CS225: Pseudorandomness	Prof. Salil Vadhan

Lecture 4: Random Walks

February 13, 2007	*Scribe: Dave Troiano and Brian Greenberg*

1 Graph Connectivity

One of the most basic problems in computer science is that of deciding connectivity in graphs, i.e.

S-T CONNECTIVITY: Given a directed graph G and two vertices s and t, is there a path from s to t in G?

This problem can of course be solved in linear time using breadth-first or depth-first search. However, these algorithms also require linear space. It turns out that S-T CONNECTIVITY can in fact be solved using much less workspace. (When measuring the space complexity of algorithms, we do not count the space for the (read-only) input and (write-only) output.)

Theorem 1 *There is an algorithm deciding* S-T CONNECTIVITY *using space* $O(\log^2 n)$ *(and time* $n^{O(\log n)}$*).*

Proof: The following recursive algorithm IsPath(G, u, v, k) decides whether there is a path of length at most k from u to v.

IsPath(G, u, v, k):

1. If $k = 0$, accept if $u = v$.
2. If $k = 1$, accept if $u = v$ or (u, v) is an edge in G.
3. Otherwise, loop through all vertices w in G and accept if both IsPath$(G, u, w, \lceil k/2 \rceil)$ and IsPath$(G, w, v, \lfloor k/2 \rfloor)$ accept for some w.

We can solve S-T CONNECTIVITY by running IsPath(G, s, t, n), where n is the number of vertices in the graph. The algorithm has $\log n$ levels of recursion and uses $\log n$ space per level of recursion (to store the vertex w), for a total space bound of $\log^2 n$. Similarly, the algorithm uses polynomial time per level of recursion, for a total time bound of $\mathrm{poly}(n)^{\log n} = n^{O(\log n)}$. ■

It is not known how to improve the space bound in Theorem 1 or to get the running time down to polynomial while maintaining space $n^{o(1)}$. For *undirected graphs*, however, we can do much better using a randomized algorithm. Specifically, we can place it in the following class:

Definition 2 *A language L is in* **RL** *if there exists a randomized algorithm A that always halts, uses space at most $O(\log n)$ on inputs of length n, and satisfies:*

- $x \in L \Rightarrow \Pr[A(x) \text{ accepts}] \geq \frac{1}{2}$
- $x \notin L \Rightarrow \Pr[A(x) \text{ accepts}] = 0$

Recall that our model of a randomized space-bounded machine is one that has access to a coin-tossing box (rather than an infinite tape of random bits), and thus must explicitly store in its workspace any random bits it needs to remember. The requirement that A always halts ensures that its running time is at most $2^{O(\log n)} = \mathrm{poly}(n)$, because otherwise there would be a loop in its configuration space. Similarly to **RL**, we can define **L** (deterministic logspace), **co-RL** (one-sided error with errors only on NO instances), and **co-BPL** (two-sided error).

Now, we can state the theorem regarding connectivity in undirected graphs.

UNDIRECTED S-T CONNECTIVITY**:** Given an undirected graph G and two vertices s and t, is there a path from s to t in G?

Theorem 3 UNDIRECTED S-T CONNECTIVITY *is in* **RL***.*

Proof: The algorithm simply does a polynomial-length random walk starting at s.

Random-Walk Algorithm for UNDIRECTED S-T CONNECTIVITY**.**
On input (G, s, t), where $G = (V, E)$ has n vertices:

1. Let $v = s$.
2. Repeat up to n^4 times:
   (a) If $v = t$, halt and accept.
   (b) Else let $v \stackrel{\mathrm{R}}{\leftarrow} \{w : (v, w) \in E\}$.
3. Reject (if we haven't visited t yet).

Notice that this algorithm only requires space $O(\log n)$, to maintain the current vertex v as well as a counter for the number of steps taken. Clearly, it never accepts when there isn't a path from s to t. In the next section, we will prove that if G is a d-regular graph, then a random walk of length $\tilde{O}(d^2 n^3)$ from s will hit t with high probability. Note that this suffices for the theorem, because make an arbitrary undirected graph 3-regular while preserving s-t connectivity by replacing each vertex v with a cycle of length $\deg(v)$. In fact, the algorithm actually works as described above for general undirected graphs and even directed graphs in which each connected component is Eulerian ($\mathrm{indeg}(v) = \mathrm{outdeg}(v)$ for every vertex), but we will not prove it here. But it does not work for arbitrary directed graphs. Indeed, it is not difficult to construct directed graphs in which there is a path from s to t but a random walk from s takes exponential time to hit t. ■

This algorithm, dating from the 1970's, was derandomized only in 2005. We will cover this result in just a few weeks! However, the general question derandomizing space-bounded algorithms remains open.

Open Problem 4 *Does* **RL** $=$ **L***? Does* **co-BPL** $=$ **L***?*

2 Random Walks on Graphs

Throughout this section, G will be a d-regular directed (multi)graph on n vertices. By *d-regular*, we mean that every vertex has indegree d and outdegree d. By multigraph, we mean that we allow G to have parallel edges and self-loops. To analyze the random-walk algorithm of the previous section, it suffices to prove a bound on the *hitting time* of random walks.

Definition 5 $\mathrm{hit}(G) = \max_{i,j}\{$*expected number of steps for a random walk started at* i *to visit* $j\}$

Today, we will prove:

Theorem 6 *For every connected and regular undirected graph* G *on* n *vertices, we have* $\mathrm{hit}(G) = O(d^2 n^3 \log n)$.

Thus, if s and t are in the same connected component C of G and we do a random walk from s for $n^4 \geq 2\mathrm{hit}(C)$ steps (for $d = 3$), then we will visit t with probability at least $1/2$.

There are combinatorial methods for proving the above theorem, but we will prove it using a linear-algebraic approach, as these methods will be very useful in our study of expander graphs. For a d-regular digraph G on n vertices, we can define its *random-walk transition matrix* M, which is simply the adjacency matrix of G divided by d. That is, $M_{i,j}$ is the probability of going from vertex i to vertex j in one step. Notice that for every probability distribution $\pi \in \mathbb{R}^n$ on the vertices of G (written as a row vector), the vector πM is the probability distribution obtained by selecting a vertex i according to π and then taking one step of the random walk to end at a vertex j. This is because $(\pi M)_j = \sum_i \pi_i M_{i,j}$.

In our application, we start at a probability distribution π concentrated at vertex s, and are interested in the distribution πM^k we get after taking k steps on the graph. Specifically, we'd like to show that it places nonnegligible mass on vertex t for $k = \mathrm{poly}(n)$. We will do this by showing that it in fact converges to the uniform distribution $u = (1/n, 1/n, \ldots, 1n) \in \mathbb{R}^n$ within a polynomial number of steps. Note that $uM = u$ by the regularity of G, so convergence to u is possible (and will be guaranteed given some additional conditions on G).

Will measure the rate of convergence in ℓ_2 norm. For vectors $x, y \in \mathbb{R}^n$, define $\langle x, y\rangle = \sum_i x_i y_i$, and $\|x\| = \sqrt{\langle x, x\rangle}$. We write $x \perp y$ to mean that $\langle x, y\rangle = 0$. We want to determine how large k needs to be so that $\|\pi M^k - u\|$ is 'small'. This is referred to as the *mixing time* of the random walk. Mixing time can be defined with respect to various distance measures and the ℓ_2 norm is not the most natural one, but it has the advantage that we will be able to show that the distance decreases in every step. This is captured by the following quantity.

Definition 7 *For a regular directed graph* G *with random-walk matrix* M*, we define*

$$\lambda(G) \stackrel{\text{def}}{=} \max_{\pi} \frac{\|\pi M - u\|}{\|\pi - u\|} = \max_{x \perp u} \frac{\|xM\|}{\|x\|},$$

where the first maximization is over all probability distributions $\pi \in [0,1]^n$ *and the second is over all vectors* $x \in \mathbb{R}^n$ *such that* $x \perp u$. *We write* $\gamma(G) \stackrel{\text{def}}{=} 1 - \lambda(G)$.

To see that the first definition of $\lambda(G)$ is smaller than or equal to the second, note that for any probability distribution π, the vector $x = (\pi - u)$ is orthogonal to uniform (i.e. the sum of its entries is zero). For the converse, observe that given any vector $x \perp u$, the vector $\pi = u + \alpha x$ is a probability distribution for a sufficiently small α. It can be shown that $\lambda(G) \in [0, 1]$. (For undirected graphs, this follows from Problem Set 2 and the material in the next section.)

The following lemma is immediate from the definition of M.

Lemma 8 *Let G be a regular digraph with random-walk matrix M. For every initial probability distribution π on the vertices of G and every $k \in \mathbb{N}$, we have*

$$\|\pi M^k - u\| \le \lambda(G)^k \cdot \|\pi - u\| \le \lambda(G)^k.$$

Thus a smaller value of $\lambda(G)$ (equivalently, a larger value of $\gamma(G)$) mean that the random walk mixes more quickly. Specifically, for $k = \ln(n/\varepsilon)/\gamma(G)$, it follows that every entry of πM^k has probability mass at least $1/n - (1 - \gamma(G))^k \ge (1 - \varepsilon)/n$. So the mixing time of the random walk on G is at most $O((\log n)/\gamma(G))$, with respect to any reasonable distance measure. Note that $O(1/\gamma(G))$ steps does not suffice, because a distribution with ℓ_2 distance ε from uniform could just assign equal probability mass to $1/\varepsilon^2$ vertices (and thus be very far from uniform in any intuitive sense).

Corollary 9 $\mathrm{hit}(G) = O(n \log n/\gamma(G))$.

Proof: As argued above, a walk of length $k = O(\log n/\gamma(G))$ has a probability of at least $1/2n$ of ending at j. Thus, we expect to hit j within $2n$ such walks, for a total expected walk length of $2n \cdot k = O(n \log n/\gamma(G))$. ■

Thus we are left with the task of showing that $\gamma(G) \ge 1/\mathrm{poly}(n)$. You will do this on Problem Set 2, using a connection with eigenvalues described in the next section.

3 Eigenvalues

Recall that $v \in \mathbb{R}^n$ is an *eigenvector* of $n \times n$ matrix M if $vM = \lambda v$ for some $\lambda \in \mathbb{R}$, which is called the corresponding *eigenvalue*. A useful feature of *symmetric* matrices is that they can be described entirely in terms of their eigenvectors and eigenvalues.

Theorem 10 *If M is a symmetric $n \times n$ real matrix with distinct eigenvalues $\mu_1, \ldots, \mu_k$, then the subspaces $W_i = \{x : x$ is an eigenvector of eigenvalue $\mu_i\}$ are orthogonal (i.e. $x \in W_i$, $y \in W_j \Rightarrow x \perp y$ if $i \ne j$) and span $\mathbb{R}^n$ (i.e. $\mathbb{R}^n = W_1 + \cdots + W_k$). We refer to the dimension of W_i as the* multiplicity *of eigenvalue λ_i. In particular, $\mathbb{R}^n$ has a basis consisting of orthogonal eigenvectors $v_1, \ldots, v_n$ having respective eigenvalues $\lambda_1, \ldots, \lambda_n$, where the number of times μ_i occurs among the λ_j's exactly equals the multiplicity of μ_i.*

Notice that if G is an undirected graph, then its random-walk matrix M is symmetric. We know that $uM = u$, so the uniform distribution is an eigenvector of eigenvalue 1. Let $v_2, \ldots, v_n$ and $\lambda_2, \ldots, \lambda_n$

be the remaining eigenvectors and eigenvalues, respectively. Given any probability distribution π, we can write it as $\pi = u + c_2v_2 + \cdots + c_nv_n$. Then the probability distribution after k steps on the random walk is

$$\pi M^k = u + \lambda_2^k c_2 v_2 + \cdots + \lambda_n^k c_n v_n.$$

On Problem Set 2, you will show that all of the λ_i's have absolute value at most 1. Notice that if they all have have magnitude strictly smaller than 1, then πM^k indeed converges to u. Thus it is not surprising that our measure of mixing rate, $\lambda(G)$, equals the absolute value of the second largest eigenvalue.

Lemma 11 *Let G be an undirected graph with random-walk matrix M. Let $1 = \lambda_1 \geq |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$ be the eigenvalues of M. Then $\lambda(G) = |\lambda_2|$.*

Proof: Let $u = v_1, v_2, \ldots, v_n$ be the basis of orthogonal eigenvectors corresponding to the λ_i's. Given any vector $x \perp u$, we can write $x = c_2v_2 + \cdots + c_nv_n$. Then:

$$\begin{aligned}
\|xM\|^2 &= \|\lambda_2 c_2 v_2 + \cdots + \lambda_n c_n v_n\|^2 \\
&= \lambda_2^2 c_2^2 \|v_2\|^2 + \cdots + \lambda_n^2 c_n^2 \|v_n\|^2 \\
&\leq |\lambda_2|^2 \cdot (c_2^2\|v_2\|^2 + \cdots + c_n^2\|v_n\|^2) \\
&= |\lambda_2|^2 \cdot \|x\|^2
\end{aligned}$$

Equality is achieved with $x = v_2$. ■

Thus, bounding $\lambda(G)$ amounts to bounding the eigenvalues of G. On Problem Set 2, you will prove:

Theorem 12 *If G is a connected, nonbipartite, and regular undirected graph, then $\gamma(G) = \Omega(1/(dn)^2)$*

Combining Theorem 12 with Corollary 9, we deduce Theorem 6. (The nonbipartite assumption in Theorem 12 can achieved by adding a self-loop to each vertex, which only increases the hitting time.) We note that the bounds presented in this lecture are not tight.

4 Markov Chain Monte Carlo

Random walks are a very widely used tool in the design of randomized algorithms. In particular, they are the heart of the "Markov Chain Monte Carlo" method, which is widely used in statistical physics and for solving approximate counting problems. In these applications, the goal is generate a random sample from an exponentially large space, such as an (almost) uniformly random perfect matching for a given bipartite graph G. (It turns out that this is equivalent to approximately counting the number of perfect matchings in G.) The approach is to do a random walk on an appropriate (regular) graph $\hat{G}$ defined on the state space (e.g. by doing random local changes on the current perfect matching). Note that $\hat{G}$ is of size exponential in the input size $n = |G|$. Nevertheless, in many cases it can be proven to have mixing time $\mathrm{poly}(n) = \mathrm{polylog}(|\hat{G}|)$ (sometimes via eigenvalues) so that we can get an almost-uniform sample in polynomial time. These Markov Chain Monte Carlo methods provide some of the best examples of problems where randomization yields algorithms that are exponentially faster than all known deterministic algorithms.