Lecture 01: Randomized Algorithm for Reachability Problem

Topics in Pseudorandomness and Complexity (Spring 2018)
Rutgers University
Swastik Kopparty

Scribes: Neelesh Kumar and Justin Semonsen

The lecture is an introduction to randomized algorithms, and builds a motivation for using such algorithms by taking the example of reachability problem.

1 Deterministic Algorithm vs. Randomized Algorithm

We are given some function $f : \{0,1\}^n \to \{0,1\}$, and we wish to compute $f(x)$ quickly. A deterministic algorithm $\mathcal{A}$ for computing f is one which always produces the same $\mathcal{A}(x) = f(x)$, $\forall x \in \{0,1\}^n$.
On the other hand, a randomized algorithm $\mathcal{A}$ for computing f takes two arguments- the input x and a random string r such that $\forall x \in \{0,1\}^n$,

$$Pr_{r\in\{0,1\}^n}[\mathcal{A}(x,r) = f(x)] \geq 0.9 \tag{1}$$

i.e. with high probability over choices of r, $\mathcal{A}$ produces the correct result. It should be noted that the algorithm $\mathcal{A}$ itself may not have inherent randomness, rather it might make random calls deterministically. It should also be noted that (1) should hold true for all inputs x, and for most choices of r. Some examples of such algorithms are polynomial identity testing, primality testing (prior to 2002, the only way to do this was using a randomized algorithm), random sampling, etc.

2 Motivating Example

Let us consider an interesting example to motivate the use of randomized algorithm. We are given as input a graph G and two of its vertices s and t. We wish to know if there exists a path from s to t in G.

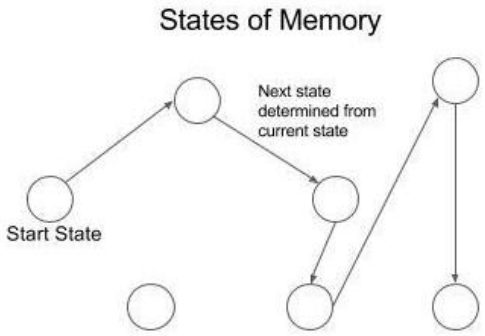
Approach 1: BFS

A common approach to solve this is using Breadth First Search (BFS). Essentially, starting from s, we mark all the neighbors of s, i.e. vertices directly reachable from s. We do this recursively until all the neighbors have been marked. We then check if vertex t is marked or not. The algorithm uses n bits of space where n is the number of vertices.

A question that arises is can we accomplish this using less space?

Approach proposed in class: Exponential Time

An idea that was proposed in the class is to leverage space for time, i.e. reduce the space complexity by taking exponential time. Basically, the proposed idea is to traverse all possible paths from s and check if t is in any of them. However to use less than n bits of space, the algorithm must take less than exponential time by the following logic:



For any given state of memory, the next state is determined from current state. If we use s bits of space, memory can be in only one of the 2^s possible configurations of states of memory. So for an algorithm with runtime $O(2^s)$, one needs s bits of space to specify the current state.

Approach 2: Savitch's Theorem

This approach uses Savitch's theorem. The key is to define an intermediate problem and do recursion on it. We define the recursive intermediate problem as follows:

$$h(u,v,i) = \begin{cases} 1, \text{if there is a path from } u \text{ to } v \text{ with } \leq 2^i \text{ steps} \\ 0, \text{otherwise} \end{cases}$$

We wish to understand when the above recursion would be equal to 1. Lets pick a path from u to v. If we encounter a vertex w between u and v such that the length of the path between u and w and between w and v is less than 2^{i-1} each, then we can say that $h(u,v,i) = 1$, otherwise it is 0. Hence

$$h(u,v,i) = \bigvee_w (h(u,w,i-1) \bigwedge h(w,v,i-1)$$

We can reformulate our reachability problem as computing $h(s,t,\log n)$. Next we analyze the space and time requirement of this recursion. Let $f(i)$ be the total space needed to compute $h(u,v,i)$. Then $f(i)$ is equal to the sum of the space required to maintain a counter over space of all w, the space needed to compute $h(u,v,i-1)$ and some constant space needed to compute AND/OR operations. Notice that since the expressions $h(u,w,i-1)$ and $h(w,v,i-1)$ are computed sequentially, the space needed to compute one of them can be reused to compute the other. Mathematically:

$$f(i) \leq \log n + f(i-1) + 10$$

On solving the recurrence and substituting for $i = \log n$, we get:

$$f(\log n) \leq O(\log^2 n)$$

which is the space complexity of the algorithm.
Now let $t(i)$ be the time complexity of the algorithm. Using argument similar to that of space complexity, we can write

$$t(i) \leq n(2t(i-1) + 5)$$

Unlike space, the calls to $(h(u, w, i-1)$ and $h(w, v, i-1)$ will each have their own contribution to time since they are run sequentially.

$$t(\log n) \leq n^{O(\log n)}$$

This is a significant improvement in space complexity as compared to BFS. But we would like to do even better. We would also like to do it in polynomial time. Fortunately, there exists a randomized algorithm that uses even lesser space and polynomial time.

Approach 3: Randomized algorithm for undirected $s - t$ connectivity

The Random Walk algorithm is remarkably simple and consists of the following steps:

1. Initialize $u = s$.
2. Pick a random vertex v.
3. If v is adjacent to u, set $u = v$.
4. If $u = t$, then return CONNECTED.
5. Else, go to step 2. (Stop after n^{10} steps).
6. Return NOT CONNECTED.

It should be noted that the above algorithm only works for undirected connectivity, while approach 2 that uses Savitch's theorem works for both directed and undirected connectivity. The space requirement for this algorithm is $O(\log n)$ and the time requirement is $n^{O(1)}$. With high probability, the algorithm will return the correct result and the following theorem formalizes this:

Theorem 1. *(a) If there is no path from s to t , then Pr[Algo says NOT CONNECTED] =1.*

(b) If there is a path from s to t, Pr[Algo says CONNECTED] $\geq 1 - 2^{-O(n)}$

Theorem $1(a)$ is trivial. We will now build arguments to prove theorem $1(b)$. To make the analysis easy, we will slightly modify the random walk algorithm, without affecting its time and space requirement. The modification is that in step 5, instead of stopping after n^{10} steps, the algorithm stops after n^5 steps, and the entire process is repeated n^5 times.

We are given an n vertex graph with the vertices labeled numerically from 1 through n. Let $P \in \mathbb{R}^n$ be a probability distribution on the vertices which implies:

1. $P_i \geq 0$
2. $\Sigma_i P_i = 1$

Consider the following process:

1. Pick $u \in [n]$ according to P.
2. Pick a random neighbor v of u, uniformly from amongst the neighbors of u
3. Output v

The probability that we output a vertex $v = v_0$ can be written as:

$$Pr[v = v_0] = \sum_{u \text{ adjacent to } v_0} P_u . \frac{1}{deg(u)}$$

where P_u is the probability of picking u according to probability distribution P and $\frac{1}{deg(u)}$ is the probability of picking v_0 (picked uniformly from neighbors of u). The new probability distribution can then be written as MP where M is a matrix such that:

$$M_{uv} = \begin{cases} 0, \text{if } u \text{ is not adjacent to } v \\ \frac{1}{deg(v)}, \text{if } u \text{ is adjacent to } v \end{cases}$$

Then

$$(MP)_u = \Sigma_v M_{uv} P_v$$

$$(MP)_u = \sum_{\text{v adjacent to u}} \frac{1}{deg(v)} P_v$$

We now adapt the above process to our reachability problem. We start at the vertex s. The initial probability distribution is $P = (0\ 0\ 0\ ..\ 1\ 0\ 0..)$ where probability is 1 for picking s and 0 for all other vertices. Every time we take a step, the distribution keeps becoming more and more uniform. After i steps, the probability distribution is $M^i P$. The matrix M essentially spreads out the distribution. For large i, P is well distributed and spread over all connected components containing s. All other components should have zero mass. We will now formally prove this intuition.

Sidenote: Eigenvalues
If M is a symmetric matrix, then M has an orthonormal basis of eigenvectors:

$$M = \sum_{i=1}^{n} \lambda_i v_i v_i^T$$

where v_i are orthonormal basis and $Mv_i = \lambda_i v_i$, λ_i are real.

We will assume that G is a d-regular graph which means that every vertex has degree d. Consequently, M is symmetric. The analysis can easily be extended to general graphs.
If G is connected, then we will shortly show that $M^i P \to \frac{1}{\sqrt{n}} v_1$ where the vector $v_1 = (\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}....., \frac{1}{\sqrt{n}})$

is an eigenvector with eigenvalue $\lambda_1 = 1$. If G is disconnected, then the same applies for the connected component containing s.
To prove this, we express P as a linear combination of eigenvectors:

$$P = \Sigma a_i v_i$$

Also since $v_i's$ are orthonormal,

$$\Sigma P_i^2 = \Sigma a_i^2 = 1$$

Thus,

$$a_1 =< P, v_1 >= \frac{1}{\sqrt{n}}$$

Now, for any j, we can write:

$$M^j P = \Sigma_i a_i M^j v_i$$

Since v_i is an eigenvector,

$$M^j P = \Sigma_i a_i \lambda_i^j v_i$$

Separating out the first eigenvector,

$$= a_1 \lambda_1^j v_1 + \sum_{i \geq 2} a_i \lambda_i^j v_i$$

Using the fact that $\lambda_1 = 1$,

$$= a_1 v_1 + \sum_{i \geq 2} a_i \lambda_i^j v_i$$

For the second part of the above sum, if $\forall i \geq 2$, $|\lambda_i| << 1$, i.e. the eigenvalues are noticeably lesser than 1, then λ_i^j will go down very fast, and the sum will disappear. For large j, the above sum reduces to $a_1 v_1$ which is equal to $\frac{1}{\sqrt{n}} v_1$, and hence approaches uniform distribution amongst all the vertices in the connected components containing s. The implication of this is that the algorithm will almost surely see t if there is a path between s and t. We have the following claim:

Claim 2. *(a) All $\lambda_i \in [-1, 1]$*

(b) $\lambda_2 = 1$ iff G is disconnected.

(c) $\lambda_n = -1$ iff G has a bipartite component.

For proving the claims, we assume that we have a graph G on $[n]$ that is d-regular. Let M be the normalized adjacency matrix of G. Note that M is symmetric and thus has a basis of eigenvectors. let the eigenvalues of M be $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$. It is clear that $\frac{\mathbf{1}}{\sqrt{n}}$ is an eigenvector of M with eigenvalue 1.

Proof. 1. Assume $Mv = \lambda_v$. Then $(Mv)_i = \sum_{j \sim i} \frac{(v)_j}{d} = \mathbb{E}_{j \sim i}[(v)_j] = \lambda(v)_i$. Choose i maximizing $|(v)_i|$. Then $|\lambda|\,|(v)_i| = |\mathbb{E}_{j \sim i}[(v)_j]| \leq \mathbb{E}_{j \sim i}[|(v)_j|] \leq |(v)_i|$, so $|\lambda| \leq 1$.

2. Let C_1 and C_2 be disconnected components. Let $w_1 = \mathbf{1}_{C_1}$ and $w_2 = \mathbf{1}_{C_2}$ be uniform on each component respectively. Then $Mw_1 = w_1$ and $Mw_2 = w_2$ with $w_1 \perp w_2$, so there are multiple eigenvectors with eigenvalue 1.

   Let $\lambda_2 = 1$, meaning $\exists\, v \perp \frac{\mathbf{1}}{\sqrt{n}}$ with $Mv = v$. Let i be such that $(v)_i$ is maximum. Then $(v)_i = \mathbb{E}_{j\sim i}[|(v)_j|]$, so $(v)_j = (v)_i$ for any $j \sim i$. By extension, this means that v is uniform on the component containing i. If G were connected, then $v = (v)_i\mathbf{1}$, but this contradicts that $v \perp \frac{\mathbf{1}}{\sqrt{n}}$, meaning G is not connected.

3. Let A and B be the partitions of a bipartite component. Let $v = \mathbf{1}_A - \mathbf{1}_B$ be uniform on A, negative and uniform on B (with the same value), and 0 elsewhere. Then clearly $Mv = -v$, so $\lambda_n = -1$.

   If $\lambda_n = -1$, let v be a vector such that $Mv = -v$. Let i be such that $|(v)_i|$ is maximum. Then $(v)_i = -\mathbb{E}_{j\sim i}[|(v)_j|]$, so $-(v)_j = (v)_i$ for any $j \sim i$. The same logic applies to the neighbors of these vertices and we can see that this is true for all vertices in the component of i. Let $A = \{j : v_j = v_i\}$ and $B = \{j : v_j = -v_i\}$. Note that this is a bipartite partition of this component.

□

3 Lazy Random Walk

Instead of using a random walk, we use a 'lazy' version to avoid potential issues with G being bipartite. A lazy random walk is done as follows:

1. Start at s.
2. For k iterations, move to a random neighbor of the current vertex with probability $\frac{1}{2}$, and otherwise stay at the same vertex.

Note that the state transition matrix for this process is $\frac{I+M}{2}$, with the I representing staying in place, and the denominator accounting for the probability of doing each of the two options. Note that the eigenvectors of $\frac{I+M}{2}$ are the same as those of M, but the eigenvalues are $\frac{1+\lambda_i}{2}$ instead of each λ_i. The same properties from above carry over, modifying the numbers accordingly.

This means that $\left(\frac{I+M}{2}\right)^n p = a_1v_1 + \sum_{i\geq 2}\left(\frac{1+\lambda_i}{2}\right)^n a_iv_i$ for v_i eigenvectors and a_i such that $p = \sum_i a_iv_i$.

4 Lazy Random Walks in s-t Connectivity

We use a similar algorithm as before, doing n^5 independent random walks of length n^5 from s, stopping and saying they are connected if we find t on any of these walks. If we never find t we say it is not connected to s. Note that this algorithm will always be right in this second case, so we only need analyze the case where s is connected to t, but we don't find it. In this case p is the vector with a 1 at s and zeros elsewhere. This also means $a_1 = \frac{1}{\sqrt{n}}$.

In this case, we only consider the graph to be the component of that contains S, as the rest doesn't apply. We then can prove the following lemma:

Lemma 3. $\lambda_2 \leq 1 - \frac{1}{dn^3}$

Using this lemma and that $d \leq n$, we have:

$$\left|\left(\frac{I+M}{2}\right)^k p - a_1 v_1\right|^2 \leq \sum_{i\geq 2} a_i^2 \left(\frac{1+\lambda_i}{2}\right)^{2k} \leq \left(\sum_i a_i^2\right)) \left(1 - \frac{1}{2dn^3}\right)^{2k} < e^{\frac{-k}{n^4}}$$

Because we do $k = n^5$ length paths, we know that $\forall\, i, \left(\left(\frac{I+M}{2}\right)^k p\right)_i \in \left(\frac{1}{n} - e^{-n}, \frac{1}{n} + e^{-n}\right)$.

This means that if s is in the same component as t (and $e^{-n} \leq \frac{1}{2n}$), $Pr[\text{path doesn't end on } t] < 1 - \frac{1}{2n}$. This means that the probability none of the paths end on t (not even including passing through!) is less than $\left(1 - \frac{1}{2n}\right)^{n^5} < e^{\frac{n^4}{2}}$, so this algorithm rarely fails.

All that remains is a proof of the Lemma:

Proof. Note that $\langle v, (I-M)v\rangle = \sum_i (v)_i^2 - \sum_i \sum_{j\sim i} (v)_i (v)_j / d = \frac{1}{2}\sum_i \mathbb{E}_{j\sim i}[((v)_i - (v)_j)^2]$. This means that $\frac{1}{n}\langle v, (I-M)v\rangle = \frac{1}{2}\mathbb{E}_i \mathbb{E}_{j\sim i}[((v)_i - (v)_j)^2]$.

This means that if $v \perp \frac{\mathbf{1}}{\sqrt{n}}$ and $|v| = 1$, we know that $\sum_i (v)_i^2 = 1$ and $\sum_i (v)_i = 0$. The first equality means that $\exists\, i$ such that $|(v)_i| \geq \frac{1}{\sqrt{n}}$. If $(v)_i < 0$, we can negate the vector v, and thus we may assume $(v)_i \geq \frac{1}{\sqrt{n}}$.

Because $\sum_i (v)_i = 0, \exists\, j$ such that $(v)_j < 0$. Because i and j are connected, there is a path of length at most n between them. Therefore there must be some r, s on this path such that $(v)_r - (v)_s \geq \frac{1}{\sqrt{n^3}}$.

Just using this term, $\mathbb{E}_i \mathbb{E}_{j\sim i}[((v)_i - (v)_j)^2] \geq \frac{2}{dn^4}$, so $\langle v, (I-M)v\rangle \geq \frac{1}{dn^3}$. If v is the eigenvector corresponding to λ_2, this means $1 - \lambda_2 \geq \frac{1}{dn^3}$. □