**CS787: Advanced Algorithms**

**Scribe:** Blayne Field | **Lecturer:** Shuchi Chawla
**Topic:** Random Walks and Markov Chains | **Date:** November 26, 2007

## 27.1 Introduction

As we have seen in the course up to this point, randomization has been a useful tool for developing simple and efficient algorithms. So far, most of these algorithms have used some sort of independent coin tosses. In this lecture, we started looking at choosing a sample from a large space, with our choices being dependent on previous choices. We looked at this concept through the framework of random walks and Markov chains.

**Definition 27.1.1** *A random walk is a process in which at every step we are at a node in an undirected graph and follow an outgoing edge chosen uniformly at random. A Markov chain is similar, except the outgoing edge is chosen according to an arbitrary distribution.*

### 27.1.1 Examples of random walks

Random walks can occur in many situations, and are useful for analyzing various scenarios. Consider the following betting game: a player bets \$1, and either loses it or wins an addition dollar with probability $\frac{1}{2}$. Since the probability of either thing happening is equal, we can think of this as a random walk on a line graph, where each node represents the amount of wealth at any point of time. This allows us to learn about numerous aspects of the game, such as the probability distribution of the amount of money at a given time. We can also ask about the probability of the player running out of money before winning a certain amount, and if that happens, what is the expected amount of time before that happens.

Another situation where random walks occur is shuffling a deck of cards. One possible way of drawing a permutation u.a.r. is to start at an arbitrary permutation and apply a local shuffling operation multiple times. This can be thought of as a random walk on a graph of all permutations. Our graph would contain nodes for each of the 52! permutations, and the connectivity of the graph would be determined by what local operations are allowed. For example, if we could just randomly choose 1 card, and move it to the top, each node would have degree 52 (if we include self-loops). Analyzing this random walk may allow us to determine how many local steps we would need to take in order to be reasonably close to the uniform distribution.

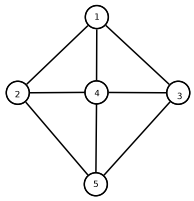
## 27.2 Properties of random walks

There are numerous properties of random walks that we may be interested in. They are listed and defined informally below:
Stationary distribution: what is the distribution of our current location if we run the random walk for an infinite number of steps.
Hitting time: denoted $h_{uv}$ is the expected time to get from $u$ to $v$.

Commute time: denoted $C_{uv}$ is the expected time to get from u to v, and back to u
Cover time (starting at u): denoted $C_u$ is the expect time to visit every node (starting at node u)
Cover time (for a graph): denoted $C(G) = \max_u C_u$.

## 27.3 Transition matrix

A random walk (or Markov chain), is most conveniently represented by its transition matrix P. P is a square matrix denoting the probability of transitioning from any vertex in the graph to any other vertex. Formally, $P_{uv}$ = Pr[going from u to v, given that we are at u]. Thus for a random walk, $P_{uv} = \frac{1}{d_u}$ if $(u, v) \in E$, and 0 otherwise (where $d_u$ is the degree of u). Thus for the example graph given below the transition matrix would be:




$$\mathbf{P} = \begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}$$

## 27.4 Stationary distributions

Thus if we have a distribution $\pi$ over the nodes, we can obtain the distribution after one step by computing $\pi' = P^T \cdot \pi$. Using this definition, we can define a stationary distribution $\pi_s$ as a distribution with the property that $P^T \cdot \pi_s = \pi_s$. Stationary distributions are not always unique, but under certain conditions, they are. It also is the case that under certain conditions, $lim_{t\to\infty}(P^T)^t\pi = \pi_s$ for all starting distributions $\pi$. We will discuss these conditions in the next lecture.

For the following examples, consider a graph $G = (V, E)$, with $n = |V|$ and $m = |E|$. Let $d_u$ denote the degree of vertex $u$.

**Lemma 27.4.1** *$\pi_v = \frac{d_v}{2m}$ is a stationary distribution for G*

**Proof:** $(P^T \cdot \pi)_u = \sum_v P_{vu}\pi_v = \sum_{v:(v,u)\in E} \frac{d_v}{2m} * \frac{1}{d_v} = \sum_{v:(v,u)\in E} \frac{1}{2m} = \pi_u$ (because $P_{vu} = \frac{d_v}{2m}$ if $(v, u) \in E$ and 0 otherwise). ■

## 27.5 Random walks on edges

For our next result, we will think about random walks in a slightly different way, where instead of walking from node to node, our walk goes from edge to edge. Whenever we are on an edge $uv$, we choose the next edge u.a.r from the set of edges incident to $v$. Then we can see that the uniform distribution on edges ($\pi_{u\to v} = \frac{1}{2m}\ \forall(u \to v) \in E$) is a stationary distribution. This is because $(P^T \cdot \pi)_{v\to w} = \sum_{u:(u,v)\in E} \frac{1}{2m}\frac{1}{d_v} = \frac{1}{2m} = \pi_{v\to w}$

**Lemma 27.5.1** *$\forall(u, v) : (u, v) \in E$, we have $C_{uv} \leq 2m$*

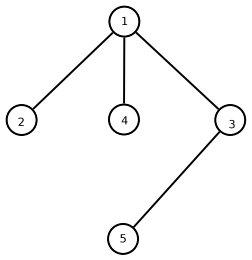
**Proof:** Note: this is just a sketch of the proof, as certain technical details are omitted. If we view the process as a random walk on sequence of edges, we can bound the commute time by the expected amount of time between consecutive occurences of the edge $u \to v$. The expected length of the gap between consecutive occurences if we run for $t$ steps is simply $t$ divided by the actual number of times we see the edge $u \to v$. We also know that since the stationary distribution is uniform, we expect to see the edge $\frac{t}{2m}$ times. As $t$ goes to infinity, the actual number of times we see $u \to v$ appraoches its expectation $\frac{t}{2m}$ with probability 1 (due to the law of large numbers). We can then approximate the actual number seen by the expected number seen, and thus we expect the length of the gap to be $\frac{t}{\frac{t}{2m}} = 2m$. ■

## 27.6 Cover time

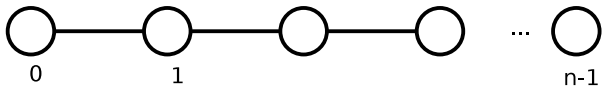
If we had a bound on the commute time for all pairs $(u, v)$ (call this bound $x$), we could get a bound (in expectation) on the cover time by running the random walk for $x * n$ steps. Unfortunately, the bound given by lemma 27.5.1 is only valid for pairs $(u, v)$ where there is an edge between $u$ and $v$. However, we can still come up with a different method for bounding the cover time.

**Lemma 27.6.1** $C(G) \leq 2m(n-1)$

**Proof:** Let $T$ be an arbitrary spanning tree of $G$. For each edge $(u, v)$ in $T$, add the edge $(v, u)$. We can then bound the cover time of $G$ with the expected time needed to complete an Euler tour of $T$. Since each node in $T$ has even degree (due to the doubling of the edges), we know that an Euler tour must exist. If we list the vertices visited as $v_1, v_2, \ldots, v_k = v_0$, we have $C(G) \leq h(v_0v_1) + h(v_1v_2) + \ldots + h(v_{k-1}v_0) = \sum_{uv\in T} h(u, v) + h(v, u) = \sum_{uv\in T} C_{uv} \leq 2m(n-1)$, since each of the $(n-1)$ edges that was in $T$ orginally shows up in both directions. For example, the cover time of the graph given before could be bounded by using the spanning tree below, so $C(G) \leq h_{21} + h_{14} + h_{41} + h_{13} + h_{35} + h_{53} + h_{31} + h_{12}$. ■

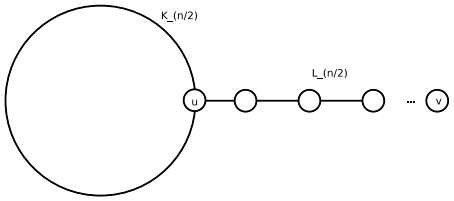


It turns out that for some graphs, the bound given in lemma 27.6.1 is tight. One example of this is the line graph with $n$ vertices, depicted below ($L_n$). According to the lemma, $C(G) \leq 2(n-1)^2 = O(n^2)$. Also, we can note that $h_{1,0} \leq 2(n-1)-1$, since $h_{0,1} = 1$, and $C_{uv} = h_{uv} + h_{vu}$ by linearity of expectation. We will show in the next lecture that the cover time for this graph is indeed $\Theta(n^2)$.




However, the bound is not always tight, as in the case of the complete graph, $K_n$. In this case, $m = \frac{n(n-1)}{2}$, so $C(K_n) = O(n^3)$ by the lemma. However, we can get a much tighter bound on the cover time for this graph. Since in the complete graph, we can go from any node to any other in one step, the problem of visiting all nodes can be viewed as an instance of the coupon collector problem. As we saw in a previous lecture, this would give us a bound of $O(n \log n)$.

One last example is the lollipop graph (pictured below) , which has n vertices, half of which form $K_{\frac{n}{2}}$, with the remainder forming $L_{\frac{n}{2}}$ (and attached to the complete graph portion). The lemma in this case gives $C(G) = O(n^3)$, which happens to be tight. This is because it takes $\Omega(n^3)$ time to get from $u$ to $v$. We can see this by the following analysis: for just the line graph, it should take $\Omega(n^2)$ steps, $\Omega(n)$ of which will be spent at $u$, since the nodes should approach a uniform distribution. However, if we are at $u$, there is a $\frac{2}{n}$ probability of leaving the clique, see we need to visit $u$ $\Omega(n)$ times to 'escape' back to the line graph. However, if we are in the clique portion it takes $\Omega(n)$ steps to get back to $u$. Thus each time we end up in $u$ from the line graph, we expect to take $\Omega(n^2)$ steps to get back into the line graph. Thus the expected number of steps is $\Omega(n^3)$. This illustrates that the number of edges doesn't always give the best estimate, since both this and the last example had $O(n^2)$ edges.




## 27.7 Testing s-t connectivity

Since any graph has at most $O(n^2)$ edges, the cover time is $O(n^3)$ for any graph. Thus we can use random walks to test for s-t connectivity using very smalls amounts of space. This problem can be solved using either BFS or DFS, but they use $\Omega(n)$ space. This bound on the cover time suggests a simple randomized algorithm for testing s-t connectivity. We just run a random walk starting at s for $2n^3$ steps and output 'connected' if we ever encounter $t$ on our walk. This algorithm will succeed with probability $\frac{1}{2}$, so if we desired gives an upper bound on the expected time of our algorithm, and allows us to use $\log n$ bits to store our current location in the walk, and $O(\log n)$

bits as a counter for our time in the walk. This shows that $s - t$ connectivity can be solved in randomized log space. It was recently shown that the procedure can be derandomized, while still still using logarithmic space.