CS 466/666: Algorithm Design and Analysis **University of Waterloo: Fall 2024**

Lecture 22

November 26, 2024

Instructor: Sepehr Assadi

Topics of this Lecture

1 Motivating Examples

As our final topic in the course, we will be studying *random walks*. Before we get to define them formally however, it is worth seeing two motivating examples.

1.1 A Randomized Algorithm for 2-SAT

Consider the 2-SAT problem: Given a 2-CNF Φ, i.e., a formula of m clauses on n variables of the form:

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_1) \wedge (\neg x_3 \vee \neg x_4) \ \cdots$$

decide if Φ is satisfiable or not. We design the following algorithm for this problem:

Algorithm 1.

1. Start with any arbitrary assignment $x \in \{0,1\}^n$ to the variables (say, the all-zero assignment).
2. While there is any unsatisfied clause:
   (a) Pick an unsatisfied clause C arbitrarily.
   (b) Pick *one* of the two variables in C *uniformly at random* and flip its value.

Of course as it is, this algorithm never terminates on unsatisfiable inputs, and we will fix that eventually. But the main question for now is how long does it take for this algorithm to terminate if we run it on a satisfiable formula Φ?

Since Φ is satisfiable, there exists at least one assignment $x^* \in \{0,1\}^n$ that satisfies it. Let $x^{(t)}$ be the assignment maintained by the algorithm at the beginning of its t-iteration of the while-loop. Define the

distance of $x^{(t)}$ and x^*, denoted by $dist(x^{(t)}, x^*)$, as the number of variables that are assigned different values between the two. Thus, $dist(x^{(t)}, x^*) \leqslant n$ always and if at any point $dist(x^{(t)}, x^*)$ becomes 0, the algorithm terminates[1]. As such, our goal is to understand the progression of $dist(x^{(t)}, x^*)$ through the time $t \geqslant 1$.

Fix any iteration $t \geqslant 1$. Suppose the algorithm picks clause C in this iteration. Since C was not satisfied, we know that at least one of the variables in C are assigned a different value in $x^{(t)}$ and x^* (and it is possible that the other variable has been assigned the same value). Hence, with probability at least half, we pick the different variable in $x^{(t)}$ and make it the same as x^* in $x^{(t+1)}$ which decreases the distance by 1, and with the remaining probability, we may even increase the distance by 1. As such, at any iteration $t \geqslant 1$,

- $dist(x^{(t+1)}, x^*) = dist(x^{(t)}, x^*) - 1$ with probability at least $1/2$;
- $dist(x^{(t+1)}, x^*) = dist(x^{(t)}, x^*) + 1$ with probability at most $1/2$.

Let us first use this naively. The implication of the above bounds for the expected distance is that

$$\mathbb{E}\left[dist(x^{(t+1)}, x^*)\right] \leqslant \frac{1}{2} \cdot \left(\mathbb{E}\left[dist(x^{(t)}, x^*)\right] - 1\right) + \frac{1}{2} \cdot \left(\mathbb{E}\left[dist(x^{(t)}, x^*)\right] + 1\right) = \mathbb{E}\left[dist(x^{(t)}, x^*)\right].$$

This means that *in expectation*, we are not increasing the distance but we may in fact keep it as it was throughout. Does this mean the algorithm never terminates because it is "not making any progress *in expectation*"? Of course not! This is one good example of a scenario where focusing solely on the expectation fails to capture the *real* behavior of the underlying variables. In particular, one can see that these variables also have a non-trivial *variance* and thus over time, we "expect" the value of $dist(x^{(t+1)}, x^*)$ to be quite different than $dist(x^{(1)}, x^*)$ – if it ever happens that this value is different by n, then we are definitely done! So, how many iterations do we need for this to happen?

A general way of handling such scenarios[2] is through *Random Walks*. Consider this process: we have $n+1$ *states* corresponding to the different values $dist(x^{(t)}, x^*)$ can take. There is a "particle" (or agent, or walker, or anything else you like to call it) that starts from one of these states. At each point in time, if the particle is currently at a state $i \notin \{0, n\}$, it goes to state $i+1$ with probability half and to state $i-1$ with probability half. If the particle is at n, it goes to state $n-1$ with probability 1 and if it is at 0, the process terminates. If we start at some arbitrary state, in expectation, how many steps it takes for the random walk to terminate, namely, the particle reaches state 0? What about with high probability?

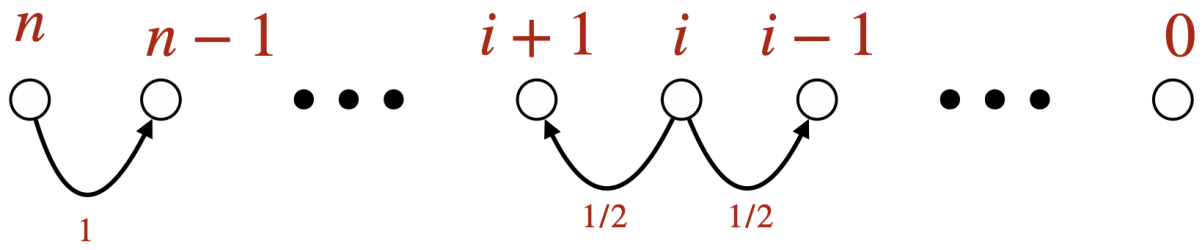

Figure 1: An illustration of the random walk process on the integer line.

Notice that this provides an *upper bound* on the number of iterations of our algorithm because in the algorithm, the probability of moving toward 0 is *at least* half and not exactly half (as in, in some cases, we will definitely reduce the distance regardless of which variable we pick to flip).

We will see later in this lecture how very simple tools in random walks—notions such as **hitting times** and **cover times**—allow us to infer this process finishes in $O(n^2)$ steps in expectation and in $O(n^2 \log n)$

[1] Notice that the algorithm may terminate even earlier if there are more than one satisfying assignment.

[2] Although this particular example has more direct solutions also based on directly calculating the probabilities; you may want to work that out on your own for practice.

steps with high probability. This means that Algorithm 1 finishes in $O(n^2 \log n)$ time with high probability on satisfiable inputs.

Thus, the way we can use this algorithm for deciding satisfiability of any input in to run it for $O(n^2 \log n)$ time and if it still did not find a satisfying assignment, we simply terminate and output the answer is not satisfiable. This gives us an $O(n^2 \log n)$ time algorithm that solves the 2-SAT problem with high probability.

1.2 Undirected s-t Connectivity (USTCONN) with Small Space

Given an undirected graph $G = (V, E)$ and two vertices s and t, are s and t connected? This problem is often stated as the USTCONN problem. Of course, we can solve this problem easily in $O(m+n)$ time using any standard graph search algorithm, say, DFS or BFS. But these standard graph search algorithms need to "remember" which vertices they have visited and as such they all need extra $O(n)$ space (in addition to the input) to work. But, what if we like to solve the problem with a much smaller space, say, only $O(\log n)$ bits – this means we can effectively only "remember" a constant number of vertices at a time and have to repeatedly go back and revisit the same vertices. Can we do anything non-trivial?

One can design the following super simple algorithm for this problem.

Algorithm 2.

1. Start with the vertex $v = s$.
2. While $v \neq t$:
   (a) Pick one of the neighbors of v *uniformly at random* and update v to be that vertex.

Again, this algorithm clearly never terminates if s and t are not in the same connected component; we shall fix that later easily. The main question is how long this algorithm takes to find t if s and t are inside the same connected component?

Let $v^{(k)}$ denote the value of the vertex v in the algorithm in the k-th iteration of the while-loop. We have $v^{(0)} = s$ and we will terminate the process at step k when $v^{(k)} = t$. In each step also, $v^{(k+1)}$ is a random neighbor of $v^{(k)}$. This process is called the (standard) *random walk* on an undirected graph. The question we are interested in is the expected number of steps it takes this random walk to reach t if we start from s. This is called the **hitting time** of t from s.

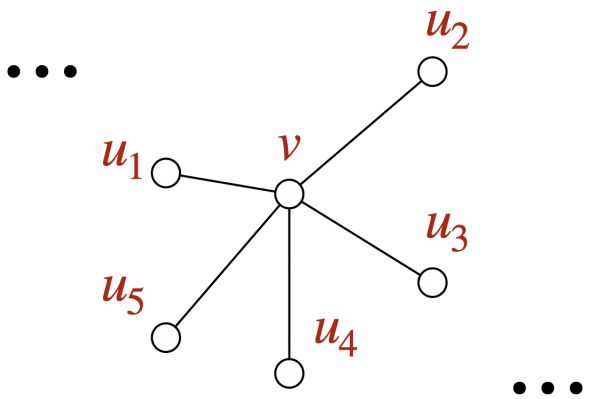

Figure 2: An illustration of the standard random walk on an undirected graph. If we are at this vertex v in some step, in the next step, we will be at one of its neighbors $\{u_1, \ldots, u_5\}$ chosen uniformly at random.

Again, we shall see later in the lecture that this quantity is at most $O(n^3)$ time in expectation and

$O(n^3 \log n)$ with high probability[3]. Using a similar trick as in for Algorithm 1 mentioned earlier, this gives an algorithm that in $O(n^3 \log n)$ time with high probability can decide if s is connected to t or not using only $O(\log n)$ space!

Remark. The above algorithm shows that USTCONN belongs to the complexity class RL, namely, problems that can be decided in logarithmic space and polynomial time using randomization. While this is already extremely space efficient, one can get an even stronger result. A celebrated result of Reingold [Rei05] proves that one can solve USTCONN in $O(\log n)$ space *deterministically*! This means that USTCONN actually belongs to the complexity class L.

2 Markov Chains

We can model the random walk processes described above, and much more, via *Markov chains*.

Definition 1. A (discrete-time) Markov chain over the **state space** Ω is a stochastic process that generates random variables $(X_0, X_1, X_2, \ldots, X_t, \ldots)$ supported on Ω such that for every $t \geqslant 0$,

$$\Pr\left(X_{t+1} = y \mid X_t = x, X_{t-1}, \ldots, X_0\right) = \Pr\left(X_{t+1} = y \mid X_t = x\right) := P(x, y);$$

in words, the probability distribution of the next step X_{t+1} only depends on X_t and not the "history" of these variables.

Thus, the entire Markov chain can be described by an $|\Omega| \times |\Omega|$ matrix P called the **transition matrix**: for every $x, y \in \Omega$, $P(x, y)$ denotes the probability that the next chosen state is y conditioned on the current state being x. Notice that P is a *stochastic* matrix meaning that it is non-negative and each row-sum is one.

We can model our previous examples via Markov chains as follows.

Example 1: 2-SAT. The underlying Markov chain there has state space $\Omega = \{0, 1, \ldots, n\}$, with the following transition matrix

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \ldots & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & \ldots & 0 \\ & & & \ddots & & \end{bmatrix},$$

i.e., $P(0, 1) = 1$, $P(i, i+1) = P(i, i-1)$ for every $0 < i < n$, and $P(n, n-1) = 1$ (we allow the chain to "move away" from state 0 also for technical reasons; of course, in our application, we only care about "hitting" the state 0 and thus transition out of this state is irrelevant for us.)

Example 2: Undirected random walk. The Markov chain there has state space V, namely, vertices of the graph. The transition matrix of the Markov chain is the following:

$$\forall u \in V \qquad P(u, v) := \frac{1}{\deg(u)} \quad \text{if } u \in N(v) \text{ and 0 otherwise.}$$

[3]You are encouraged to think about how you would prove this; in particular, why the "distance-based" approach as in the first application seems to completely fail here.

Definition 2. A **random walk** in a Markov chain P starts with some distribution for the random variable X_0 denoted by the row-vector $p^{(0)}$. Each step $t \geqslant 1$ of the walk then involves updating X_t from X_{t-1} according the transition matrix P. In the matrix form, this corresponds to updating

$$p^{(t+1)} \;=\; p^{(t)} \cdot P \;=\; p^{(0)} \cdot P^t.$$

Notice that the equality in the above equation holds because for every state $i \in \Omega$:

$$\begin{aligned}&\Pr\left(\text{probability of being at state } i \text{ in step } t+1\right)\\ &\quad= \sum_{j \in \Omega} \Pr\left(\text{probability of starting at state } j \text{ in step } t\right) \cdot \Pr\left(\text{going from } j \text{ to } i \text{ in step } t \mid \text{being at state } j\right)\end{aligned}$$

which is equivalent to

$$p_i^{(t+1)} = \sum_{j \in \Omega} p_j^{(t)} \cdot P_{j,i}$$

which in turn gives us the above equation.

Notice that in both our previous examples, our starting distribution/random variable had a *point mass*, i.e. had probability 1 on one of the states and zero elsewhere. However, we can consider general distributions over the states for the starting distribution.

2.1 Hitting Times and Stationary Distributions

The following definition is key for obtaining the aforementioned bounds we needed for our earlier algorithms.

Definition 3. For any pairs of states $i, j \in \Omega$ in a Markov chain P, the **hitting time** of i to j, denoted by $h_{i,j}$, is the expected length of the random walk starting from i until it visits j for the first time.

In order to be able to talk about hitting times, we need another key concepts in random walks. A key property of random walks in Markov chains is that they are "history independent" in the sense that

$$X_{t+1} \perp X_0, X_1, \ldots, X_{t-1} \mid X_t;$$

knowing the current state of the random walk is enough to predicate all its future states without knowing anything about its past. Can we push this further and say that for large enough $t \to \infty$, we get $X_t \perp X_0$ *unconditionally*? I.e., if we run the random walk "long enough", the distribution of the states "mixes" so well that it is no longer at all clear where we started?

Definition 4. A distribution π over the states is a **stationary distribution** for a Markov chain P iff

$$\pi = \pi \cdot P.$$

In words, this means that if the distribution of the current state is π, taking another step in the walk does not change the distribution at all.

Notice that once we are in a stationary distribution, it no longer matters where we started the walk, as the distribution of the walk remains the same from now on.

Examples. In the previous examples, the stationary distributions are:

- **2-SAT:** $\pi(0) = \pi(n) = \frac{1}{2n}$ and $\pi(i) = \frac{1}{n}$ for all $0 < i < n$.
- **Undirected random walk:** $\pi(v) = \deg(v)/2m$ for every $v \in V$.

In both cases, you are encouraged to check the equation $\pi = \pi \cdot P$ to convince yourself these are indeed the stationary distributions.

So, when can we hope for our walk to converge to a stationary distribution eventually? Notice that any transition matrix P defines a *directed* graph $G = (V, E)$ with vertices corresponding to the states and an edge (i, j) from any state vertex i to vertex j if $P_{i,j} > 0$, i.e., there is a non-zero probability of moving directly from state i to state j. There are two immediate conditions on the graph G that clearly disallow having a stationary distribution (see Figure 3).

- **Reducibility:** Suppose the graph G has more than one strongly connected component (SCC), where a SCC is a set of vertices where every pair can reach other. We call such a Markov chain **reducible**.

  In this case, depending on which state we start the random walk, there are some states that we may never reach.

- **Periodicity:** Suppose the graph G is such that there is a pair of states i and j such that the greatest common divisor of the set $\left\{t : P^t_{i,j} > 0\right\}$ is more than 1. We call such a Markov chain **periodic**.

  In this case, if we start the walk from state i, then we can only be in the state j at certain time steps, which means that the walk cannot converge to a stationary distribution (rather, it may oscillate between different distributions on states).

It turns out these are the only conditions that stops a Markov chain from converging to its stationary distributions.

Theorem 5 (**Fundamental Theorem of Markov Chains**). *If a Markov chain P is neither reducible nor periodic, then it has a unique and strictly positive stationary distribution π, satisfying $\pi = \pi \cdot P$.*

Moreover, $P^t(i, j) \to \pi_j$ when $t \to \infty$ for every states i and j. I.e., no matter which state we start the random walk from, the probability of being at a state j after t steps of the random walk, for $t \to \infty$, is π_j.

If we denote $N^t(i, j)$ as the number of times a random walk of length t starting from a state i is visiting a state j, we obtain that for all states i and j,

$$\lim_{t\to\infty} \frac{N^t(i, j)}{t} = \pi_j.$$

I.e., a sufficiently long walk visits the state j in π_j fraction of times.

Finally, for any state i, we have that hitting time of i to i is inverse of its stationary probability, i.e.,

$$h_{i,i} = \frac{1}{\pi_i}.$$

The property for hitting time holds even if P is only irreducible (i.e., it may even be periodic).

We will not be proving this theorem in this course but you can find its proof in most standard textbooks in statistics or probability theory. We only provide a quick intuition on why we expect $h_{i,i}$ to be $1/\pi_i$: consider running the walk for a long period of time and notice that the fraction of times i appears in this sequence is π_i; this in turn means that the average distance between occurrences of i should be $1/\pi_i$. Thus, the "average distance" between appearance of i in a long enough random walk should be $1/\pi_i$; but this average distance should also be the same as $h_{i,i}$. We emphasize that this is only an intuition and in no way should be considered a formal proof.

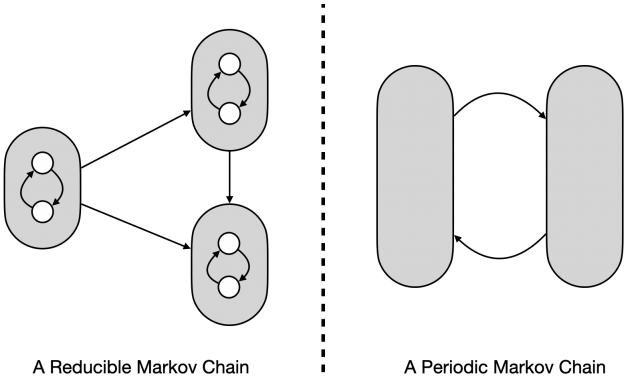

Figure 3: An illustration of reducibility and periodicity in random walks.

3 Random Walk on Undirected Graphs

Given the definitions earlier and the fundamental theorem of Markov chains, we can now start analyzing various other properties of the standard random walk on undirected graphs (which encompasses both our previous applications). The first one is to formally prove the stationary distribution.

Proposition 6. *In any undirected graph $G = (V, E)$, the stationary distribution π of the standard random walk is so that for every $v \in V$:*

$$\pi_v = \frac{\deg(v)}{2m}.$$

Proof. We need to prove $\pi = \pi \cdot P$ for the specific choice of π and P for the undirected random walk. Fix any vertex $v \in V$. We have,

$$(\pi \cdot P)_v = \sum_u \pi_u \cdot P_{u,v} = \sum_{u \in N(v)} \frac{\deg(u)}{2m} \cdot \frac{1}{\deg(u)} = \frac{\deg(v)}{2m} = \pi_v.$$

Given that $\sum_v \pi_v = 1$ also, this concludes the proof. □

A corollary of this and Theorem 5 is that in a random walk in an undirected graph, it takes in expectation $h_{v,v} = 2m/\deg(v)$ time for a walk starting from a vertex v to visit v again. A more useful property we are typically interested in is on how long does it take for v to visit another vertex u, i.e., hitting time of $h_{v,u}$ (in both our applications, this is a type of question we are interested in). It turns out that this quantity on its own does not have an "easy" solution. However, we will see how we can find "good" upper bounds on it.

Proposition 7. *In a standard random walk on any undirected graph $G = (V, E)$, for every* edge $(u, v) \in E$,

$$h_{u,v} + h_{v,u} \leqslant 2m.$$

Proof. The proof of this result involves two separate parts.

Part one. Fix any edge $(u, v) \in E$ in the graph. We claim that:

> *The expected number of steps that the random walk takes before it passes the edge (u, v) in this direction, assuming it just passed the edge (u, v) is $2m$.*

Consider a Markov chain P' on $2m$ states each corresponding to one direction of each edge in G, where in P' there is an edge from the state (u, v) to (v, w) with transition probability $1/\deg(v)$. Notice that a

random walk in the original graph corresponds exactly to a random walk in this Markov chain (the new chain explicitly keep track of traversed edges in the walk but transition probabilities are the same).

The stationary distribution π' of P' is uniform over all the states, because for any state (u,v),

$$(\pi' \cdot P')_{(u,v)} = \sum_{(w,z)} \pi'_{(w,z)} \cdot P_{(w,z),(u,v)} = \sum_{(w,u)\in E} \pi'_{(w,u)} \cdot P_{(w,u),(u,v)} = \deg(u) \cdot \frac{1}{2m} \cdot \frac{1}{\deg(u)} = \frac{1}{2m} = \pi_{(u,v)}.$$

By Theorem 5, we have that

$$h_{(u,v),(u,v)} = \frac{1}{\pi_{(u,v)}} = 2m,$$

implying that the hitting time of (u,v) to (u,v) in P' is $2m$. By the equivalence of this random walk on P' and the standard random walk in the original graph, we obtain the desired claim.

Part two. Now, consider the edge $(u,v) \in E$. We have that

$$\begin{aligned} h_{u,v} + h_{v,u} &\leqslant \mathbb{E}\,[\text{length of the walk that starts from } u \text{ and goes to } v \text{ and then takes the edge } (u,v)] \\ &\qquad\qquad (\text{as we consider a special way back from } v \text{ to } u \text{ only via the edge } (v,u)) \\ &= 2m, \end{aligned}$$

where in the equality we used the claim from the first part, plus the fact that random walk is history independent: whether we have taken the edge (v,u) to get to u or any other edge to be at u, the length of the walk before it sees the edge (v,u) in expectation is the same (as the walk starts from the vertex u any way). This proves the statement. □

Let us emphasize that Proposition 7 only holds for pairs of vertices u, v that are connected by an edge in G; for other vertices, the value of $h_{u,v} + h_{v,u}$ can be much larger. However, the following result now allows us to bound those as well.

Proposition 8. *For any pairs of vertices s and t in an undirected graph $G = (V,E)$,*

$$h_{s,t} \leqslant 2m \cdot \text{dist}_G(u,v),$$

where $\text{dist}(s,t)$ denotes the length of the shortest path between s and t in G.

Proof. Consider the shortest path from in G from s to t and denote it by $s = u_0, u_1, u_2, \ldots, u_k = t$. We have,

$$h_{s,t} \leqslant h_{u_0,u_1} + h_{u_1,u_2} + h_{u_2,u_3} + \ldots + h_{u_{k-1},t}$$

because one way of going from s to t is to go from s to u_1, then from u_1 to u_2, and so on and so forth until we end up at t (and by linearity of expectation). The statement now follows from Proposition 7 as each $h_{u_i,u_{i+1}} \leqslant 2m$ as (u_i, u_{i+1}) is an edge, and since the length of the path is $\text{dist}_G(u,v)$. □

Proposition 8 is enough for both our applications from earlier in the lecture. In both the 2-SAT problem and the USTCONN problem, we are performing a random walk on an undirected graph and thus $h_{s,t} = O(n^2)$ in the first one (where s is any state the 2-SAT algorithm starts from and t is the state 0) and $h_{s,t} = O(mn)$ time in the second one. Nevertheless, we can prove an even stronger statement.

Definition 9. For any undirected graph $G = (V,E)$, we define the **cover time** of a vertex $v \in V$, denoted by C_v, as the expected length of the random walk that starts from v, visits *all* vertices, and gets back to v, i.e., cover the entire graph.

Proposition 10. *For any connected graph $G = (V,E)$ and any vertex $v \in V$, the cover time of v in G is*

$$C_v \leqslant 2m \cdot (n-1).$$

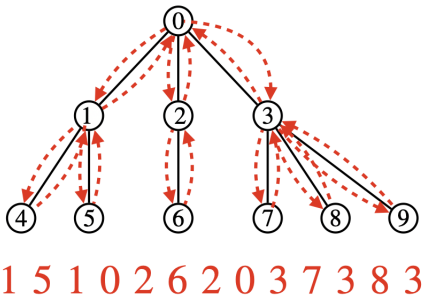

Figure 4: An example of a tree T with solid (black) edges and its DFS traversal with dashed (red) edges and the corresponding output sequence.

Proof. Pick a spanning tree T of G rooted at v. Consider the DFS traversal of T starting from v and outputting each vertex whenever we enter it and whenever we leave it. See Figure 4 for an illustration.

Let $v = u_0, u_1, u_2, \ldots, u_k = v$ be the output vertices. Since this tree visits all vertices, we certainly have

$$C_v \leqslant \sum_i h_{u_i,u_{i+1}} = \sum_{e=(u,w)\in T} h_{u,w} + h_{w,u},$$

because we traverse each edge exactly once in each direction in the traversal. Since edges in T are also in G, applying Proposition 7 to each term of the sum and noting that T has $n-1$ edges implies

$$C_v \leqslant (n-1) \cdot 2m,$$

proving the statement. □

Proposition 10 implies that where we start a random walk in our graph, after $O(mn)$ steps in expectation, we have visited all vertices. We can extend this to a high probability result as follows. Suppose we run a random walk for $4m \cdot (n-1)$ steps from any vertex v. By Markov bound, the probability that the walk has not visited all vertices is at most $1/2$. But, no matter which vertex we are in currently, we can say the same bound also holds for the next $4m \cdot (n-1)$ steps, i.e., again, the probability that this part does not visit all vertices is at most $1/2$. Thus, after $2 \log n$ of such "batches" of random walks, the probability that we have not visited all vertices is at most

$$\left(\frac{1}{2}\right)^{2\log n} = \frac{1}{n^2}.$$

But this is equivalent with running a random walk of length $8m \cdot (n-1) \cdot \log n$. Hence, we obtain that a random walk of length $O(mn \log n)$ starting from any vertex with high probability visits all vertices.

This concludes our introduction to random walks and Markov chains.

References

[Rei05] Omer Reingold. Undirected st-connectivity in log-space. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 376–385. ACM, 2005. 4