that take up more than $1 - \rho/10$ portion of their clusters and those that take up less than that. For the former use the expansion of G, while for the latter use the expansion of G'.