Computational Complexity: A Modern Approach

Sanjeev Arora and Boaz Barak
Princeton University

http://www.cs.princeton.edu/theory/complexity/

complexitybook@gmail.com

Chapter 21

Pseudorandom constructions: expanders and extractors

"How difficult could it be to find hay in a haystack?"
Howard Karloff

The probabilistic method is a powerful method to show the existence of objects (e.g., graphs, functions) with certain desirable properties. We have already seen it used in Chapter 6 to show the existence of functions with high circuit complexity, in Chapter 19 to show the existence of good error correcting codes, and in several other places in this book. But sometimes the mere *existence* of an object is not enough: we need an *explicit* and *efficient* construction. This chapter provides such constructions for two well-known (and related) families of pseudorandom objects, *expanders* and *extractors*. They are important in computer science because they can often be used to replace (or reduce) the amount of randomness needed in certain settings. This is reminiscent of derandomization, the topic of Chapter 20, and indeed we will see several connections to derandomization throughout the chapter. However, a big difference between Chapter 20 and this one is that all results proven here are *unconditional*, in other words do not rely on unproven assumptions. Another topic that is related to expanders is constructions of error correcting-codes and related hardness-amplification results which we saw in Chapter 19. For a brief discussion of the many deep and fascinating connections between codes, expanders, pseudorandom generators, and extractors, see the Chapter notes.

Expanders are graphs whose connectivity properties (how many edges lie between every two sets A, B of vertices) are similar to those of "random" graphs —in this sense they are "pseudorandom" or "like random." Expanders have found a vast number of applications ranging from fast sorting networks, to counterexamples in metric space theory, to proving the **PCP** Theorem. The study of expanders is closely tied to study of *eigenvalues* of adjacency matrices. In Section 21.1 we lay the groundwork for this study, showing how random walks on graphs can be analyzed in terms of the adjacency matrix's eigenvalues. Then in Section 21.2 we give two equivalent definitions for expander graphs. We also describe their use in randomness-efficient error reduction of probabilistic algorithms. In Section 21.3 we show an explicit construction of expander graphs. Finally, in Section 21.4, we use this construction to show a *deterministic* logspace algorithm for undirected graph connectivity.

Our second example of an explicit construction concerns the following issue: while randomized algorithms are modeled using a sequence of unbiased and independent coin tosses, real-life randomness sources are imperfect and have correlations and biases. Philosophically speaking, it is unclear if there is even a single source of unbiased random bits in the universe. Therefore researchers have tried to quantify ways in which a source of random bits could be imperfect and still be used to run randomized algorithms.

In Section 21.5 we define *weakly random* sources. This definition encapsulates the minimal notion of "randomness" that still allows an imperfect source to be used in randomized algorithms. We also define *randomness extractors* (or extractors for short)— algorithms

that extract unbiased random bits from such a source — and give explicit constructions for them. One philosophical consequence of these results is that the model of randomized polynomial-time Turing machines (and the associated classes like **BPP**) is realistic if and only if weakly random sources exist in the real world.

In Section 21.6 we use extractors to derandomize probabilistic logspace computations, albeit at the cost of some increase in the space requirement. We emphasize that in contrast to the results of Chapters 19 and 20, this derandomization (as well as all the other results of this chapter) is *unconditional* and uses no unproven assumptions.

Both the constructions and analysis of this chapter are somewhat involved. You might wonder why should coming up with explicit construction be so difficult. After all, a proof of existence via the probabilistic method shows not only that an object with the desired property exists but in fact the vast majority of objects have the property. As Karloff said (see quote above), how difficult can it be to find a single one? But perhaps it's not so surprising this task is so difficult: after all, we know that almost all Boolean functions have exponential circuit complexity, but finding even a single one in **NP** with this property will show that $\mathbf{P} \neq \mathbf{NP}$!

21.1 Random walks and eigenvalues

In this section we study random walks on graphs. Using elementary linear algebra we relate eigenvalues of the graph's adjacency matrix to the behavior of the random walk on that graph. As a corollary we obtain the proof of correctness for the random-walk space-efficient algorithm for undirected connectivity described in Section 7.7. We restrict ourselves here to *regular* graphs, in which every vertex has the same degree. However, we do allow our graphs to have self-loops and parallel edges. Most of the definitions and results can be suitably generalized to undirected graphs that are not regular.

Some linear algebra. We will use some basic properties of the linear space $\mathbb{R}^n$. These are covered in Section A.5 of the appendix, but here is a quick review. If $\mathbf{u}, \mathbf{v} \in R^n$ are two vectors, then their *inner product* is defined as $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{n} \mathbf{u}_i \mathbf{v}_i$. We say that $\mathbf{u}$ and $\mathbf{v}$ are *orthogonal*, denoted by $\mathbf{u} \perp \mathbf{v}$, if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$. The *$L_2$-norm* of a vector $\mathbf{v} \in \mathbb{R}^n$, denoted by $\|\mathbf{v}\|_2$ is defined as $\sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{\sum_{i=1}^{n} \mathbf{v}_i^2}$. A vector whose L_2-norm equals 1 is called a *unit vector*. A simple but useful fact is the *Pythagorean Theorem*, that says that if $\mathbf{u}$ and $\mathbf{v}$ are orthogonal then $\|\mathbf{u} + \mathbf{v}\|_2^2 = \|\mathbf{u}\|_2^2 + \|\mathbf{v}\|_2^2$. The *$L_1$-norm* of $\mathbf{v}$, denoted by $|\mathbf{v}|_1$ is defined as $\sum_{i=1}^{n} |\mathbf{v}_i|$. Both these norms satisfy the basic properties **(1)** $\|\mathbf{v}\| > 0$ with $\|\mathbf{v}\| = 0$ iff $\mathbf{v}$ is the all zero vector, **(2)** $\|\alpha \mathbf{v}\| = |\alpha| \|\mathbf{v}\|$ for every $\alpha \in \mathbb{R}$, and **(3)** $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$. The relation between these norms is captured in the following claim, whose proof is left as Exercise 21.1:

Claim 21.1 *For every vector $\mathbf{v} \in \mathbb{R}^n$,*

$$|\mathbf{v}|_1 / \sqrt{n} \leq \|\mathbf{v}\|_2 \leq |\mathbf{v}|_1 . \qquad \diamondsuit$$

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

Let G be a d-regular n-vertex graph and 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 $\mathbf{p}$ is equal to 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 easily compute $\mathbf{q}$, since the probability $\mathbf{q}_j$ that j is chosen is equal to the sum over all of j's neighbors i of the probability $\mathbf{p}_i$ that i is chosen times $1/d$ (since vertex i touches d edges, for each edge $\overline{i\,j}$ the probability that conditioned on i being chosen then the next move will take this edge is $1/d$). Thus $\mathbf{q} = A\mathbf{p}$,

where $A = A(G)$ is the matrix such that for every two vertices i, j of G, $A_{i,j}$ is equal to the number of edges between i and j divided by d. (In other words, A is equal to the adjacency matrix of G multiplied by $1/d$.) We call A the *random-walk matrix* of G. Note that A is a symmetric matrix[1] with all its entries 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.

The relation between the matrix A and random walks on the graph G is straightforward—for every $\ell \in \mathbb{N}$ and $i \in [n]$, the vector $A^\ell \mathbf{e}^i$ (where $\mathbf{e}^i$ is the vector that has 1 in the i^{th} coordinate and zero everywhere else) represents the distribution X_ℓ of the last step in an ℓ-step random walk starting from the i^{th} vertex.

Definition 21.2 *(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 $\|\mathbf{v}\|_2 = 1$.

Relation to eigenvalues. The value $\lambda(G)$ is 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 $\lambda_1, \ldots, \lambda_n$ (see Section A.5.3) 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 21.5) 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 define the parameter $\lambda(G)$ using the standard (un-normalized) adjacency matrix (rather than the random-walk matrix), in which case $\lambda(G)$ is a number between 0 and d and the spectral gap is defined to be $d - \lambda(G)$. Knowledge of basic facts on eigenvalues and eigenvectors (all covered in the appendix) can serve as useful background for this chapter, but is not strictly necessary to follow the results and proofs.

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

Lemma 21.3 *Let G ba an n-vertex regular graph and $\mathbf{p}$ a probability distribution over G's vertices, then*

$$\|A^\ell \mathbf{p} - \mathbf{1}\|_2 \leq \lambda^\ell \qquad \diamondsuit$$

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 subspace $\mathbf{1}^\perp$ to itself. Note that the eigenvectors that are different from $\mathbf{1}$ span this subspace, and A shrinks each of these eigenvectors by at least λ factor in ℓ_2 norm. Hence A must shrink every vector in $\mathbf{1}^\perp$ by at least λ. Thus A^ℓ shrinks every vector in $\mathbf{1}^\perp$ by a factor at least λ^ℓ and we conclude $\lambda(A^\ell) \leq \lambda(A)^\ell$. (In fact, using the eigenvalue definition of λ, it can be shown that $\lambda(A^\ell) = \lambda(A)^\ell$.)

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,

$$A^\ell \mathbf{p} = A^\ell(\mathbf{1} + \mathbf{p}') = \mathbf{1} + A^\ell \mathbf{p}'$$

[1] A matrix A is *symmetric* if $A_{i,j} = A_{j,i}$ for every i, j. That is, $A = A^\dagger$ where $A^\dagger$ denotes the *transpose* of A (see Section A.5).

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, $\|\mathbf{p}\|_2 \leq |\mathbf{p}|_1 = 1$ (see Claim 21.1). Hence $\|\mathbf{p}'\|_2 \leq 1$ and

$$\|A^\ell \mathbf{p} - \mathbf{1}\|_2 = \|A^\ell \mathbf{p}'\|_2 \leq \lambda^\ell \tag{1}$$

■

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

Lemma 21.4 *If G is a regular connected graph with self-loops at each vertex, then $\lambda(G) \leq 1 - \frac{1}{12n^2}$.* ♢

PROOF: Let $\epsilon = \frac{1}{6n^2}$, let $\mathbf{u} \perp \mathbf{1}$ be a unit vector and let $\mathbf{v} = A\mathbf{u}$. We need to prove that $\|\mathbf{v}\|_2 \leq 1 - \epsilon/2$ and for this it suffices to prove that $1 - \|\mathbf{v}\|_2^2 \geq \epsilon$. (Indeed, if $\|\mathbf{v}\|_2 > 1 - \epsilon/2$ then $\|\mathbf{v}\|_2^2 > 1 - \epsilon$ and hence $1 - \|\mathbf{v}\|_2^2 < \epsilon$.) Since $\mathbf{u}$ is a unit vector, $1 - \|\mathbf{v}\|_2^2 = \|\mathbf{u}\|_2^2 - \|\mathbf{v}\|_2^2$. 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,

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

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

Thus it suffices to show $\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \epsilon$. Since $\mathbf{u}$ is a unit vector with coordinates summing to zero, there must exist vertices i, j such that $\mathbf{u}_i > 0, \mathbf{u}_j < 0$ and at least one of these coordinates has absolute value $\geq \frac{1}{\sqrt{n}}$, meaning that $\mathbf{u}_i - \mathbf{u}_j \geq \frac{1}{\sqrt{n}}$. Furthermore, because G is connected there is a path between i and j containing at most $D + 1$ vertices (where D is the *diameter* of the graph[2] G). By renaming vertices, let's assume that $i = 1$, $j = D + 1$ and the coordinates $2, 3, \ldots, D$ correspond to the vertices on this path in order. Now, we have

$$\tfrac{1}{\sqrt{n}} \leq \mathbf{u}_1 - \mathbf{u}_{D+1} = (\mathbf{u}_1 - \mathbf{v}_1) + (\mathbf{v}_1 - \mathbf{u}_2) + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1}) \leq$$
$$\mathbf{u}_1 - \mathbf{u}_{D+1} = |\mathbf{u}_1 - \mathbf{v}_1| + |\mathbf{v}_1 - \mathbf{u}_2| + \ldots + |\mathbf{v}_D - \mathbf{u}_{D+1}| \leq$$
$$\sqrt{(\mathbf{u}_1 - \mathbf{v}_1)^2 + (\mathbf{v}_1 - \mathbf{u}_2)^2 + \ldots + (\mathbf{v}_D - \mathbf{u}_{D+1})^2}\sqrt{2D+1}\,, \tag{2}$$

where the last inequality follows by relating the L_2 and L_1 norms of the vector $(\mathbf{u}_1 - \mathbf{v}_1, \mathbf{v}_1 - \mathbf{u}_2, \ldots, \mathbf{v}_D - \mathbf{u}_{D+1})$ using Claim 21.1. But this means that

$$\sum_{i,j} A_{i,j}(\mathbf{u}_i - \mathbf{v}_j)^2 \geq \tfrac{1}{dn(2D+1)}\,, \tag{3}$$

since the left hand side of (3) is a sum of non-negative terms and (2) implies that the terms of the form $A_{i,i}(\mathbf{u}_i - \mathbf{v}_i)^2$ and $A_{i,i+1}(\mathbf{v}_i - \mathbf{u}_{i+1})^2$ (for $i = 1, \ldots, D$) contribute at least $\frac{1}{dn(2D+1)}$ to this sum (both $A_{i,i}$ and $A_{i,i+1}$ are at least $1/d$ since they correspond to self-loops and edges of the graph).

Plugging in the trivial bound $D \leq n - 1$ this already shows that $\lambda(G) \leq 1 - \frac{1}{4dn^2}$. To prove the lemma as stated, we use the fact (left as Exercise 21.4) that for every d-regular connected graph, $D \leq 3n/(d+1)$. ■

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 random-walk matrix of a bipartite graph then one can find a vector $\mathbf{v}$ such that $A\mathbf{v} = -\mathbf{v}$ (Exercise 21.3).

[2]The *diameter* of a graph G is the maximum distance (i.e., length of shortest path) between any pair of vertices in G. Note that the diameter of a connected n-vertex graph is always at most $n - 1$.

21.1.2 Analysis of the randomized algorithm for undirected connectivity.

Together, lemmas 21.3 and 21.4 imply that, at least for regular graphs, if s is connected to t then a sufficiently long random walk from s will hit t in polynomial time with high probability:

Corollary 21.5 *Let G be a d-regular n-vertex graph with all vertices having a self-loop. Let s be a vertex in G. Let $\ell > 24n^2 \log n$ and let X_ℓ denote the distribution of the vertex of the ℓ^{th} step in a random walk from s. Then, for every t connected to s, $\Pr[X_\ell = t] > \frac{1}{2n}$.*$\diamondsuit$

PROOF: By Lemmas 21.3 and 21.4 , 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 $\ell \geq 13n^2$, $\|A^\ell \mathbf{p} - \mathbf{1}\|_2 < (1 - \frac{1}{12n^2})^{24n^2 \log n} < \frac{1}{n^2}$, where $\mathbf{1}$ here is the uniform distribution over this component. But this means that in particular for every coordinate i, $|A^\ell \mathbf{p} - \mathbf{1}|_i < \frac{1}{n^2}$ and hence every element in the connected component appears in $A^\ell \mathbf{p}$ with probability at least $1/n - 1/n^2 \geq 1/(2n)$. ■

Note that Corollary 21.5 implies that if we repeat the $24n^2 \log n$ walk for $O(n \log n)$ times (or equivalently, if we take a walk of, say, length $100n^3 \log^2 n$) then we will hit *every* vertex t connected to s with high probability.

21.2 Expander graphs.

Expander graphs are extremely useful combinatorial objects, which we encounter several times in the book. They can be defined in two equivalent ways. At a high level, these two equivalent definitions can be described as follows:

- *Combinatorial definition:* A constant-degree regular graph G is an *expander* if for every subset S of less than half of G's vertices, a constant fraction of the edges touching S are from S to its complement in G; see Figure 21.1.
- *Algebraic expansion:* A constant-degree regular graph G is an *expander* if its parameter $\lambda(G)$ bounded away from 1 by some constant. That is, $\lambda(G) \leq 1 - \epsilon$ for some constant $\epsilon > 0$.

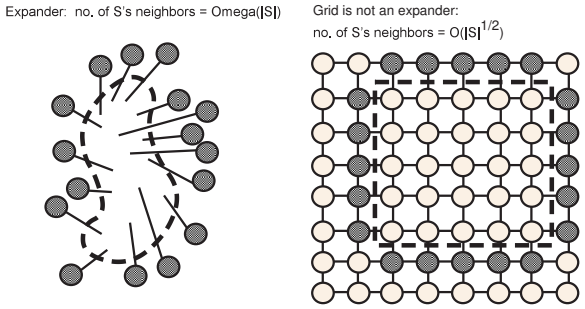


Figure 21.1 In an *edge expander*, every subset S of the vertices that is not too big has at least $\Omega(|S|)$ edges to neighbors outside the set. The grid (and every other planar graph) is not an edge expander as a $k \times k$ square in the grid has only $O(k)$ neighbors outside it.

What do we mean by a constant? By *constant* we refer to a number that is independent of the size of the graph. We will typically talk about graphs that are part of an infinite *family* of graphs, and so by constant we mean a value that is the same for all graphs in the family, regardless of their size. Below we make the definitions more precise, and show their equivalence.

21.2.1 The Algebraic Definition

The Algebraic definition of expanders is as follows:

Definition 21.6 *((n, d, λ)-expander 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.
A family of graphs $\{G_n\}_{n\in\mathbb{N}}$ is an *expander graph family* if there are some constants $d \in \mathbb{N}$ and $\lambda < 1$ such that for every n, G_n is an (n, d, λ)-graph.

Many texts use simply the name (n, d, λ)-graphs for such graphs. Also, as mentioned above, some texts use *un-normalized* adjacency matrices, and so have λ range between 0 and d. The smallest λ can be for a d-regular n-vertex graph is $(1-o(1))\frac{2\sqrt{d-1}}{d}$ where $o(1)$ denotes a function tending to 0 as the number of vertices grows. This is called the Alon-Boppana bound and graphs meeting this bound are called *Ramanujan graphs* (see also Exercises 21.9 and 21.10).

Explicit constructions. As we will see in Section 21.2.2, it is not hard to show that expander families exist using the probabilistic method. But this does not yield *explicit* constructions of such graphs which are often needed for applications. We say that an expander family $\{G_n\}_{n\in\mathbb{N}}$ is *explicit* if there is a polynomial-time algorithm that on input 1^n outputs the adjacency matrix of G_n (or, equivalently, the random-walk matrix). We say that the family is *strongly explicit* if there is a polynomial-time algorithm that on inputs $\langle n, v, i\rangle$ where $v \in [n], i \in [d]$ outputs the (index of the) i^{th} neighbor of v. Note that in the strongly explicit case, the lengths of the algorithm's inputs and outputs are $O(\log n)$ and so it runs in time polylog(n).

Fortunately, several explicit and strongly explicit constructions of expander graphs are known. Some of these constructions are very simple and efficient, but their analysis is highly non-trivial and uses relatively deep mathematics.[3] In Section 21.3 we will see a strongly explicit construction of expanders with elementary analysis. This construction also introduces a tool that we'll use to derandomize the random-walk algorithm for UPATH in Section 21.4.

21.2.2 Combinatorial expansion and existence of expanders.

We now describe a combinatorial criterion that is roughly equivalent to Definition 21.6. One advantage of this criterion 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.

Definition 21.7 *(Combinatorial (edge) expansion)*
An n-vertex d-regular graph $G = (V, E)$ is called an (n, d, ρ)-*combinatorial edge expander* if for every subset S of vertices satisfying $|S| \leq n/2$,

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

where $\overline{S}$ denotes the complement of S and for subsets S, T of vertices, $E(S, T)$ denotes the set of edges $\overline{i\,j}$ with $i \in S$ and $j \in T$.

[3] An example is the following 3-regular expander graph: the vertices are the numbers 0 to $p-1$ for some prime p, and each number x is connected to $x+1$,$x-1$ and x^{-1} modulo p (letting $0^{-1} = 0$). The analysis uses some deep results in mathematics (i.e., Selberg's 3/16 Theorem), see Section 11.1.2 in [HLW06].

Note that in this case the bigger ρ is the better the expander. We will often use the shorthand "edge expander" (dropping the prefix "combinatorial"). Also we'll loosely use the term "expander" for any (n, d, ρ)-combinatorial edge expander with ρ a positive constant (independent of n). Using the probabilistic method, one can prove the following theorem: (Exercise 21.11 asks you to prove a slightly weaker version)

Theorem 21.8 *(Existence of expanders) Let $\epsilon > 0$ be any constant. Then there exists $d = d(\epsilon)$ and $N \in \mathbb{N}$ such that for every $n > N$ there exists an $(n, d, 1/2 - \epsilon)$ edge expander.* ♢

Theorem 21.8 is tight in the sense that there is no (n, d, ρ) edge expander for $\rho > 1/2$ (Exercise 21.13). The following theorem relates combinatorial expansion with our previous Definition 21.6

Theorem 21.9 *(Combinatorial vs. algebraic expansion)*

1. *If G is an (n, d, λ)-expander graph then it is an $(n, d, (1-\lambda)/2)$-edge expander.*
2. *If G is an (n, d, ρ) edge expander then its second largest eigenvalue (*without *taking absolute values) 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 $\epsilon = \min\left\{\frac{2}{d}, \frac{\rho^2}{2}\right\}$.*

The condition that G has all the self-loops of Theorem 21.9 is used again to rule out bipartite graphs, which can be very good edge expanders but have one eigenvalue equal to -1 and hence a spectral gap of zero.

21.2.3 Algebraic expansion implies combinatorial expansion.

The first part of Theorem 21.9 follows immediately from the following lemma:

Lemma 21.10 *Let G be an (n, d, λ) graph, S a subset of G's vertices and T its complement. Then*

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

♢

Proof: Let $\mathbf{x} \in \mathbb{R}^n$ denote the following vector:

$$\mathbf{x}_i = \begin{cases} +|T| & i \in S \\ -|S| & i \in T \\ 0 & \text{otherwise} \end{cases}\,.$$

Note that $\|\mathbf{x}\|_2^2 = |S||T|^2 + |T||S|^2 = |S||T|(|S|+|T|)$ and $\mathbf{x} \perp \mathbf{1}$.

Let $Z = \sum_{i,j} A_{i,j}(x_i - x_j)^2$. On the one hand $Z = \frac{2}{d}|E(S,T)|(|S|+|T|)^2$, since every edge $\overline{i\,j}$ with $i \in S$ and $j \in T$ appears twice in this sum, each time contributing $\frac{1}{d}(|S|+|T|)^2$ to the total. On the other hand,

$$Z = \sum_{i,j} A_{i,j}x_i^2 - 2\sum A_{i,j}x_i x_j + \sum_{i,j} A_{i,j}x_j^2 = 2\|\mathbf{x}\|_2^2 - 2\langle \mathbf{x}, A\mathbf{x}\rangle$$

(using the fact that A's rows and columns sum up to one). Since $\mathbf{x} \perp \mathbf{1}$ and $\|A\mathbf{x}\|_2 \leq \lambda\|\mathbf{x}\|_2$, we get that

$$\tfrac{1}{d}|E(S,T)|(|S|+|T|)^2 \geq (1-\lambda)\|x\|_2^2\,.$$

Plugging in $\|x\|_2^2 = |S||T|(|S|+|T|)$ completes the proof. ■

Algebraic expansion also allows us to obtain an estimate on the number of edges between not-too-small subsets S and T, even if they are not disjoint:

Lemma 21.11 *(Expander Mixing Lemma) Let $G = (V, E)$ be an (n, d, λ)-expander graph. Let $S, T \subseteq V$, then*

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

♢

The Mixing Lemma gives a good idea of why expanders are "pseudorandom." In a random d-regular graph, we would expect $|E(S,T)|$ to be about $\frac{d}{n}|S||T|$. The Lemma says that in an expander, $|E(S,T)|$ is close to this expectation for *all* S, T that are sufficiently large. We leave the proof of Lemma 21.11 as Exercise 21.14.

21.2.4 Combinatorial Expansion Implies Algebraic Expansion

We now prove the second part of Theorem 21.9. 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 random-walk matrix.

Let λ be the second largest eigenvalue of A (not taking absolute values). We need to prove that $\lambda \leq 1 - \rho^2/2$. By the definition of an eigenvalue 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. (Since $\mathbf{u} \perp \mathbf{1}$, both $\mathbf{v}$ and $\mathbf{w}$ are nonzero.) We can assume that $\mathbf{v}$ is nonzero on at most $n/2$ of its coordinates (otherwise take $-\mathbf{u}$ instead of $\mathbf{u}$). Let $Z = \sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2)$. Part 2 of the theorem (except for the "furthermore" clause) follows immediately from the following two claims:

Claim 1: $Z \geq 2\rho\|\mathbf{v}\|_2^2$.

Claim 2: $Z \leq \sqrt{8(1-\lambda)}\|\mathbf{v}\|_2^2$.

Proof of Claim 1: 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, using $\mathbf{v}_i^2 - \mathbf{v}_j^2 = \sum_{k=i}^{j+1}(\mathbf{v}_k^2 - \mathbf{v}_{k+1}^2)$,

$$Z = \sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2) = 2\sum_{i<j} A_{i,j} \sum_{k=i}^{j-1}(\mathbf{v}_k^2 - \mathbf{v}_{k+1}^2)\,.$$

Note that every term $(\mathbf{v}_k^2 - \mathbf{v}_{k+1}^2)$ appears in this sum once (with a weight of $2/d$) per each edge $\overline{i\,j}$ such that $i \leq k < j$. Since $\mathbf{v}_k = 0$ for $k > n/2$, this means that

$$Z = \tfrac{2}{d}\sum_{k=1}^{n/2} |E(\{1..k\}, \{k+1..n\})|(\mathbf{v}_k^2 - \mathbf{v}_{k+1}^2) \geq \tfrac{2}{d}\sum_{k=1}^{n/2} \rho k(\mathbf{v}_k^2 - \mathbf{v}_{k+1}^2)\,,$$

by G's expansion. But, rearranging the terms (and using the fact that $\mathbf{v}_k = 0$ for $k > n/2$), the last sum is equal to

$$\tfrac{2}{d}d\rho\sum_{k=1}^{n/2} k\mathbf{v}_k^2 - (k-1)\mathbf{v}_k^2 = 2\sum_{k=1}^{n}\mathbf{v}_k^2 = 2\rho\|\mathbf{v}\|_2^2\,.$$

■

Proof of Claim 2: 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 not positive, $\langle A\mathbf{v}, \mathbf{v}\rangle / \|\mathbf{v}\|_2^2 \geq \lambda$, meaning that

$$1 - \lambda \geq 1 - \frac{\langle A\mathbf{v}, \mathbf{v}\rangle}{\|\mathbf{v}\|_2^2} = \frac{\|\mathbf{v}\|_2^2 - \langle A\mathbf{v}, \mathbf{v}\rangle}{\|\mathbf{v}\|_2^2} = \frac{\sum_{i,j} A_{i,j}(\mathbf{v}_i - \mathbf{v}_j)^2}{2\|\mathbf{v}\|_2^2}\,, \tag{4}$$

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 of the last term in (4) by $\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 + \mathbf{v}_j^2)$. The new numerator is

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

using the Cauchy-Schwartz inequality.[4] Hence, using $(a-b)(a+b) = a^2 - b^2$,

$$1-\lambda \geq \frac{\left(\sum_{i,j} A_{i,j}(\mathbf{v}_i^2 - \mathbf{v}_j^2)\right)^2}{2\|\mathbf{v}\|_2^2 \sum_{i,j} A_{i,j}(\mathbf{v}_i + \mathbf{v}_j)^2} = \frac{Z^2}{2\|\mathbf{v}\|_2^2\left(\sum_{i,j} A_{i,j}\mathbf{v}_i^2 + 2\sum_{i,j} A_{i,j}\mathbf{v}_i\mathbf{v}_j + \sum_{i,j} A_{i,j}\mathbf{v}_j^2\right)} =$$
$$\frac{Z^2}{2\|\mathbf{v}\|_2^2\left(2\|\mathbf{v}\|_2^2 + 2\langle A\mathbf{v}, \mathbf{v}\rangle\right)} \geq \frac{Z^2}{8\|\mathbf{v}\|_2^4},$$

where the last inequality is due to the fact that A is a symmetric stochastic matrix, and hence $\|A\mathbf{v}\|_2 \leq \|\mathbf{v}\|_2$ for every $\mathbf{v}$, implying that $\langle A\mathbf{v}, \mathbf{v}\rangle \leq \|\mathbf{v}\|_2^2$.

The "furthermore" part is obtained by noticing that adding all the self-loops to a $d-1$-regular graph is equivalent to transforming its random-walk matrix A into the matrix $\frac{d-1}{d}A + \frac{1}{d}I$ where I is the identity matrix. Since A's smallest eigenvalue (not taking absolute values) is at least -1, the new smallest eigenvalue is at least $-\frac{d-1}{d} + \frac{1}{d} = -1 + \frac{2}{d}$.■

21.2.5 Error reduction using expanders.

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 it seems hard to think of a way to save on randomness. Nonetheless, 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 strongly explicit family that is an $(M = 2^m, d, 1/10)$-expander graph for some constant d. (Note that we can use graph powering to transform any explicit expander family into an expander family with parameter $\lambda < 1/10$; see also Section 21.3.) Choose a vertex v_1 at random, and take a 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 $v_1, \ldots, v_k$ for the random coins (we identify the set $[M]$ of vertices with the set $\{0,1\}^m$ of possible random coins for the algorithm) and output the majority answer.

To keep things simple, 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 1/2 (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 $\mathcal{B} \subseteq [M]$ denote the "bad" set of coins on which the algorithms rejects. We know that $|\mathcal{B}| \leq \frac{M}{3}$. Plugging in $\beta = 1/3$ and $\lambda = 1/10$ in the following theorem immediately implies that the probability the above procedure will reject an input in the language is bounded by $2^{-\Omega(k)}$:

[4] The Cauchy-Schwartz inequality says that for every two vectors $\mathbf{x}, \mathbf{y}$, $\langle \mathbf{x}, \mathbf{y}\rangle \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2$. Here we index over (i,j), and use $\mathbf{x}_{i,j} = \sqrt{A_{i,j}}(\mathbf{v}_i^2 - \mathbf{v}_j^2)$ and $\mathbf{y}_{i,j} = \sqrt{A_{i,j}}(\mathbf{v}_i^2 + \mathbf{v}_j^2)$.

Theorem 21.12 *(Expander walks)*
Let G be an (n, d, λ) graph, and let $\mathcal{B} \subseteq [n]$ satisfying $|\mathcal{B}| <= \beta n$ for some $\beta \in (0, 1)$. Let $X_1, \ldots, X_k$ be random variables denoting a $k-1$-step random walk in G from X_1, where X_1 is chosen uniformly in $[n]$. Then,

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

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 $X_i \in \mathcal{B}$. Note that the probability we're trying to bound is

$$\Pr[\wedge_{i=1}^{k} B_i] = \Pr[B_1] \Pr[B_2|B_1] \cdots \Pr[B_k|B_1, \ldots, B_{k-1}] . \quad (5)$$

Denote by B the linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^n$ that "zeroes out" the coordinates that are not in $\mathcal{B}$. That is, for every $i \in [n]$, $(B\mathbf{u})_i = \mathbf{u}_i$ if $i \in \mathcal{B}$ and $(B\mathbf{u})_i = 0$ otherwise. It's not hard to verify that for every probability vector $\mathbf{p}$ over $[n]$, $B\mathbf{p}$ is a vector whose coordinates sum up to the probability that a vertex i is chosen according to $\mathbf{p}$ is in $\mathcal{B}$. Furthermore, if we normalize the vector $B\mathbf{p}$ to sum up to one, we get the probability vector corresponding to the conditional distribution of p conditioned on the event that the vertex chosen this way is in $\mathcal{B}$.

Thus, if we let $\mathbf{1} = (1/n, \ldots, 1/n)$ denote the uniform distribution over $[n]$ and $\mathbf{p}^i \in \mathbb{R}^N$ be the distribution of X_i conditioned on the events $B_1, \ldots, B_i$, then

$$\mathbf{p}^1 = \frac{1}{\Pr[B_1]} B\mathbf{1}$$
$$\mathbf{p}^2 = \frac{1}{\Pr[B_2|B_1]} \frac{1}{\Pr[B_1]} BAB\mathbf{1}$$

and more generally

$$\mathbf{p}^i = \frac{1}{\Pr[B_i|B_{i-1} \ldots B_1] \cdots \Pr[B_1]} (BA)^{i-1} B\mathbf{1} .$$

Since every probability vector $\mathbf{p}$ satisfies $|\mathbf{p}|_1 = 1$, it follows that the probability on the LHS of (5) is equal to

$$|(\hat{B}A)^{k-1} \hat{B}\mathbf{1}|_1 . \quad (6)$$

Using the relation between the L_1 and L_2 norms (Claim 21.1) we can bound (6) by showing

$$\|(\hat{B}A)^{k-1} B\mathbf{1}\|_2 \leq \frac{((1-\lambda)\sqrt{\beta}+\lambda)^{k-1}}{\sqrt{n}} . \quad (7)$$

To prove (7), we will use the following definition and Lemma:

Definition 21.13 *(Spectral norm)* For every matrix A, the *spectral norm* of A, denoted by $\|A\|$, is defined as the maximum of $\|A\mathbf{v}\|_2$ over all vectors $\mathbf{v}$ satisfying $\|\mathbf{v}\|_2 = 1$. ♢

Exercises 21.5 and 21.6 ask you to prove that the spectral norm of every random-walk matrix is 1, and that for every two n by n matrices A, B, $\|A + B\| \leq \|A\| + \|B\|$ and $\|AB\| \leq \|A\|\|B\|$.

Lemma 21.14 *Let A be a random-walk matrix of an (n, d, λ)-expander graph G. Let J be the random-walk matrix of the n-clique with self loops (i.e., $J_{i,j} = 1/n$ for every i, j). Then*

$$A = (1-\lambda)J + \lambda C \quad (8)$$

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)-expander graph as going to the uniform distribution with probability $1-\lambda$, and to a different distribution with probability λ. This is of course completely inaacurate, 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 (8) will be just as good.[5]

PROOF OF LEMMA 21.14: 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 $\mathbf{u} = \alpha\mathbf{1}$ for some $\alpha \in \mathbb{R}$ and $\mathbf{w} \perp \mathbf{1}$. 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 $\mathbf{w}' = A\mathbf{w}$. Then $\|\mathbf{w}'\|_2 \leq \lambda\|\mathbf{w}\|_2$ and, as we saw in the proof of Lemma 21.3, $\mathbf{w}' \perp \mathbf{1}$. In other words, the sum of the coordinates of $\mathbf{w}$ is zero, meaning that $J\mathbf{w} = \mathbf{0}$. We get that $C\mathbf{w} = \frac{1}{\lambda}\mathbf{w}'$. Since $\mathbf{w}' \perp \mathbf{u}$, $\|C\mathbf{v}\|_2^2 = \|\mathbf{u} + \frac{1}{\lambda}\mathbf{w}'\|_2^2 = \|\mathbf{u}\|_2^2 + \|\frac{1}{\lambda}\mathbf{w}'\|_2^2 \leq \|\mathbf{u}\|_2^2 + \|\mathbf{w}\|_2^2 = \|\mathbf{v}\|_2^2$, where we use twice the Pythagorean theorem that for $\mathbf{u} \perp \mathbf{w}$, $\|\mathbf{u}+\mathbf{w}\|_2^2 = \|\mathbf{u}\|_2^2 + \|\mathbf{w}\|_2^2$. ■

Returning to the proof of Theorem 21.12, we can write $BA = B\big((1-\lambda)J + \lambda C\big)$, and hence $\|BA\| \leq (1-\lambda)\|BJ\| + \lambda\|BC\|$. Since J's output is always a vector of the form $\alpha\mathbf{1}$, and it can be easily verified that $\|B\mathbf{1}\|_2 = \sqrt{\frac{\beta n}{n^2}} = \frac{\sqrt{\beta}}{\sqrt{n}} = \sqrt{\beta}\|\mathbf{1}\|_2$, $\|BJ\| = \sqrt{\beta}$. Also, because B is an operation that merely zeros out some parts of its input, $\|B\| \leq 1$ implying that $\|BC\| \leq 1$. Thus, $\|BA\| \leq (1-\lambda)\sqrt{\beta} + \lambda$. This means that $\|(BA)^{k-1}B\mathbf{1}\|_2 \leq ((1-\lambda)\sqrt{\beta} + \lambda)^{k-1}\frac{\sqrt{\beta}}{\sqrt{n}}$, establishing (7). ■

The success of the error reduction procedure for *two-sided error* algorithms is obtained by the following theorem, whose proof we omit (but see Exercise 21.12):

Theorem 21.15 *(Expander Chernoff Bound)*
Let G be an (n, d, λ)-expander 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 $\delta > 0$,

$$\Pr\left[\left|\frac{\sum_{i=1}^{k} B_i}{k} - \beta\right| > \delta\right] < 2e^{(1-\lambda)\delta^2 k/4}$$

21.3 Explicit construction of expander graphs

We now show a construction of a very explicit expander graph family. The main tools in our construction will be several types of graph products. 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' and 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 random-walk matrix (denoted λ), and study how different products affect these parameters. We then use these products to obtain a construction of a strongly explicit expander graph family. In the next section we will use the same products to show a *deterministic* logspace algorithm for undirected connectivity.

21.3.1 Rotation maps.

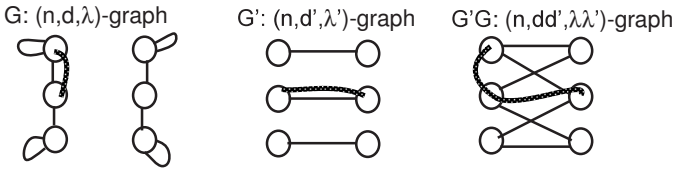
Thus far we usually represented a graph via its adjacency matrix, or as in this chapter, via its random-walk matrix. If the graph is d-regular we can also represent it via its *rotation*

[5] Algebraically, the reason (8) 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.

map. If G is an n-vertex degree-d graph this involves giving a number from 1 to d to each neighbor of each vertex, and then letting a rotation map $\hat{G}$ be a function from $[n] \times [d]$ to $[n] \times [d]$ that maps a pair $\langle v, i \rangle$ to $\langle u, j \rangle$ where u is the ith neighbor of v and v is the jth neighbor of u. Clearly, this map is a permutation (i.e., is one-to-one and onto) of $[n] \times [d]$. The reader may wonder why one should not renumber the neighbors at each node so 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). This is indeed possible but it requires some global computation that will turn out to be too complicated in the scenarios we will be interested in, where the graph is constructed by some space-bounded computation.

Below we will describe graph products, which is usually a way to map two graphs into one. We use whichever graph representation happens to be most natural, but it would be a good exercise for the reader to to work out the equivalent descriptions in the other representations (e.g., in terms of random-walk matrices and rotation maps).

21.3.2 The matrix/path product

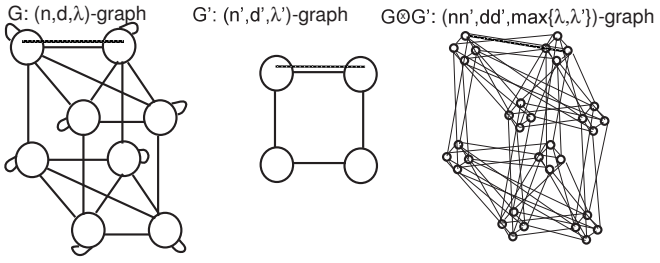


For every two n-vertex graphs G, G' with degrees d, d' and random-walk matrices A, A', the graph $G'G$ is the graph described by the random-walk matrix $A'A$. That is, $G'G$ has an edge (u, v) for every length 2-path from u to v where the first step in the path is taken on an edge of G and the second 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 have already encountered this case before in Lemma 21.3, and similar analysis yields the following lemma (whose proof we leave as Exercise 21.8):

Lemma 21.16 *(Matrix product improves expansion)* $\lambda(G'G) \leq \lambda(G')\lambda(G')$ ♢

Note that one can easily compute the rotation map of $G'G$ using the rotation maps of G and G'.

21.3.3 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 $\widehat{G \otimes G'}$ is the permutation over $([n] \times [n']) \times ([d] \times [d'])$ defined as

follows

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

where $(u', i') = \hat{G}(u, i)$ and $(v', j') = \hat{G}'(v, j)$. That is, the vertex set of $G \otimes G'$ consists of 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 random-walk matrices, the tensor product is also quite easy to describe. If $A = (a_{i,j})$ is the $n \times n$ random-walk matrix of G and $A' = (a'_{i',j'})$ is the $n' \times n'$ random-walk matrix of G', then the random-walk 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 $a_{i,j}$:

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

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.

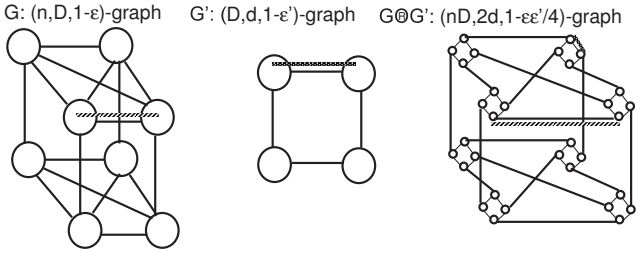
Lemma 21.17 *(Tensor product preserves expansion) Let $\lambda = \lambda(G)$ and $\lambda' = \lambda(G')$ then $\lambda(G \otimes G') \leq \max\{\lambda, \lambda'\}$.* ♢

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 of Lemma 21.17: This is immediate from some basic facts about tensor products and eigenvalues (see Exercise 21.22). If $\lambda_1, \dots, \lambda_n$ are the eigenvalues of A (where A is the random-walk matrix of G) and $\lambda'_1, \dots, \lambda'_{n'}$ are the eigenvalues of A' (where A' is the random-walk 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 $\lambda(G) \cdot 1$ ■

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 21.23). Even this weaker bound suffices for our applications.

21.3.4 The replacement product



In both the matrix 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 $G \textcircled{r} G'$ is the nn'-vertex $2d$-degree graph obtained as follows:

1. For every vertex u of G, the graph $G \textcircled{r} G'$ 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.)

Some texts use the term "replacement product" for the variant of this product that uses only a single edge (as opposed to d parallel edges) in Item 2 above. The addition of parallel edges ensures that a random step from a vertex v in $G \textcircled{r} G'$ will move with probability $1/2$ to a neighbor within the same cluster and with probability $1/2$ to a neighbor outside the cluster.

The replacement product also has a simple description in terms of rotation maps: since $G \textcircled{r} G'$ has nD vertices and $2d$ degree, its rotation map $G \hat{\textcircled{r}} G'$ can be thought of as a permutation over $([n] \times [D]) \times ([d] \times \{0,1\})$ that takes 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 random-walk matrices the replacement product is described as follows:

$$A \textcircled{r} A' = 1/2\hat{A} + 1/2(I_n \otimes A'), \tag{9}$$

where A, A' denote the random-walk matrices of G and G' respectively, and $\hat{A}$ denotes the permutation matrix corresponding to the rotation map of G. That is, $\hat{A}$ is an $(nD) \times (nD)$ matrix whose $(i,j)^{th}$ column is all zeroes except a single 1 in the $(i',j')^{th}$ place where $(i',j') = \hat{G}(i,j)$.

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:

Lemma 21.18 *(Expansion of replacement product) If $\lambda(G) \leq 1 - \epsilon$ and $\lambda(H) \leq 1 - \delta$ then $\lambda(G \textcircled{r} H) \leq 1 - \frac{\epsilon\delta^2}{24}$.* ♢

The intuition behind Lemma 21.18 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 21.2.5, 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 $G \textcircled{r} G'$ is motivated by this intuition: a random walk on $G \textcircled{r} G'$ 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 $G \textcircled{r} G'$ 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 21.24, that analyzes the *combinatorial* (edge) expansion of the resulting graph as a function of the edge expansion of the input graphs.

Proof of Lemma 21.18: It suffices to show that $\lambda(G \textcircled{r} H)^3 \leq 1 - \frac{\epsilon\delta^2}{8}$. Since for every graph F, $\lambda(F^k) = \lambda(F)^k$, we will do so by showing $\lambda((G \textcircled{r} H)^3) \leq 1 - \frac{\epsilon\delta^2}{8}$. Let A be the $n \times n$ random-walk matrix of G (with $\hat{A}$ the $(nD) \times (nD)$ permutation matrix corresponding to the rotation map $\hat{G}$), let B be the $D \times D$ random-walk matrix of H, and let C be the random-walk matrix of $(G \textcircled{r} H)^3$. Then, (9) implies that

$$C = (1/2\hat{A} + 1/2(I_n \otimes B))^3, \tag{10}$$

Now Lemma 21.14 implies that $B = (1-\delta)B' + \delta J_D$ for some matrix B' with norm at most 1 (where J_D is the $D \times D$ all $1/D$ matrix). We plug this into (10), expand all terms and then collect together all the terms except for the one corresponding to $1/2\delta(I_n \otimes J)1/2\hat{A}1/2\delta(I_n \otimes J)$. The reader can verify that all terms correspond to matrices of norm at most 1 and hence (10) becomes

$$C = (1 - \tfrac{\delta^2}{8})C' + \tfrac{\delta^2}{8}(I_n \otimes J_D)\hat{A}(I_n \otimes J_D)\,, \tag{11}$$

where C' is some $(nD) \times (nD)$ matrix of norm at most 1. The lemma will follow from the following claim:

CLAIM: $(I_n \otimes J_D)\hat{A}(I_n \otimes J_D) = A \otimes J_D$

PROOF: Indeed, the left-hand side is the random-walk matrix of the graph on nD vertices on which a step from a vertex (i, j) corresponds to: 1) choosing a random $k \in [D]$ 2) letting i' be the k^{th} neighbor of i in G 3) choosing j' at random in $[D]$ moving to the vertex (i, k). We can equivalently describe this as going to a random neighbor i' of i in G and choosing j' at random in $[D]$, which is the graph corresponding to the matrix $A \otimes J_D$. ■

The claim concludes the proof since $\lambda(A \otimes J_D) \leq \max\{\lambda(A), \lambda(J_D)\} = \max\{\lambda(A), 0\}$. The lemma follows by plugging this into (11) and using the fact that $\lambda(C') \leq 1$ for every matrix of norm at most 1. ■

21.3.5 The actual construction.

We now use the three graph products of described above to show a strongly explicit construction of an expander graph family. That is, we prove the following theorem:

Theorem 21.19 *(Explicit construction of expanders)*
There exists a strongly explicit (λ, d)-expander family for some constants $d \in \mathbb{N}$ and $\lambda < 1$.

Note that using the matrix/graph product, Theorem 21.19 can be improved to yield a strongly explicit (λ, d)-expander family for *every* $\lambda > 0$ (albeit at the expense of allowing d to be an arbitrarily large constant depending on λ.

PROOF: We will start by showing something slightly weaker: a very explicit family of graphs $\{G_k\}$ where G_k is not a graph on k vertices but on roughly c^k vertices for some constant c. That is, rather than showing a family of graphs for *every* size n, we will only show a family of graphs that contains a graph of size n for every n that is a power of c. We will then sketch how the construction can be improved to yield a graph family containing a graph of every size n.

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 products 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. The actual definition is as follows:

- Let H be a $(D = (2d)^{100}, d, 0.01)$-expander graph, which we can find using brute force search. (We choose d to be a large enough constant that such a graph exists) We let G_1 be a $((2d)^{100}, 2d, 1/2)$-expander graph and G_2 be a $((2d)^{200}, 2d, 1/2)$-expander graphs (again, such graphs can be easily found via brute force).

- For $k > 2$ define
$$G_k = \left(G_{\left\lfloor \frac{k-1}{2} \right\rfloor} \otimes G_{\left\lfloor \frac{k-1}{2} \right\rfloor} \right).$$

We prove the following claim:

CLAIM: For every k, G_k is a $((2d)^{100k}, 2d, 1 - 1/50)$-expander graph. Furthermore, there is a poly(k)-time algorithm that given a label of a vertex i in G_k and an index j in $[2d]$ finds the j^{th} neighbor of i in G_k.

PROOF: We prove the first part by induction. Verify directly that it holds for $k = 1, 2$. For $k > 2$, if we let n_k be the number of vertices of G_k then $n_k = n^2_{\lfloor (k-1)/2 \rfloor}(2d)^{100}$. By induction we assume $n_{\lfloor (k-1)/2 \rfloor} = (2d)^{100\lfloor (k-1)/2 \rfloor}$ which implies that $n_k = (2d)^{100k}$ (using the fact that $2\lfloor (k-1)/2 \rfloor + 1 = k$). It's also easy to verify that G_k has degree $2d$ for every j: if G has degree $2d$ then $G \otimes G$ has degree $(2d)^2$, $(G \otimes G)^{50})$ has degree $(2d)^{100}$ and $(G \otimes G)^{50}) \circledR H$ has degree $(2d)$. The eigenvalue analysis also follows by induction: if $\lambda(G) \leq 1 - 1/50$ then $\lambda(G \otimes G)^{50} \leq 1/e < 1/2$. Hence, by Lemma 21.18, $\lambda((G \otimes G)^{50} \circledR H) \leq 1 - 1/2(0.99)^2/24 \leq 1 - 1/50$.

For the furthermore part, note that there is a natural algorithm to compute the neighborhood function of G_k that makes 100 recursive calls to the neighborhood function of $G_{\lfloor (k-1)/2 \rfloor}$, thus running in time roughly $n^{\log 100}$. ■

The above construction and analysis yields an expander graph family containing an n vertex graph for every n of the form c^k for some constant c. The proof of Theorem 21.19 is completed by observing that one can transform an (n, d, λ)-expander graph to an (n', cd, λ')-expander graph (where $\lambda' < 1$ is a constant depending on λ, d) for any $n/c \leq n' \leq n$ by joining together into a "mega-vertex" sets of at most c vertices (Exercise 21.16). ■

Remark 21.20
The quantitative bounds obtained from the proof of Theorem 21.19 are pretty bad, both in terms of the relation between degree and expansion and the running time (in particular the initial brute force search alone will take more than 2^{100} steps). This is partly because for pedagogical reasons we chose to present this construction in its simplest form, without covering various known optimizations. However, even with these optimizations this construction is not the most efficient known.

There are different known constructions of expanders that are highly practical and efficient (e.g., [LPS86, Mar88]). However, their analysis typically uses deep facts in number theory. Also, the replacement product (and its close cousin, the zig-zag product) have found applications beyond the proof of Theorem 21.15. One such application is the deterministic logspace algorithm for undirected connectivity described in the next section. Another application is a construction of combinatorial vertex expanders with a greater expansion of small sets that what is implied by the parameter λ ([CRVW02], see also Exercise 21.15).

21.4 Deterministic logspace algorithm for undirected connectivity.

This section describes a recent result of Reingold, showing that at least the most famous randomized logspace algorithm, the random walk algorithm for the problem UPATH of s-t-connectivity in undirected graphs (see Chapter 7) can be completely "derandomized."

Theorem 21.21 *(Reingold's Theorem)*
UPATH $\in \mathbf{L}$.

Reingold describes a set of poly(n) walks starting from s such that if s is connected to t then one of the walks is guaranteed to hit t. The *existence* of such a small set of walks

can be shown using the probabilistic method and Corollary 21.5. The point here is that Reingold's enumeration of walks can be carried out deterministically in logspace.

Proof outline. As before we are interested in undirected graphs that may have parallel edges. We restrict our attention to checking connectivity for d-regular graphs for say $d = 4$. This is without loss of generality: if a vertex has degree $d'' < 3$ we add a self-loop of multiplicity to bring the degree up to d, and if the vertex has degree $d' \geq 3$ we can replace it by a cycle of d' vertices, and each of the d' edges that were incident to the old vertex then attach to one of the cycle nodes. Of course, a logspace machine does not have space to store the modified graph, but it can pretend that these modifications have taken place, since it can perform them on the fly whenever it accesses the graph. (To put this more formally, the transformation is implicitly computable in logspace as per Definition 4.16.) In fact, the proof below will perform a series of other local modifications on the graph, each with the property that the logspace algorithm can perform them on the fly.

We start by observing that checking connectivity in *expander* graphs is easy. Specifically, 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 with a path of length at most ℓ. Indeed, Lemma 21.3 implies that in every n-vertex regular graph G, the distribution of the ℓ^{th} vertex in a random walk is within $\sqrt{n}\lambda^\ell$ statistical (or L_1) distance from the uniform distribution. In particular this means that if each connected component H of G is an *expander* graph, having $\lambda(H)$ bounded away from 1, then a random walk of length $\ell = O(\log n)$ from a vertex u in H will reach every vertex of H with positive probability.

The idea behind Reingold's algorithm is 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.

21.4.1 The logspace algorithm for connectivity (proof of Theorem 21.21)

By adding more self-loops we may assume that the input graph G is of degree d^{50} for some constant d that is sufficiently large to ensure the existence of a $(d^{50}, d/2, 0.01)$-expander graph H. Since the size of H is constant, we can store all of it in memory using $O(1)$ bits.[6] Let $G_0 = G$ and for $k \geq 1$, define $G_k = (G_{k-1} \textcircled{r} H)^{50}$, where $\textcircled{r}$ denotes the balanced replacement product defined in Section 21.3.4.

If G_{k-1} is an N-vertex graph with degree d^{50}, then $G_{k-1} \textcircled{r} H$ is a $d^{50}N$-vertex graph with degree d and thus $G_k = (G_{k-1} \textcircled{r} H)^{50}$ is a $d^{50}N$-vertex graph with degree d. Note also that if two vertices were connected (resp., disconnected) in G_{k-1}, then they are still connected (resp., disconnected) in G_k. The key observation that the graph $G_{10 \log n}$ is an expander, and therefore an easy instance of UPATH. Specifically, we have:

Claim: For every k, G_k is an $(d^{50k}n, d^{20}, \max\{1-1/20, 2^k/(12n^2)\})$-graph, where n denotes the number of vertices in $G = G_0$.

Proof: Indeed, by Lemmas 21.16 and 21.18, for every $\epsilon < 1/20$ and D-degree graph F, if $\lambda(F) \leq 1-\epsilon$ then $\lambda(F \textcircled{r} H) \leq 1 - \epsilon/25$ and hence $\lambda\left((F \textcircled{r} H)^{50}\right) \leq 1 - 2\epsilon$. By Lemma 21.4, every connected component of G has expansion parameter at most $1 - \frac{1}{12n^2}$ (note that n is at least as large as the number of vertices in the connect component). It follows that for $k = 10 \log n$, in the graph G_k every connected component has expansion parameter at most $\max\{1 - 1/20, 2^k/(12n^2)\} = 1 - 1/20$. ■

Since $G_{10 \log n}$ is an expander, to find whether a pair of vertices s, t are connected in $G_{10 \log n}$ we simply need to enumerate over all paths in $G_{10 \log n}$ that start at s and have length $\ell = O(\log n)$, and see whether any one of these hits t. The catch is of course that the graph provided to our algorithm is G, not $G_{10 \log n}$. A simpler question is whether, given G, our algorithm can perform even a *single* step of a random walk on G_k for $k = 10 \log n$.

[6] We can either use an explicit construction of such a graph or simply find it using an exhaustive search among all graphs of this size.

Specifically, given a description of a vertex s in G_k and an index $i \in [d^{20}]$, it has to compute the i^{th} neighbor of s in G_k using only logarithmic space. It is easy to see that if we can perform this single step in logarithmic space, then we can just as easily perform ℓ steps as well by repeating the single step again and again while keeping a counter, and reusing the same space to compute each step.

The graph G_k is equal to $(G_{k-1} \circledR H)^{50}$ and thus it suffices to show that we can take a single step in the graph $G_{k-1} \circledR H$ in logspace (we can then repeat the same process for 50 times). Now by the definition of the replacement product, a vertex in $G_{k-1} \circledR H$ is represented by a pair $\langle u, v \rangle$ where u is a vertex of G_{k-1} and v is a vertex of H. The index of a neighbor of $\langle u, v \rangle$ is represented by a pair $\langle b, i \rangle$ where $b \in \{0,1\}$ and $i \in [d/2]$. If $b = 0$ then the $\langle b, i \rangle^{th}$ neighbor of $\langle u, v \rangle$ is $\langle u, v' \rangle$ where v' is the i^{th} neighbor of v' in H. If $b = 1$ then the $\langle b, i \rangle^{th}$ neighbor of $\langle u, v \rangle$ is the pair $\langle u', v' \rangle$ denoting the result of applying G_{k-1}'s rotation map to $\langle u, v \rangle$. (That is, u' is the v^{th} neighbor of u in G_{k-1}, and v' is the index of u as a neighbor of u' in G_{k-1}.) This description already implies an obvious recursive algorithm to compute the rotation map of G_k. Letting s_k denotes the space needed to compute a rotation map of G_k by this algorithm, we see that s_k satisfies the equation $s_k = s_{k-1} + O(1)$, implying that $s_{10 \log n} = O(\log n)$.[7] ■

21.5 Weak Random Sources and Extractors

Suppose, that despite any philosophical difficulties, we are happy with probabilistic algorithms, and see no need to "derandomize" them, especially at the expense of some unproven assumptions. We still need to tackle the fact that real world sources of randomness and unpredictability rarely, if ever, behave as a sequence of perfectly uncorrelated and unbiased coin tosses. Can we still execute probabilistic algorithms using real-world "weakly random" sources?

21.5.1 Min Entropy

For starters, we try to define what we could mean by a weakly random source. Historically speaking, several definitions were proposed, which are recalled in Example 21.23. The following definition (due to D. Zuckerman) encompasses all previous definitions.

Definition 21.22 Let X be a random variable. The *min entropy* of X, denoted by $H_\infty(X)$, is the largest real number k such that $\Pr[X = x] \leq 2^{-k}$ for every x in the range of X.

If X is a distribution over $\{0,1\}^n$ with $H_\infty(X) \geq k$ then it is called an (n,k)-*source*. ♢

It is not hard to see that if X is a random variable over $\{0,1\}^n$ then $H_\infty(X) \leq n$ with $H_\infty(X) = n$ if and only if X is distributed according to the uniform distribution U_n. Our goal in this section is to be able to execute probabilistic algorithms given access to a distribution X with $H_\infty(X)$ as small as possible. It can be shown that min entropy is a *minimal requirement* in the sense that a general simulation of a probabilistic algorithm that uses k random bits requires access to a distribution X that is (close to) having min entropy at least k (see Exercise 21.18).

Example 21.23

We will now see that min entropy is a pretty general notion, and can allow us to model many other models of "imperfectly random" sources. Here are some examples for distributions X over $\{0,1\}^n$.

[7] When implementing the algorithm one needs to take care *not* to make a copy of the input when invoking the recursive procedure, but rather have all procedure operate on a globally accessible memory that contains the index k and the vertex and edge labels; otherwise we'd get an $O(\log n \log\log n)$-space algorithm. For more details see the original paper [Rei05] or [Gol08, Section 5.2.4].

- (von Neumann's model: biased coins) X is composed of n independent coin tosses, each outputting 1 with probability $\delta < 1/2$ and 0 with probability $1-\delta$. It is easily checked that[8] $H_\infty(X) = \log(1/(1-\delta))n$.
- (Santha-Vazirani sources) X has the property that for every $i \in [n]$, and every string $x \in \{0,1\}^{i-1}$, conditioned on $X_1 = x_1, \ldots, X_{i-1} = x_{i-1}$ it holds that both $\Pr[X_i = 0]$ and $\Pr[X_i = 1]$ are between δ and $1-\delta$. This generalizes von Neumann's model and can model sources such as stock market fluctuations, where current measurements do have some limited dependence on the previous history. It is easily checked that $H_\infty(X) \geq \log(1/(1-\delta))n$.
- (Bit fixing and generalized bit fixing sources) In a *bit-fixing* source, there is a subset $S \subseteq [n]$ with $|S| = k$ such that X's bits in the coordinates given by S are uniformly distributed over $\{0,1\}^k$, and X's bits in the coordinates given by $[n] \setminus S$ is a fixed string (say the all-zeros string). Then $H_\infty(X) = k$. The same holds if X's projection to $[n] \setminus S$ is a fixed deterministic function of its projection to S, in which case we say that X is a *generalized bit-fixing source*. For example, if the bits in the odd positions of X are independent and uniform and for every even position $2i$, $X_{2i} = X_{2i-1}$ then $H_\infty(X) = \lceil \frac{n}{2} \rceil$. This may model a scenario where we measure some real world data at too high a rate (think of measuring every second a physical event that changes only every minute).
- (Linear subspaces) If X is the uniform distribution over a linear subspace of $\mathrm{GF}(2)^n$ of dimension k, then $H_\infty(X) = k$. (In this case X is actually a generalized bit-fixing source — can you see why?)
- (Uniform over subset) If X is the uniform distribution over a set $S \subseteq \{0,1\}^n$ with $|S| = 2^k$ then $H_\infty(X) = k$. As we will see, this is a very general case that "essentially captures" all distributions X with $H_\infty(X) = k$.

21.5.2 Statistical distance

Next we formalize what it means to *extract* random —more precisely, almost random— bits from an (n,k) source. We will use the notion of *statistical distance* (see Section A.2.6 in the appendix) to qualify when two distributions are close to one another. Recall that if X and Y are two distributions over some domain Ω then the *statistical distance* between X and Y, denoted by $\Delta(X,Y)$ is equal to

$$\max_{f:\Omega\to\{0,1\}} |\mathsf{E}[f(X)] - \mathsf{E}[f(Y)]| \ . \tag{12}$$

It is also known that $\Delta(X,Y) = 1/2|\mathbf{x} - \mathbf{y}|_1$, where $\mathbf{x}$ and $\mathbf{y}$ are the vectors in $\mathbb{R}^\Omega$ that represent the distributions X and Y respectively. For any $\epsilon > 0$, we say that two distribution X and Y are ϵ-*close*denoted $X \approx_\epsilon Y$, if $\Delta(X,Y) \leq \epsilon$.

21.5.3 Definition of randomness extractors

We can now define randomness extractors - these are functions that transform an (n,k) source into an almost uniform distribution. The extractor uses a small number of additional truly random bits, called a *seed* and denoted by d in the definition below.

[8] In fact, as n grows X is close to a distribution with min-entropy $H(\delta)n$ where H is the Shannon entropy function defined as $H(\delta) = \delta \log \frac{1}{\delta} + (1-\delta)\log\frac{1}{1-\delta}$. The same holds for Santha-Vazirani sources defined below. See [DFR+07] for this and more general results of this form.

Definition 21.24 *(Randomness extractors)*
A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ϵ) extractor if for any (n,k)-source X, the distribution $\mathsf{Ext}(X, U_d)$ is ϵ-close to U_m. (For every ℓ, U_ℓ denotes the uniform distribution over $\{0,1\}^\ell$.)

Why an additional input? Our stated motivation for extractors is to execute probabilistic algorithms without access to perfect unbiased coins. Yet, it seems that an extractor is not sufficient for this task, as we only guarantee that its output is close to uniform if it is given an an additional *seed* that is uniformly distributed. We have two answers to this objection. First, note that the requirement of an additional input is necessary: for every function $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ and every $k \leq n-1$ there exists an (n,k)-source X such that the first bit of $\mathsf{Ext}(X)$ is constant (i.e, is equal to some value $b \in \{0,1\}$ with probability 1), and so is at least of statistical distance $1/2$ from the uniform distribution (Exercise 21.17). Second, if the length t of the second input is sufficiently short (e.g., $t = O(\log n)$) then, for the purposes of simulating probabilistic algorithms, we can do without any access to true random coins, by enumerating over all the 2^t possible inputs. Clearly, d has to be somewhat short for the extractor to be non-trivial. The completely trivial case is when $d \geq m$, in which case the extractor can simply ignore its first input and output the seed!

21.5.4 Existence proof for extractors.

It turns out that at least if we ignore issues of computational efficiency, very good extractors exist:

Theorem 21.25 *For every $k, n \in \mathbb{N}$ and $\epsilon > 0$, there exists a (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^k$ with $d = \log n + 2\log(1/\epsilon) + O(1)$* ♢

PROOF: Call an (n,k) source X *flat* if X is the uniform distribution over a 2^k-sized subset of $\{0,1\}^n$. In Exercise 19.7 it is shown that every (n,k) source can be expressed as a convex combination of flat (n,k)-sources. Because the statistical distance of a convex combination of distributions $Y_1, \ldots, Y_N$ from a distribution U is at most the maximum of $\Delta(Y_i, X)$ (Exercise 21.19), it suffices to show a function Ext such that $\mathsf{Ext}(X, U_d)$ is close to the uniform distribution when X is an (n,k)-flat source.

We will prove the existence of such an extractor by the probabilistic method, choosing Ext as a random function from $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^k$. Let X be an (n,k) flat source and let f be a function from $\{0,1\}^k \to \{0,1\}$. If we choose Ext at random then the expectation $\mathsf{E}[f(\mathsf{Ext}(X, U_d))]$ is obtained by evaluating f on $2^k \times 2^d$ random points, and hence by the Chernoff bound the probability that this expectation deviates from $\mathsf{E}[f(U_k)]$ by more than ϵ is bounded by $2^{-2^{k+d}/4\epsilon^2}$. This means that if $d > \log n + 2\log(1/e) + 3$ then this probability is bounded by $2^{-2n(2^k)}$. But the number of flat distributions is at most $(2^n)^{2^k}$ and the number of functions from $\{0,1\}^k \to \{0,1\}$ is 2^{2^k} and hence the union bound implies that there is a choice of Ext guaranteeing

$$|\mathsf{E}[f(\mathsf{Ext}(X, U_d))] - \mathsf{E}[f(U_k)]| < \epsilon$$

for *every* (n,k)-flat source and function $f : \{0,1\}^k \to \{0,1\}$. In other words, $\mathsf{Ext}(X, U_d)$ is ϵ-close to U_k for every (n,k)-flat source and hence for every (n,k)-source. ■

This extractor is optimal in the sense that there is an absolute constant c such that every (k,ϵ) extractor that is non-trivial (has output longer than seed length and $\epsilon < 1/2$) must satisfy $d \geq \log(n-k) + 2\log(1/\epsilon) - c$ [NZ93, RTS97].

21.5.5 Extractors based on hash functions

The non-explicit extractor of Theorem 21.25 is not very useful: for most applications we need *explicit* extractors— namely extractors computable in polynomial time. One such explicit extractor (though with a long seed length) can be obtained using pairwise independent hash functions.

Recall (Section 8.2.2) that a collection $\mathcal{H}$ of functions from $\{0,1\}^n$ to $\{0,1\}^m$ is *pairwise independent* if for every $x \neq x'$ in $\{0,1\}^n$ and $y, y' \in \{0,1\}^m$, the probability that $h(x) = y$ and $h(x') = y'$ for a random $h \in_R \mathcal{H}$ is 2^{-2m}. There are known construction of such collections where each function h can be described by a string of length $n+m$ (we abuse notation and call this string also h). Choosing a random function from the collection is done by choosing a random string in $\{0,1\}^{n+m}$. The next famous lemma shows that with an appropriate setting of parameters, the map $x, h \mapsto h(x) \circ h$ (where $\circ$ denotes concatenation) is an extractor. This is not a superb extractor in terms of parameter values but it is useful in many settings.

Lemma 21.26 *(Leftover hash lemma* [BBR88, ILL89]*)* *Let $m = k - 2\log(1/\epsilon)$, then for every (n,k) source X,*

$$\Delta(H(X) \circ H, U_n \circ H) < \epsilon\,,$$

where H denotes a randomly chosen (description of) function in a pairwise independent hash function collection from $\{0,1\}^n$ to $\{0,1\}^m$. ♢

PROOF: We study the *collision probability* of $H(X) \circ H$, where we identify H with U_ℓ where $\ell = n + m$ is the length of description of the hash function. That is, the probability that $h(x) \circ h = h'(x') \circ h'$ for random $h, h' \in_R \mathcal{H}$ and $x, x' \in_R X$. This is bounded by the probability that $h = h'$ (which is equal to $2^{-\ell}$) times the probability that $h(x) = h(x')$. The latter is bounded by 2^{-k} (a bound on the probability that $x = x'$ implies by the fact that X is an (n,k)-source) plus 2^{-m} (the probability that $h(x) = h(x')$ for a random $h \in_R \mathcal{H}$ and $x \neq x'$). Thus the collision probability of $(H(X), H)$ is at most $2^{-\ell}(2^{-k} + 2^{-m}) = 2^{-(\ell+m)} + 2^{-\ell-k}$.

Now, treat this distribution as a probability vector $\mathbf{p} \in \mathbb{R}^{2^{\ell+m}}$. Then the collision probability is precisely the L_2-norm of $\mathbf{p}$ squared. We can write $\mathbf{p} = \mathbf{1} + \mathbf{w}$ where $\mathbf{1}$ is the probability vector corresponding to the distribution $U_n \circ H = U_{n+\ell}$ and $\mathbf{w}$ is orthogonal to $\mathbf{1}$. (For a general vector $\mathbf{p}$ we'd only be able to write $\mathbf{p} = \alpha\mathbf{1} + \mathbf{w}$ for some $\alpha \in \mathbb{R}$, but since $\mathbf{p}$ is a probability vector it must hold that $\alpha = 1$, as otherwise the entries of the righthand side will not sum up to one.) Thus by the Pythagorean Theorem $\|\mathbf{p}\|_2^2 = \|\mathbf{u}\|_2^2 + \|\mathbf{w}\|_2^2$, and since $\|\mathbf{u}\|_2^2 = 2^{-\ell-m}$ we get that

$$\|\mathbf{w}\|_2^2 = \|\mathbf{p} - \mathbf{1}\|_2^2 \leq 2^{-\ell-m}\,.$$

Using the relation between the L_1 and L_2 norms (Claim 21.1), we see that

$$\Delta(H(X) \circ H, U_{\ell+m}) = 1/2|\mathbf{p} - \mathbf{1}|_1 \leq 1/2 2^{(m+\ell)/2}\|\mathbf{v} - \mathbf{1}\|_2 \leq 2^{k/2+\ell/2-\log(1/\epsilon)} 2^{-k/2-\ell/2} < \epsilon\,.$$

■

21.5.6 Extractors based on random walks on expanders

We can also construct explicit extractors using expander graphs:

Lemma 21.27 *Let $\epsilon > 0$. For every n and $k \leq n$ there exists an explicit (k,ϵ)-extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^n$, where $t = O(n - k + \log 1/\epsilon)$.* ♢

PROOF: Suppose X is an (n,k)-source and we are given a sample a from it. Let G be a $(2^n, d, 1/2)$-expander graph for some constant d (see Definition 21.6 and Theorem 21.19).

Let z be a truly random seed of length $t = \log d(n/2 - k/2 + \log 1/\epsilon + 1) = O(n - k + \log 1/\epsilon)$. We interpret z as a random walk in G of length $\ell = n/2 - k/2 + \log 1/\epsilon + 1$ starting from the node whose label is a. (That is, we think of z as ℓ labels in $[d]$ specifying the steps taken in the walk.) The output $\mathsf{Ext}(a, z)$ of the extractor is the label of the final node on the walk.

Following the proof of Lemma 21.3 (see Equation (1)) we see that, letting $\mathbf{p}$ denote the probability vector corresponding to X and A the random-walk matrix of G,

$$\|A^\ell \mathbf{p} - \mathbf{1}\|_2 \leq 2^{-\ell} \|\mathbf{p} - \mathbf{1}\|_2 .$$

But since X is an (n, k) source, $\|\mathbf{p}\|_2^2$ (which is equal to the collision probability of X) is at most 2^{-k}, and hence in particular $\|\mathbf{p} - \mathbf{1}\|_2 \leq \|\mathbf{p}\|_2 + \|\mathbf{1}\|_2 \leq 2^{-k/2} + 2^{-n/2} \leq 2^{-k/2+1}$. Thus for our choice of ℓ,

$$\|A^\ell \mathbf{p} - \mathbf{1}\|_2 \leq 2^{-n/2+k/2-\log(1/\epsilon)+1} 2^{-k/2+1} \leq \epsilon 2^{-n/2} ,$$

which completes the proof using the relation between the L_1 and L_2 norms. ■

21.5.7 Extractors from pseudorandom generators

For many years explicit constructions of randomness extractors fell quite a bit behind the parameters achieved by the optimal non-explicit construction of Theorem 21.25. For example, we did not have explicit extractors that allowed us to run any randomized polynomial time algorithm using $\sim k$ bits using an (n, k) source where $k = n^\epsilon$ for arbitrarily small constant $\epsilon > 0$. (Generally, the smaller k is as a function of n, the harder the problem of constructing extractors; intuitively if $n \gg k$ then it's harder to "distill" the k bits of randomness that are hidden in the n-bit input.) To realize this goal, one should try to design an extractor that uses a seed of $O(\log n)$ bits to extract from an (n, n^ϵ)-source at least a polynomial number of bits (i.e., at least n^δ bits for some $\delta > 0$).[9] In 1999 Trevisan showed how to do this using an improved extractor construction. But more interesting than the result itself was Trevisan's idea: he showed that *pseudorandom generators* such as the ones we've seen in Chapters 20 and 19, when viewed in the right way, are in fact also randomness extractors. This was very surprising, since these pseudorandom generators rely on *hardness assumptions* (such as the existence of a function in **E** with high circuit complexity). Thus it would seem that these generators will not be useful in the context of randomness extractors, where we are looking for constructions with *unconditional* analysis and are not willing to make any unproven assumptions.

But thinking further, we realize that the above-mentioned difference between the two notions arises due to the type of "adversary" or "distinguisher" they have to work against. For a generator, the set of possible adversaries is the class of computationally limited algorithms (i.e., those that can be computed by circuits of some prescribed size). For an extractor, on the other hand, the set of adversaries is the set of all Boolean functions. The reason is that an extractor needs to produce a distribution $\mathcal{D}$ on $\{0,1\}^m$ whose statistical difference from U_m is at most ϵ, meaning that $|\Pr_{x \in \mathcal{D}}[D(x) = 1] - \Pr_{x \in U_m}[D(x) = 1]| \leq \epsilon$ for every function $D : \{0,1\}^m \to \{0,1\}$.

Trevisan noticed further that while we normally think of a pseudorandom generator G as having only one input, we can think of it as a function that takes two inputs: a short seed and the truth table of a candidate hard function f. While our theorems state that the pseudorandom generator works if f is a hard function, the proofs of these theorems are actually *constructive*: they transform a distinguisher D that distinguishes between the generator's output and a random string into a small circuit A that computes the function f. This circuit A uses the distinguisher D as a *black-box*. Therefore we can apply this transformation even when the distinguisher D is an arbitrary function that is not necessarily computable by a small circuit. This is the heart of Trevisan's argument.

[9] The work of Ta-Shma [TS96] did come close to this goal, achieving such an extractor with slightly super-logarithmic seed length.

Concretely, to make this all work we will need the stronger constructions of pseudorandom generators (e.g. of Theorem 20.7) that start with functions with high *worst-case* complexity. If there is a distinguisher D that distinguishes the output of such a generator from the uniform distribution, then the proof of correctness of the generator gives a way to compute the candidate hard function f on *every* input. Formally, we have the following theorem: (Below G^f refers to the algorithm G using f as a black box.)

Theorem 21.28 *(Constructive version of Theorem 20.7)*
For every time-constructible function $S : \mathbb{N} \to \mathbb{N}$ (the "security parameter"), there is a constant c and algorithms G and R satisfying the following:

- *On input a function $f : \{0,1\}^\ell \to \{0,1\}$ and a string $z \in \{0,1\}^{c\ell}$, algorithm G runs in $2^{O(\ell)}$ time and outputs a string $G^f(z)$ of length $m = S(\ell)^{1/c}$.*
- *If $D : \{0,1\}^m \to \{0,1\}$ is a function such that $\left|\mathsf{E}[D(G^f(U_{c\ell}))] - \mathsf{E}[D(U_m)]\right| > 1/10$ then there is an advice string a of length at most $S(\ell)^{1/4}$ such that on every input x, $R^D(a,x) = f(x)$ and furthermore R runs in time at most $S(\ell)^{1/4}$.*

The algorithm R mentioned in the theorem is just the reduction that is implicit in the proof of correctness of the pseudorandom generator in Chapter 20.

The following is Trevisan's extractor construction. Let G be as in Theorem 21.28. Let X be an (n,k)-source. Assume without loss of generality that n is a power of 2, and $n = 2^\ell$. Let $S(\ell)$, the "security parameter", stand for k. Given any string f from the source and the seed $z \in \{0,1\}^{c \log n}$, the extractor interprets f as a function from $\{0,1\}^\ell$ to $\{0,1\}$ and outputs

$$\mathsf{Ext}(f,z) = G^f(z)\,. \tag{13}$$

Thus given a string of length n and a seed of size $c \log n$, Ext produces $S(\ell)^{1/c} = k^{1/c}$ bits. Let us show that Ext is an extractor.

Claim 21.29 *For every k, n, the function Ext defined in (13) is a $(k, 1/5)$-extractor.* ♢

Proof: Suppose otherwise, that there is a (k,n)-source X and a Boolean function D that distinguishes between $\mathsf{Ext}(X, U_{c\ell})$ and U_m with bias at least $1/5$, where $m = S(\ell)^{1/c}$. Then, with probability at least $1/10$ over $f \in_R X$, function D distinguishes between $G^f(U_{c\ell})$ and U_m with bias at least $1/10$. Let's call an f for which this happens "bad". Note that for every bad f there exists an advice string $a \in \{0,1\}^{k^{1/4}}$ such that f is computed by the algorithm $x \mapsto R^D(a,x)$. Since R^D is a deterministic algorithm, this means that the number of bad f's is at most the number of choices for a, which is $2^{k^{1/4}}$. But since X is a k-source, it assigns probability not more than 2^{-k} to any particular string. Hence the probability of a random f being bad is at most $2^{k^{1/4}} 2^{-k} \ll 1/10$, and we've arrived at a contradiction to the assumption that D is a good distinguisher. ■

Remark 21.30
Readers mystified by this construction should try to look inside the generator G to get a better understanding. The extractor Ext turns out to do be very simple. Given a string $f \in \{0,1\}^n$ from the weak random source, the extractor first applies an error-correcting code (specifically, one that is *list decodable*) to f to get a string $\hat{f} \in \{0,1\}^{\mathrm{poly}(n)}$. Intuitively speaking, this has the effect of "smearing out"the randomness over the entire string. The extractor then selects a subset of the coordinates of $\hat{f}$ using the construction of the Nisan-Wigderson generator (see Section 20.2.2). That is, treating $\hat{f}$ as a Boolean function on $s = O(\log n)$ bits, we use a seed z of size $t = O(s)$ and output $\hat{f}(z_{I_1}) \circ \cdots \circ \hat{f}(z_{I_m})$, where $I_1, \ldots, I_m$ are s-sized subsets of $[t]$ that form a combinatorial design (see Definition 20.13).

21.6 Pseudorandom generators for space bounded computation

We now show how extractors can be used to obtain a pseudorandom generator for space-bounded randomized computation, which allows randomized logspace computations to be run with $O(\log^2 n)$ random bits. We stress that this generator does not use any unproven assumptions.

The goal here will be to derandomize randomized logspace computations, in other words, classes such as **BPL** and **RL**. Recall from Chapter 4 the notion of a *configuration graph* for a space-bounded TM. If we fix an input of size n for a logspace machine, then the configuration graph has size $\text{poly}(n)$. If the logspace machine is randomized, then it uses random coin tosses to make transitions within the configuration graph (i.e., each configuration has two outgoing edges, and each is taken with probability 1/2). To derandomize this computation we will replace the random string used by the logspace machine with the output of a "pseudorandom generator" (albeit one tailormade for fooling logspace computations) and show that the logspace machine cannot "tell the difference" (i.e., the probability it ends up in an accepting state at the end is not very different).

Theorem 21.31 *(Nisan's pseudorandom generator* [Nis90]*)*
For every d there is a $c > 0$ and a poly(n)-time computable function $g : \{0,1\}^{c \log^2 n} \to \{0,1\}^{n^d}$ (the "pseudorandom generator") such that for every space-bounded machine M that has a configuration graph of size $\leq n^d$ on inputs of size n:

$$\left| \Pr_{r \in \{0,1\}^{n^d}} [M(x,r) = 1] - \Pr_{z \in \{0,1\}^{c \log^2 n}} [M(x, g(z)) = 1] \right| < \frac{1}{10}. \quad (14)$$

By trying all possible choices for the $O(\log^2 n)$-bit input for the generator g in Nisan's theorem, we can simulate every algorithm in **BPL** in $O(\log^2 n)$ space. Note that Savitch's theorem (Theorem 4.14) also implies that $\mathbf{BPL} \subseteq \mathbf{SPACE}(\log^2 n)$ but it doesn't yield such a pseudorandom generator. In fact Theorem 21.31 can be strengthened to show that **BPL** can be decided using simultaneously polynomial time and space $O(\log^2 n)$, though we will not prove it here. Saks and Zhou [SZ95] improved Nisan's ideas to show that $\mathbf{BPL} \subseteq \mathbf{SPACE}(\log^{1.5} n)$, which leads many experts to conjecture that $\mathbf{BPL} = \mathbf{L}$ (i.e., randomness does not help logspace computations at all). Indeed, we've seen in Section 21.4 that the famous random-walk algorithm for undirected connectivity can be derandomized in logspace.

The main intuition behind Nisan's construction —and also the conjecture $\mathbf{BPL} = \mathbf{L}$— is that the logspace machine has one-way access to the random string and only $O(\log n)$ bits of memory. So it can only "remember" $O(\log n)$ of the random bits it has seen. To exploit this we will use the following simple lemma, which shows how to recycle a random string about which only a little information is known.

Lemma 21.32 *(Recycling lemma) Let $f : \{0,1\}^n \to \{0,1\}^s$ be any function and $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ be a $(k, \epsilon/2)$-extractor, where $k = n - (s+1) - \log \frac{1}{\epsilon}$. Then,*

$$\Delta\left(f(X) \circ U_m \,,\; f(X) \circ \mathsf{Ext}(X, U_t)\right) < \epsilon\,,$$

where X is a random variable distributed uniformly in $\{0,1\}^n$. ♢

To understand why we call it the Recycling Lemma, focus on the case $s \ll n$ and $n = m$. Suppose we use a random string X of length n to produce $f(X)$. Since $f(X)$ has length $s \ll n$, typically each string in $\{0,1\}^s$ will have many preimages under f. Thus anybody looking at $f(X)$ has only very little information about X. More formally, for every fixed choice of $f(X)$, the set of X that map to this value can be viewed as a weak random source. The Lemma says that applying an appropriate extractor (whose random seed z can have

length as small as $t = O(s + \log(1/\epsilon))$ if we use Lemma 21.27) on X we can get a new m-bit string $\mathsf{Ext}(X, z)$ that looks essentially random, even to somebody who knows $f(X)$.
PROOF: For $v \in \{0,1\}^s$ we denote by X_v the random variable that is uniformly distributed over the set $f^{-1}(v)$. Then we can express $\Delta(f(X) \circ W, f(X) \circ \mathsf{Ext}(X, z))$ as

$$= \frac{1}{2} \sum_{v,w} \left| \Pr[f(X) = v \wedge W = w] - \Pr_z[f(X) = v \wedge \mathsf{Ext}(X, z) = w] \right|$$

$$= \sum_v \Pr[f(X) = v] \cdot \Delta(W, \mathsf{Ext}(X_v, z)) \tag{15}$$

Let $V = \left\{ v : \Pr[f(X) = v] \geq \epsilon/2^{s+1} \right\}$. If $v \in V$, then we can view X_v as a (n, k)-source, where $k \geq n - (s+1) - \log \frac{1}{\epsilon}$. Thus by definition of an extractor, $\mathsf{Ext}(X_v, r) \approx_{\epsilon/2} W$ and hence the contributions from $v \in V$ sum to at most $\epsilon/2$. The contributions from $v \notin V$ are upperbounded by $\sum_{v \notin V} \Pr[f(X) = v] \leq 2^s \times \frac{\epsilon}{2^{s+1}} = \epsilon/2$. The lemma follows. ■

Now we describe how the Recycling Lemma is useful in Nisan's construction. Let M be a logspace machine. Fix an input of size n. Then for some $d \geq 1$ the graph of all configurations of M on this input has $\leq n^d$ configurations and runs in time $L \leq n^d$. Assume without loss of generality —since unneeded random bits can always be ignored— that M uses 1 random bit at each step. Assume also (by giving M a separate worktape that maintains a time counter), that the configuration graph is leveled: it has L levels, with level i containing configurations obtainable at time i. The first level contains only the start node and the last level contains two nodes, "accept" and "reject;" every other level has $W = n^d$ nodes. Each level i node has two outgoing edges to level $i + 1$ nodes and the machine's computation at this node involves using the next bit in the random string to pick one of these two outgoing edges. We sometimes call L the *length* of the configuration graph and W the *width*.

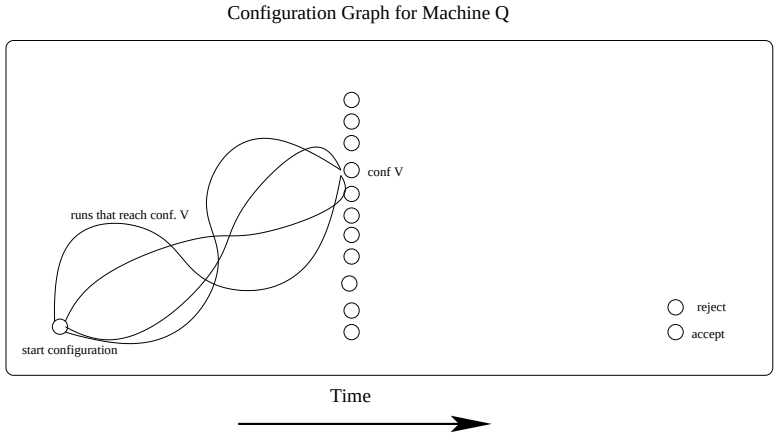


Figure 21.2 Configuration graph for machine M

For simplicity we first describe how to reduce the number of random bits by a factor 2. Think of the L steps of the computation as divided in two halves, each consuming $L/2$ random bits. Suppose we use some random string X of length $L/2$ to run the first half, and the machine is now at node v in the middle level. The only information known about X at this point is the index of v, which is a string of length $d \log n$. We may thus view the first half of the branching program as a (deterministic) function that maps $\{0,1\}^{L/2}$ bits to $\{0,1\}^{d \log n}$ bits. The Recycling Lemma allows us to use a random seed of length $O(\log n)$ to recycle X to get an almost-random string $\mathsf{Ext}(X, z)$ of length $L/2$, which can be used in the second half of the computation. Thus we can run L steps of computation using $L/2 + O(\log n)$ bits, a saving of almost a factor 2. Using a similar idea recursively, Nisan's generator runs L steps using $O(\log n \log L)$ random bits.

Now we formally define Nisan's generator.

Definition 21.33 *(Nisan's generator)* For some $r > 0$ let $\mathsf{Ext}_k : \{0,1\}^{kr} \times \{0,1\}^r \to \{0,1\}^{kr}$ be an extractor function for each $k \geq 0$. For every integer $k \geq 0$ the associated Nisan generator $G_k : \{0,1\}^{kr} \to \{0,1\}^{2^k}$ is defined recursively as (where $|a| = (k-1)r, |z| = r$)

$$G_k(a \circ z) = \begin{cases} z_1 \quad \text{(i.e., first bit of } z\text{)} & k = 1 \\ G_{k-1}(a) \circ G_{k-1}(\mathsf{Ext}_{k-1}(a,z)) & k > 1 \end{cases} \qquad \diamondsuit$$

Now we use this generator to prove Theorem 21.31. We only need to show that the probability that the machine goes from the start node to the "accept" node is similar for truly random strings and pseudorandom strings. However, we will prove a stronger statement involving intermediate steps as well.

If nodes u is a node in the configuration graph, and s is a string of length 2^k, then we denote by $f_{u,2^k}(s)$ the node that the machine reaches when started in u and its random string is s. Thus if s comes from some distribution $\mathcal{D}$, we can define a distribution $f_{u,2^k}(\mathcal{D})$ on nodes that are 2^k levels further from u.

Lemma 21.34 *Let $r = O(\log n)$ be such that for each $k \leq d \log n$, $\mathsf{Ext}_k : \{0,1\}^{kr} \times \{0,1\}^r \to \{0,1\}^{kr}$ is a $(kr - 2d\log n, \epsilon)$-extractor. For every machine of the type described in the previous paragraphs, and every node u in its configuration graph:*

$$\Delta(f_{u,2^k}(U_{2^k}), f_{u,2^k}(G_k(U_{kr}))) \leq 3^k \epsilon, \tag{16}$$

where U_l denotes the uniform distribution on $\{0,1\}^l$. $\diamondsuit$

To prove Theorem 21.31 from Lemma 21.34 let $u = u_0$, the start configuration, and $2^k = L$, the length of the entire computation. Choose $3^k\epsilon < 1/10$ (say), which means $\log 1/\epsilon = O(\log L) = O(\log n)$. Using the extractor of Section 21.5.6 as Ext_k, we can let $r = O(\log n)$ and so the seed length $kr = O(r \log L) = O(\log^2 n)$.

Proof of Lemma 21.34: Let ϵ_k denote the maximum value of the left hand side of (16) over all machines. The lemma is proved if we can show inductively that $\epsilon_k \leq 2\epsilon_{k-1} + 2\epsilon$. The case $k = 1$ is trivial. At the inductive step, we need to upper bound the distance between two distributions $f_{u,2^k}(\mathcal{D}_1)$, $f_{u,2^k}(\mathcal{D}_4)$, for which we introduce two distributions $\mathcal{D}_2, \mathcal{D}_3$ and use triangle inequality (which holds since $\Delta(\cdot,\cdot)$ is a distance function on distributions):

$$\Delta(f_{u,2^k}(\mathcal{D}_1), f_{u,2^k}(\mathcal{D}_4)) \leq \sum_{i=1}^{3} \Delta(f_{u,2^k}(\mathcal{D}_i), f_{u,2^k}(\mathcal{D}_{i+1}))\,. \tag{17}$$

The distributions will be:

$$\begin{aligned} \mathcal{D}_1 &= U_{2^k} \\ \mathcal{D}_4 &= G_k(U_{kr}) \\ \mathcal{D}_2 &= U_{2^{k-1}} \circ G_{k-1}(U_{(k-1)r}) \\ \mathcal{D}_3 &= G_{k-1}(U_{(k-1)r}) \circ G_{k-1}(U'_{(k-1)r}) \qquad (U, U' \text{ are identical but independent}). \end{aligned}$$

We bound the summands in (17) one by one.

Claim 1: $\Delta(f_{u,2^k}(\mathcal{D}_1) - f_{u,2^k}(\mathcal{D}_2)) \leq \epsilon_{k-1}$.
Denote $\Pr[f_{u,2^{k-1}}(U_{2^{k-1}}) = w]$ by $p_{u,w}$ and $\Pr[f_{u,2^{k-1}}(G_{k-1}(U_{(k-1)r})) = w]$ by $q_{u,w}$. According to the inductive assumption,

$$\frac{1}{2}\sum_w |p_{u,w} - q_{u,w}| = \Delta(f_{u,2^{k-1}}(U_{2^{k-1}}), f_{u,2^{k-1}}(G_{k-1}(U_{(k-1)r}))) \leq \epsilon_{k-1}.$$

Since $\mathcal{D}_1 = U_{2^k}$ may be viewed as two independent copies of $U_{2^{k-1}}$ we have

$$\Delta(f_{u,2^k}(\mathcal{D}_1), f_{u,2^k}(\mathcal{D}_2)) = \sum_v \frac{1}{2}\left|\sum_w p_{uw}p_{wv} - \sum_w p_{uw}q_{wv}\right|$$

where w, v denote nodes 2^{k-1} and 2^k levels respectively from u

$$= \sum_w p_{uw} \frac{1}{2} \sum_v |p_{wv} - q_{wv}|$$
$$\leq \epsilon_{k-1} \quad \text{(using inductive hypothesis and } \sum_w p_{uw} = 1)$$

Claim 2: $\Delta(f_{u,2^k}(\mathcal{D}_2), f_{u,2^k}(\mathcal{D}_3)) \leq \epsilon_{k-1}$.

The proof is similar to the previous case and is omitted.

Claim 3: $\Delta(f_{u,2^k}(\mathcal{D}_3), f_{u,2^k}(\mathcal{D}_4)) \leq 2\epsilon$.

We use the Recycling Lemma. Let $g_u : \{0,1\}^{(k-1)r} \to [1, W]$ be defined as $g_u(a) = f_{u,2^{k-1}}(G_{k-1}(a))$. (To put it in words, apply the Nisan generator to the seed a and use the result as a random string for the machine, using u as the start node. Output the node you reach after 2^{k-1} steps.) Let $X, Y \in U_{(k-1)r}$ and $z \in U_r$. According to the Recycling Lemma,

$$g_u(X) \circ Y \approx_\epsilon g_u(X) \circ \mathsf{Ext}_{k-1}(X, z),$$

and then Part 5 of Lemma A.21 implies that the equivalence continues to hold if we apply a (deterministic) function to the second string on both sides. Thus

$$g_u(X) \circ g_w(Y) \approx_\epsilon g_u(X) \circ g_w(\mathsf{Ext}_{k-1}(X, z))$$

for all nodes w that are 2^{k-1} levels after u. The left distribution corresponds to $f_{u,2^k}(\mathcal{D}_3)$ (by which we mean that $\Pr[f_{u,2^k}(\mathcal{D}_3) = v] = \sum_w \Pr[g_u(X) = w \wedge g_w(Y) = v]$) and the right one to $f_{u,2^k}(\mathcal{D}_4)$ and the proof is completed. ■

What have we learned?

- Often we can easily show that a random object has certain attractive properties, but it's non-trivial to come up with an *explicit* construction of an object with these properties. Yet, once found, such explicit constructions are often extremely useful.
- The behavior of random walks on a graph is tightly related to the eigenvalues of its adjacency matrix (or, equivalently, its normalized version— the random-walk matrix).
- An *expander* graph family is a collection of constant-degree graphs whose second largest eigenvalue is bounded away from 1. Such families can be shown to exist using the probabilistic method, but we also know of *explicit* constructions.
- An ℓ-step random walk on an expander graph is to a certain extend "pseudorandom" and behaves similarly to ℓ randomly chosen vertices under certain measures. This fact has been found useful in a variety of setting, from the randomness efficient error reduction procedure for **BPP** to the logspace algorithm for undirected connectivity.
- Extractors are functions that transform a distribution with a large min-entropy into (close to) the uniform distribution.
- Pseudorandom generators with a "black-box" analysis of their correctness can be used to construct randomness extractors, even though the latter are based on no unproven assumptions or lower bounds.

Chapter notes and history

Expanders were first defined by Bassalygo and Pinsker [BP73] and Pinsker [Pin73] proved their existence using the probabilistic method. They were motivated by the question of finding explicit

graphs to replace the random graphs in an error-correcting code construction by Gallager [Gal63]. Margulis [Mar73] gave the first explicit construction of an expander family although he did not give any bound on the parameter $\lambda(G)$ of graphs G in the family except to prove it is bounded away from 1. Gabber and Galil [GG79] improved Margulis's analysis and gave an explicit bound on $\lambda(G)$, a bound that was later improved by Jimbo and Marouka [JM85]. Lubotzky, Phillips and Sarnak [LPS86] and Margulis [Mar88] constructed *Ramanujan* graphs, that are expander with an optimal dependence between the parameter λ and their degree. The Alon-Boppanna lower bound on the second eigenvalue of a d-regular graph was first stated in [Alo86]; a tight bound on the $o(1)$ error term was given in [Nil04].

The relation between the algebraic (eigenvalue-based) and combinatorial definitions of expanders was developed by Dodziuk, Alon and Milman, and Alon in the papers [Dod84, AM84, AM85, Alo86]. Sinclair and Jerrum [SJ88] generalized this relation to the case of general reversible Markov chains. All of these results can be viewed as a discrete version of a result by Cheeger [Che70] on compact Riemannian manifolds.

Lemma 21.4 (every connected graph has some spectral gap) is from Alon and Sudakov [AS00a] and is an improved version of a result appearing as Problem 11.29 in Lovász's book [Lov07]. Lemma 21.11 (Expander Mixing Lemma) is from Alon and Chung [AC86] (though there it's stated with $T = V \setminus S$).

Karp, Pippenger and Sipser [KPS85] were the first to use expanders for derandomization, specifically showing how to use them to reduce the error of an **RP**-algorithm from 1/3 to $1/\sqrt{k}$ using only $O(k)$ additional random bits. Ajtai, Komlos, and Szemeredi [AKS87] were the first to use random walks on expander graphs for derandomization in their result that every **RL** algorithm using less $\log^2 n/\log\log n$ random bits can be simulated in deterministic log space. Cohen and Wigderson [CW89] and Impagliazzo-Zuckerman [IZ89] independently showed how to use the [AKS87] analysis to reduce the error of both **RP** and **BPP** algorithms as described in Section 21.2.5 (error reduction from 1/3 to 2^{-k} using $O(k)$ additional bits). An improved analysis of such walks was given by Gillman [Gil93] who proved the Expander Chernoff Bound (Theorem 21.15). Some additional improvements were given in [Kah97, WX05, Hea06].

The explicit construction of expanders presented in Section 21.3 is due to Reingold, Vadhan and Wigderson [RVW00], although our presentation follows [RV05, RTV06]. The expansion properties of the replacement product were also analyzed in a particular case of products of two cubes by Gromov [Gro83] and for general graphs (in a somewhat different context) by Martin and Randall [MR00].

Hoory, Linial and Wigderson [HLW06] give an excellent introduction to expander graphs and their computer science applications. The Alon-Spencer book [AS00b] also contains several results on expanders.

The problem of randomness extraction was first considered in the 1950s by von Neumann [vN51] who wanted to extract randomness from biased (but independent) random coins. This was generlized to Markov chains by Blum [Blu84]. Santha and Vazirani [SV84] studied extraction for the much more general class now known as "Santha Vazirani sources" (see Exercise 21.23), that necessitates adding a seed and allowing the output to have some small statistical distance from the uniform. Vazirani and Vazirani [VV85] showed how to simulate **RP** using a Santha-Vazirani source. Chor and Goldreich [CG85] improved the analysis of [SV84, VV85] and generalized further the class of sources. In particular they introduced the notion of *min-entropy*, and studied *block* sources, where each block has significant min-entropy even conditioned on the previous block. They also studied extraction from several (two or more) independent sources of high min-entropy (i.e., (k, n) sources for $k > n/2$). Zuckerman [Zuc90] put forward the goal of simulating probabilistic algorithms using a single source of high min-entropy and observed this generalizes all models that had been studied to date. (See [SZ94] for an account of various models considered by previous researchers.) Zuckerman also gave the first simulation of probabilistic algorithms from (k, n) sources assuming $k = \Omega(n)$. We note that extractors were also used implicitly in an early work of Sipser [Sip86] who showed certain conditional derandomization results under the assumption that certain (variants of) extractors exist (though he described them in a different way).

Extractors (albeit with long seed length) were also implicitly constructed and used in cryptography, using pairwise independent hash functions and the *leftover hash lemma* (Lemma 21.26) of Impagliazzo, Levin, and Luby [ILL89] and a related precursor by Bennett, Brassard and Robert [BBR88]. Nisan [Nis90] then showed that hashing (in particular the [VV85] generator) could be used to obtain provably good pseudorandom generators for logspace. Nisan and Zuckerman [NZ93] first defined extractors. They also gave a new extractor construction and used it to achieve their result that in general the amount of randomness used by a probabilistic algorithm can be reduced from polynomial to linear in the algorithm's space complexity. Since then a long sequence of beautiful works was dedicated to improving the parameters of extractors, on the way discovering

many important tools that were used in other areas of theoretical computer science. In particular, Guruswami et al [GUV07] (slightly improving over Lu et al [LRVW03]) constructed an extractor that has both seed length and output length within a constant factor of the optimal non-explicit extractor of Theorem 21.25. See [Sha02] for a good (though slightly outdated) survey on extractor constructions and their applications.

Trevisan's [Tre99] insight about using pseudorandom generators to construct extractors (see Section 21.5.7) has now been greatly extended. It is now understood that three combinatorial objects studied in three different fields are very similar: pseudorandom generators (cryptography and derandomization), extractors (weak random sources) and list-decodable error-correcting codes (coding theory and information theory). Constructions of any one of these objects often gives constructions of the other two. See the survey by Vadhan [Vad07].

Theorem 21.31 is by Nisan [Nis90], who also showed that all of **BPL** can be simulated using polynomial-time and $O(\log^2 n)$ space. The proof we presented is by Impagliazzo, Nisan, and Wigderson [INW94], with the extractor-based viewpoint due to Raz and Reingold [RR99]. Saks and Zhou [SZ95] extended Nisan's techniques to show an $O(\log^{1.5} n)$-space algorithm for every problem in **BPL**.

As perhaps the most important example of an **RL** problem, undirected connectivity has received special attention in the literature. Nisan, Szemeredi and Wigderson [NSW92] gave the first deterministic algorithm for undirected connectivity using $o(log^2 n)$ space, specifically $O(\log^{1.5} n)$; as mentioned above this result was later generalized to all of **RL** by [SZ95]. Armoni et al [ATSWZ97] improved the bound for undirected connectivity to $O(\log^{4/3} n)$ space. The deterministic space complexity of undirected connectivity was finally resolved by Reingold [Rei05] who showed that it lies in **L** (Theorem 21.21). Trifonov [Tri05] proved concurrently and independently the slightly weaker result of an $O(\log n \log\log n)$-space algorithm for this problem.

Exercises

21.1 Prove Claim 21.1 using the Cauchy-Schwartz Inquality— $|\langle \mathbf{u}, \mathbf{v}\rangle| \leq \|\mathbf{u}\|_2 \|\mathbf{v}\|_2$ for every two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$.

21.2 **(a)** Prove Hölder's Inequality (see Section A.5.4): For every p, q with $\frac{1}{p} + \frac{1}{q} = 1$, $\|\mathbf{u}\|_p \|\mathbf{v}\|_q \geq \sum_{i=1}^n |\mathbf{u}_i \mathbf{v}_i|$. Note that the Cauchy-Schwartz Inequality is the special case of Hölder's Inequality with $p = q = 2$. H458

(b) For a vector $\mathbf{v} \in \mathbb{R}^n$, define $\|\mathbf{v}\|_\infty = \max_{i\in[n]} |\mathbf{v}_i|$. Show that this is a norm and that for every $\mathbf{v}$,

$$\|\mathbf{v}\|_\infty = \lim_{p\to\infty} \left(\sum_{i=1}^n |\mathbf{v}_i|^p \right)^{1/p} .$$

(c) Prove that $\|\mathbf{v}\|_2 \leq \sqrt{|\mathbf{v}|_1 \|\mathbf{v}\|_\infty}$ for every vector $\mathbf{v} \in \mathbb{R}^n$. H458

21.3 Prove that if G is an n-vertex bipartite graph then there exists a vector $\mathbf{v} \in \mathbb{R}^n$ such that $A\mathbf{v} = -\mathbf{v}$ where A is the random-walk matrix of G.

21.4 Prove that for every n-vertex d-regular graph G, the diameter of G (maximum over all pairs of distinct vertices i, j in G of the length of the shortest path in G between i and j) is at most $3n/(d+1)$. H459

21.5 Recall that the *spectral norm* of a matrix A, denoted $\|A\|$, is defined as the maximum of $\|A\mathbf{v}\|_2$ for every unit vector $\mathbf{v}$. Let A be a symmetric stochastic matrix: i.e., $A = A^\dagger$ and every row and column of A has non-negative entries summing up to one. Prove that $\|A\| \leq 1$. H459

21.6 Let A, B be two $n \times n$ matrices.

(a) Prove that $\|A + B\| \leq \|A\| + \|B\|$.

(b) Prove that $\|AB\| \leq \|A\|\|B\|$.

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

21.8 Prove Lemma 21.16. H459

21.9 **(a)** Prove that if a probability distribution X has support of size at most d, its collision probability is at least $1/d$.

(b) Prove that if G is an (n, d, λ)-graph and X is the distribution over a random neighbor of the first vertex, then the collision probability of X is at most $\lambda^2 + 1/n$.

(c) Prove that $\lambda \geq \sqrt{\frac{1}{d} - \frac{1}{n}} = \frac{1}{\sqrt{d}} + o(1)$ (where $o(1)$ is a term that tends to 0 with n).

21.10 Recall that the *trace* of a Matrix A, denoted $\mathrm{tr}(A)$, is the sum of the entries along its diagonal.

(a) Prove that if an $n \times n$ matrix A has eigenvalues $\lambda_1, \ldots, \lambda_n$, then $\mathrm{tr}(A) = \sum_{i=1}^{n} \lambda_i$.

(b) Prove that if A is a random-walk matrix of an n-vertex graph G, and $k \geq 1$, then $\mathrm{tr}(A^k)$ is equal to n times the probability that a if we select a vertex i uniformly at random and take a k step random walk from i, then we end up back in i.

(c) Prove that for every d-regular graph G, $k \in \mathbb{N}$ and vertex i of G, the probability that a path of length k from i ends up back in i is at least as large as the corresponding probability in T_d, where T_d is the complete $(d-1)$-ary tree of depth k rooted at i. (That is, every internal vertex has degree d— one parent and $d-1$ children.)

(d) Prove that for even k the probability that a path of length k from the root of T_d ends up back at v is at least $2^{k - k \log d/2 - o(k)}$. H459

(e) Prove that for every n-vertex d-degree graph G, $\lambda(G) \geq \frac{2}{\sqrt{d}}(1 + o(1))$, where $o(1)$ denotes a term, depending on n and d that tends to 0 as n grows. H459

21.11 Let an 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 $1 \leq i \leq d$. Prove that a random n, d graph is an $(n, 2d, \frac{1}{10})$ edge expander with probability $1 - o(1)$ (i.e., tending to one with n). H459

21.12 In this exercise we show how to extend the error reduction procedure of Section 21.2.5 to two-sided (**BPP**) algorithms.

(a) Prove that under the conditions of Theorem 21.12, for every subset $I \subseteq [k]$,

$$\Pr[\forall_{1 \leq i \leq I} X_i \in B] \leq ((1-\lambda)\sqrt{\beta} + \lambda)^{|I|-1} .$$

(b) Conclude that if $|B| < n/10$ and $\lambda < 1/100$ then the probability that there exists a subset $I \subseteq [k]$ such that $|I| > k/10$ and $\forall_{1 \leq i \leq I} X_i \in B$ is at most $2^{-k/100}$.

(c) Use this to show a procedure that transforms every **BPP** algorithm A that uses m coins and decides a language L with probability 0.9 into an algorithm B that uses $m + O(k)$ coins and decides the language L with probability $1 - 2^{-k}$.

21.13 Prove that for every n-vertex d-regular graph, there exists a subset S of $n/2$ vertices, such that $E(S, \bar{S}) \leq dn/4$. Conclude that there does not exist an (n, d, ρ) edge expander for $\rho > 1/2$. H459

21.14 Prove the Expander Mixing Lemma (Lemma 21.11). H459

21.15 [Tan84] A graph where $|\Gamma(S)| \leq c|S|$ for every not-too-big set S (say, $|S| \leq n/(10d)$) is said to have *vertex expansion* c. This exercise shows that graphs with the minimum possible second eigenvalue $\frac{2}{\sqrt{d}}(1 + o(1))$ have vertex expansion roughly $d/4$. It is known that such graphs have in fact vertex expansion roughly $d/2$ [Kah92], and there are counterexamples showing this is tight. In contrast, random d-regular graphs have vertex expansion $(1 - o(1))d$.

(a) Prove that if $\mathbf{p}$ is a probability vector then $\|\mathbf{p}\|_2^2$ is equal to the probability that if i and j are chosen from $\mathbf{p}$, then $i = j$.

(b) Prove that if $\mathbf{s}$ is the probability vector denoting the uniform distribution over some subset S of vertices of a graph G with random-walk matrix A, then $\|A\mathbf{p}\|_2^2 \geq 1/|\Gamma(S)|$, where $\Gamma(S)$ denotes the set of S's neighbors.

(c) Prove that if G is an (n, d, λ)-expander graph, and S is a subset of ϵn vertices, then

$$|\Gamma(S)| \geq \frac{|S|}{2\lambda^2\left((1-\epsilon)^2 - 2\epsilon/\lambda^2\right)} .$$

H459

21.16 If G is a graph and S is a subset of G's vertices then by *contracting* S we mean transforming G into a graph H where all of S's members are replaced by a single vertex s with an edge $\overline{s\,v}$ in H for every edge $\overline{u\,v}$ in G where $u \in S$. Let G be an (n, d, ρ) edge expander, and let H be the $n' = n - (c-1)k$ vertex cd degree graph obtained by taking k disjoint c-sized subsets $S_1, \ldots, S_k$ of G's vertices and contracting them, and then adding self loops to the other vertices to ensure that the graph is regular. Prove that H is an $(n', cd, \rho/(2c))$ edge expander. Use this to complete the proof of Theorem 21.19. H459

21.17 Prove that for every function $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ and there exists an $(n, n-1)$-source X and a bit $b \in \{0,1\}$ such that $\Pr[\mathsf{Ext}(X)_1 = b] = 1$ (where $\mathsf{Ext}(X)_1$ denotes the first bit of $\mathsf{Ext}(X)$). Prove that this implies that $\Delta(\mathsf{Ext}(X), U_m) \geq 1/2$.

21.18 **(a)** Show that there is a deterministic poly(n)-time algorithm A that given an input distributed according to the distribution X with $H_\infty(X) \geq n^{100}$ and black box access to any function $f : \{0,1\}^n \to \{0,1\}$ outputs 1 with probability at least 0.99 if $\mathsf{E}[f(U_n)] \geq 2/3$ and outputs 0 with probability at least 0.99 if $\mathsf{E}[f(U_n)] \leq 1/3$. We call such an algorithm a *function approximator*.

(b) Show that there is no deterministic polynomial-time function approximator A without getting an additional randomized input (i.e., there is no deterministic function approximator). H459

(c) Show that for every probability distribution X, if $\Delta(X,Y) > 1/10$ for every Y with $H_\infty(Y) \geq n/2$, then there is no polynomial-time function approximator that gets X as an input. Conclude that access to a high min entropy distribution is necessary for black-box simulation of **BPP** algorithms. H459

21.19 . Say that a distribution Y is a *convex combination* of distributions $Y_1, \ldots, Y_N$ if there exist some non-negative numbers $\alpha_1, \ldots, \alpha_N$ summing up to 1 such that Y is the distribution obtained by picking i with probability α_i and sampling an element from Y_i. Prove that if this is the case then for every distribution X,

$$\Delta(X,Y) \leq \sum_i \alpha_i \Delta(X,Y_i) \leq \max_i \Delta(X,Y_i)\,.$$

H459

21.20 Suppose Boolean function f is (S,ϵ)-hard and let D be the distribution on m-bit strings defined by picking inputs $x_1, x_2, \ldots, x_m$ uniformly at random and outputting $f(x_1)f(x_2)\cdots f(x_m)$. Show that the statistical distance between D and the uniform distribution is at most ϵm.

21.21 Prove Lemma 21.26.

21.22 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 $\lambda_i \cdot \alpha_j$.

21.23 Prove that for every two graphs G, G', $\lambda(G \otimes G') \leq \lambda(G) + \lambda(G')$ without using the fact that every symmetric matrix is diagonalizable. H459

21.24 Let G be an n-vertex D-degree graph with ρ 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' edge expansion for some $\rho' > 0$. Prove that $G \circledR G'$ has at least $\rho^2\rho'/1000$ edge expansion. H459