15.1 Introduction

As we have seen in previous lectures, randomization can be a useful tool for developing simple and efficient algorithms. So far, most of these algorithms have used independent coin tosses to generate randomness. In this lecture we will look at a different type of randomized algorithms whose basic function is to traverse a fixed graph randomly.

Definition 15.1.1 *A* **random walk** *is a process for traversing a graph where at every step we follow an outgoing edge chosen uniformly at random. A* **Markov chain** *is similar except the outgoing edge is chosen according to an arbitrary fixed distribution.*

Given a random walk or a Markov chain we are usually interested in two things:

- How quickly can we reach a particular node; How quickly can we cover the whole graph?
- How quickly does our position in the graph become "random"?

In this lecture we will mostly focus on random walks on undirected graphs and in the first set of questions.

15.1.1 Uses and examples of random walks

One use of random walks and Markov chains is to sample from a distribution over a large universe. Informally, we set up a graph over the universe such that if we perform a long random walk over the graph, the distribution of our position approaches the distribution we want to sample from.

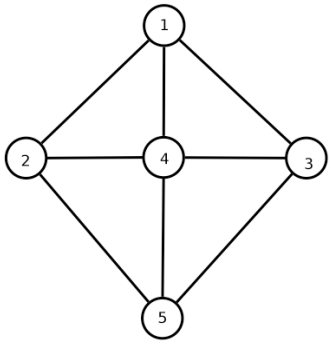
While random walks and Markov chains are useful algorithmic techniques, they are also useful in analyzing some natural processes. Consider, for example, the following betting game: a player bets \$1, and either loses it or wins an addition dollar with probability $\frac{1}{2}$. Since the probability 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 example is shuffling a deck of cards. One possible way of drawing a permutation u.a.r. is to start at an arbitrary permutation and to 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 over permutations (a perfectly shuffled deck).

15.2 Properties of random walks

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, $P_{uv} = \Pr[\text{going from u to v, given that we are at u}]$. 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).

Below is an example of a graph and the transition matrix for the random walk on this graph.



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

Stationary distribution. One of the properties of random walks that we are interested in is the distribution of our position in the graph if we run a random walk for an infinite number of steps. Formally, if we start with a distribution π on nodes in the graph, where π is an $n \times 1$ vector, and take a step in the random walk, then our final distribution over nodes is given by $P^T\pi$ where P is the transition matrix. Then, starting at π and running the random walk for an infinite number of steps gives us the following distribution on nodes in the limit:

$$\lim_{t\to\infty} (P^T)^t \pi$$

In general this distribution may depend on the position in the graph (or a distribution over it) that we start the random walk, and some times the limit may not even exist. However, under certain conditions, the limiting distribution always exists and is independent of the starting position (the details are beyond the scope of this class). When these conditions hold, this limiting distribution is identical to one that satisfies the following equation:

$$\pi^* = P^T\pi^*$$

This π^* is called the *stationary distribution* of the Markov chain.

Stationary distributions of random walks have special structure. The following two lemmas characterize these distributions. In the following discussion we will consider a graph $G = (V, E)$, with $n = |V|$ and $m = |E|$. Let d_u denote the degree of vertex u.

Lemma 15.2.1 *$\pi_v = \frac{d_v}{2m}$ is a stationary distribution for a random walk over G.*

Proof: Suppose that we start with the distribution π and take a single step in the graph G. Then the probability that we end up at node u is given by:

$$(P^T \cdot \pi)_u = \sum_v P_{vu}\pi_v \tag{15.2.1}$$

$$= \sum_{v:(v,u)\in E} \frac{d_v}{2m}\frac{1}{d_v} \tag{15.2.2}$$

$$= \sum_{v:(v,u)\in E} \frac{1}{2m} = \pi_u \tag{15.2.3}$$

where the second line follows because $P_{vu} = \frac{d_v}{2m}$ if $(v, u) \in E$ and 0 otherwise. ■

Another equivalent way of thinking about random walks is to consider the sequence of edges that the random walk follows, rather than the sequence of nodes that it visits. For example, if the random walk visits nodes v_1, v_2, v_3, v_4 in that order, then it visits the edges $(v_1 \to v_2), (v_2 \to v_3), (v_3 \to v_4)$ in that order. This process of visiting edges is a Markov chain over directed edges where each undirected edge (u, v) has two directed copies $(u \to v)$ and $(v \to u)$. Now we can ask what is the stationary distribution over directed edges in this Markov chain. It turns out that the uniform distribution over edges is a stationary distribution, that is, $\pi^*_{u\to v} = \frac{1}{2m}\ \forall (u \to v) \in E$. 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 15.2.2 *The stationary distribution induced on the edges of an undirected graph by the random walk on that graph is uniform.*

Hitting time and commute time. As mentioned earlier, one of the parameters of a random walk that we are interested in is the amount of time it takes to get from one place to another in the graph. We therefore define the following three quantities.

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

We now present a bound on the commute time. The proof below is just a sketch and certain technical details are omitted. However, we will shortly give a different formal proof of this lemma as well.

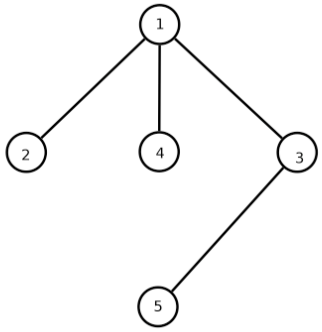
Lemma 15.2.3 *$\forall (u, v) : (u, v) \in E$, we have $C_{uv} \leq 2m$.*

Proof Sketch: 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 $(v \to u)$. This is because, one way of going from u to v and back to u is to first visit u after following the edge $(v \to u)$, then take an arbitrary path from v to u and then follow the edge $(v \to u)$ back to u. (The commute time could of course be much shorter, but this gives an upper bound.) The expected length of the gap between consecutive occurences of the directed edge 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 over directed edges 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 $\frac{t}{\frac{t}{2m}} = 2m$. ■

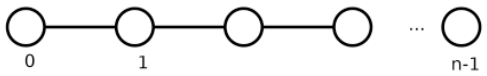
Cover time. If we had a bound on the commute time for all pairs (u, v) (call this bound x), we could get a bound (in expectation) on the cover time by running the random walk for $x * n$ steps. Unfortunately, the bound given by lemma 15.2.3 is only valid for pairs (u, v) where there is an edge between u and v. However, we can still come up with a different method for bounding the cover time.

Lemma 15.2.4 $C(G) \leq 2m(n-1)$

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 $C(G) \leq h_{21} + h_{14} + h_{41} + h_{13} + h_{35} + h_{53} + h_{31} + h_{12}$. ■



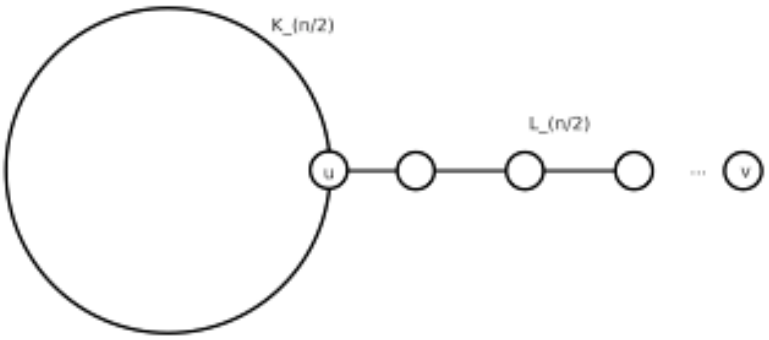
It turns out that for some graphs, the bound given in lemma 15.2.4 is tight. One example of this is the line graph with n vertices, depicted below (L_n). According to the lemma, $C(G) \leq 2(n-1)^2 = O(n^2)$. Also, we can note that $h_{1,0} \leq 2(n-1) - 1$, since $h_{0,1} = 1$, and $C_{uv} = h_{uv} + h_{vu}$ by linearity of expectation. We will show below 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 (uniformly at random) in one step, the problem of visiting all nodes can be viewed as an instance of the coupon collector problem (see addendum at the end of this lecture). This gives us a bound of $O(n \log n)$.

Another example where the bound is not tight is a star graph on n vertices. This graph has $n-1$ edges, and so the bound implies that the cover time on this graph is at most $2(n-1)^2$. However, once again appealing to the coupon collector problem, we see that the cover time on this graph is no more than $2(n-1)H_{n-1} = O(n \log n)$.

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 in a graph alone doesn't always give a good estimate of the cover time of the graph, since both this and the complete graph example have $\Theta(n^2)$ edges but very different cover times.



15.3 An application: testing s-t connectivity

A simple application of random walks is to test for connectivity in undirected graphs. In particular, suppose that we have a graph with two special nodes s and t and we wish to determine whether there is a path between the two nodes. We can do this easily using BFS, DFS, or any other graph traversal technique. However, these approaches use a linear amount of extra space (for example, to mark nodes that have been visited). Can we perform this task using lesser space?

Random walks give a solution. Note that since any graph has at most $n^2/2$ edges, the cover time is at most n^3 for any graph. Thus we run a random walk starting at s for $2n^3$ steps and output "connected" if we ever encounter t on our walk. Then, if s and t are connected, with probability $1/2$ we would have found t. If we repeat for $\log n$ times we can get the error probability down to $1/n$. We use $\log n$ bits to store our current position in the graph, 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.

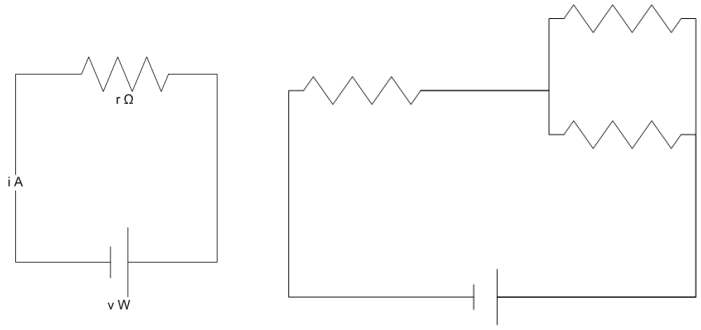
15.4 Bounding cover time through resistive networks

We will now describe a different technique for bounding commute and cover times based on resistive networks. Recall the laws governing electrical circuits. The three principle quantities we are interested in are voltage, V, current, i, and resistance r. The following two laws govern these quantities:

Definition 15.4.1 Ohm's Law: $V = ir$

Definition 15.4.2 Kirchoff's Law: *At any junction, $i_{in} = i_{out}$, that is, current is conserved.*

Here are some examples of circuits.



One application of these laws is that they allow us to compute the "effective resistance" of a circuit. The effective resistance is equal to the voltage drop across the circuit when a unit current flows through it. For example, for resistors in series the net resistance is the sum of the individual resistances, $r_{net} = \sum_{r \text{ in series}} r$. Similarly for resistors in parallel the multiplicative inverse of the net resistance is the sum of the multiplicative inverses of each individual resistor, $\frac{1}{r_{net}} = \sum_{r \text{ in parallel}} \frac{1}{r}$.

The relationship between commute time and effective resistances

Consider an undirected unweighted graph G. Replace every edge by a 1Ω resistor. Let R_{uv} be the effective resistance between nodes u and v in this circuit. The following lemma relates the commute time in the graph to the effective resistance.

Lemma 15.4.3 $C_{uv} = 2mR_{uv}$

Proof:

The proof of Lemma 15.4.3 will be shown by considering two schemes for applying voltages in resistive networks and then showing that the combination of the two schemes show the lemma.

Part 1. We will analyze what would happen if we connect u to ground, then apply a current to each other vertex w of amount d_w amps. (d_w is the degree of w.) The amount of current that flows into the ground at v is $2m - d_u$, since each edge contributes one amp at each end. Let ϕ_w be the voltage at node w.

Consider each neighbor w' of w. There is a 1Ω resistor going between them. By Ohm's law, the current across this resistor is equal to the voltage drop from w to w', which is just $\phi_w - \phi_{w'}$. Look at the sum of this quantity across all of w's neighbors:

$$d_w = \sum_{w':(w,w')\in E} (\phi_w - \phi_{w'}) = d_w\phi_w - \sum_{w':(w,w')\in E} \phi_{w'}$$

Rearranging:

$$\phi_w = 1 + \frac{1}{d_w} \sum_{w':(w,w')\in E} \phi_{w'} \tag{15.4.4}$$

At this point, we will take a step back from the interpretation of the graph as a circuit. Consider the hitting time h_{wu} in terms of the hitting time of w's neighbors, $h_{w'u}$. In a random walk from w to u, we will take one step to a w' (distributed with probability $1/d_w$ to each w'), then try to get from w' to u. Thus we can write h_{wu} as:

$$h_{wu} = 1 + \frac{1}{d_w} \sum_{w':(w,w')\in E} h_{w'u} \tag{15.4.5}$$

Note that the system of equations (15.4.5) is identical to the system of equations (15.4.4)! Both hitting times and voltages are solutions to this system of equations. So as long as these equations have a unique solution, $h_{wu} = \phi_w$. We will argue that this is the case. The voltage at a node is one more than the average voltage of its neighbors. Consider two solutions $\phi^{(1)}$ and $\phi^{(2)}$. Look at the vertex w where $\phi_w^{(1)} - \phi_w^{(2)}$ is largest. Then all of the neighbors of w must also have the same large difference in the two solutions. Then, assuming that the graph is connected, $\phi_u^{(1)}$ and $\phi_u^{(2)}$ must have the same difference. However, by assumption, $\phi_u^1 = \phi_u^2 = 0$, so we get a contradiction.

Therefore, for all w, $h_{w,u} = \phi_w$.

Part 2. We will now analyze what happens with a different application of current. Instead of applying current everywhere and drawing from u, we will apply current at v and draw from everywhere else.

We are going to apply $2m - d_v$ amps at v, and pull d_w amps for all $w \neq v$. (We continue to keep u grounded.) Let the voltage at node w under this setup be ϕ'_w.

Through a very similar argument, $h_{wv} = \phi'_v - \phi'_w$. Thus $h_{uv} = \phi'_v - 0 = \phi'_v$.

Part 3. We will now combine the conclusions of the two previous parts. At each node w, apply $\phi_w + \phi'_w$ volts. We aren't changing resistances, so currents also add. This means that each w ($\neq u$ and $\neq v$) has no current flowing into or out of it, and the only nodes with current entering or exiting are u and v.

At u, $2m - d_u$ amps were exiting during part 1, and d_u amps were exiting during part 2, which means that now $2m$ amps are exiting. By a similar argument (and conservation of current), $2m$ amps are also entering v.

Thus the voltage drop from u to v is given by Ohm's law:

$$(\phi_u + \phi'_u) - 0 = R_{uv} \cdot 2m$$

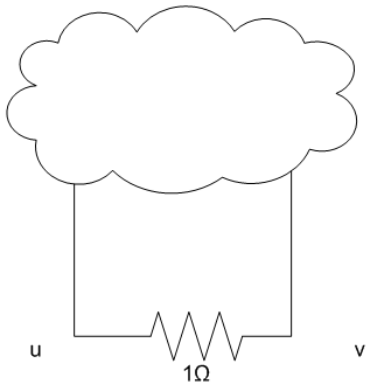
But $\phi_v = h_{vu}$ and $\phi'_v = h_{uv}$, so that gives us our final goal:

$$h_{uv} + h_{vu} = C_{uv} = 2mR_{uv}$$

■

Application of the resistance method

A couple of the formulas we developed previously can be re-derived easily using lemma 15.4.3. Lemma 15.2.3 says that, for any nodes u and v, if there is an edge $(u, v) \in E$, then $C_{uv} \leq 2m$. This statement follows immediately by noting that $R_{uv} \leq 1\Omega$. If a 1Ω resistor is connected in parallel with another circuit (for instance, see the following figure), the effective resistance R_{uv} is less than the minimum of the resistor and the rest of the circuit.



Furthermore, we previously showed (in lemma 15.2.4) that $C(G) \leq 2m(n-1)$. We can now develop a tighter bound:

Theorem 15.4.4 *Let $R(G) = \max_{u,v \in V} R_{u,v}$ be the maximum resistance between any two points. Then $mR(G) \leq C(G) \leq mR(G)2e^3 \ln n + n$.*

Proof: The lower bound is fairly easy to argue. Consider a pair of nodes, (u, v), that satisfy $R_{uv} = R(G)$. Then $\max\{h_{uv}, h_{vu}\} \geq C_{uv}/2$ because either h_{uv} or h_{vu} makes up at least half of the commute time. Lemma 15.4.3 and the above inequality shows the lower bound.

To show the upper bound on $C(G)$, we proceed as follows. Consider running a random walk over G starting from node u. Run the random walk for $2e^3 mR(G)$ steps. For some vertex v, the chance that we have not seen v is $1/e^3$. We know that from 15.4.3 the hitting time from any u to v is at most $2mR(G)$. From Markov's inequality:

$$\begin{aligned}
\mathbf{Pr}\big[\#\text{ of steps it takes to go from } u \text{ to } v \geq 2e^3 mR(G)\big] &\leq \frac{\mathbf{E}[\#\text{ of steps it takes to go from } u \text{ to } v]}{2e^3 mR(G)} \\
&\leq \frac{2mR(G)}{2e^3 mR(G)} \\
&\leq \frac{1}{e^3}
\end{aligned}$$

(Note that this holds for any starting node $u \in V$.)

If we perform this process $\ln n$ times — that is, we perform $\ln n$ random walks starting from u ending at u' the probability that we have not seen v on *any* of the walks is $(1/e^3)^{\ln n} = 1/n^3$. Because $h_{uv} \leq 1/e^3$ for all u, we can begin each random walk at the last node of the previous walk. By union bound, the chance that there exists a node that we have not visited is $1/n^2$.

If we have still not seen all the nodes, then we can use the algorithm developed last time (generating a spanning tree then walking it) to cover the graph in an expected time of $2n(m-1) \leq 2n^3$.

Call the first half of the algorithm (the $\ln n$ random walks) the "goalless portion" of the algorithm, and the second half the "spanning tree portion" of the algorithm.

Putting this together, the expected time to cover the graph is:

$$\begin{aligned}
C(G) &\leq \mathbf{Pr}[\text{goalless portion reaches all nodes}] \cdot (\text{time of goalless portion}) \\
&\quad + \mathbf{Pr}[\text{goalless portion omits nodes}] \cdot (\text{time of spanning tree portion}) \\
&\leq \left(1 - \frac{1}{n^2}\right) \cdot (2e^3 mR(G) \cdot \ln n) + (1/n^2) \cdot (n^3) \\
&\leq 2e^3 mR(G) \ln n + n
\end{aligned}$$

■

Examples

Line graphs. Above we noted that that $C(L_n) = O(n^2)$ for line graphs. We now have the tools to show that this bound is tight. Consider u at one end of the graph and v at the other; then $R_{uv} = n - 1$, so by lemma 15.4.3, $C_{uv} = 2mR_{uv} = 2m(n-1)$, which is exactly what the previous bound gave us.

Lollipop graphs. For lollipop graphs we previous argued (informally) that $C(G) = \Omega(n^3)$ for lollipop graphs on n nodes. We will now confirm this. Consider u at the intersection of the

two sections of the graph, and v at the other end of the line segment. Then $R_{uv} = n/2$, so $C_{uv} = 2m\frac{n}{2} = 2\Theta(n^2)\frac{n}{2} = \Theta(n^3)$. Thus again our previous big-O bound was tight.

15.5 Another application: 2-SAT

We conclude with an example of using random walks to solve a concrete problem. The 2-SAT problem consists of finding a satisfying assignment to a 2-CNF formula. That is, the formula takes the form of $(x_1 \vee x_2) \wedge (x_3 \vee \overline{x_1}) \wedge \ldots$. Let n be the number of variables.

The algorithm works as follows:

1. Begin with an arbitrary assignment.
2. If the formula is satisfied, halt and output satisfiable.
3. If the number of iterations exceeds n^3, halt and output unsatisfiable.
4. Pick any unsatisfied clause. Pick one of the variables in that clause u.a.r. and invert its value.
5. Return to step 2.

Assume that the formula is satisfiable. Each step of this algorithm is linear in the length of the formula, so we just need to figure out how many iterations we expect to have before finding a satisfying assignment.

Consider any satisfying assignment for the formula. We will measure progress by counting how far from this satisfying assignment we are in terms of how many variables we differ from. In the worst case, we start by being "wrong" on all the n variables, and the algorithm terminates (no later than) when the number of "wrong" variables goes down to 0. The algorithm can thus be viewed as performing a random walk on a line graph with $n+1$ nodes. Each node corresponds to the number of variables in the assignment that differ from the satisfying assignment we chose. When we invert some variable x_i, either we change it from being correct to incorrect and we move one node away from 0, or we change it from being incorrect to being correct and move one step closer to the 0 node.

However, there is one problem with this analogy, which is in random walks on line graphs we assume that we go left or right with equal probability. Thus we need to argue that the probability of going left or right is $\frac{1}{2}$. In the case where the algorithm chooses a clause with both a correct and an incorrect variable, the chances in fact do work out to be $\frac{1}{2}$ in each direction. In the case where the algorithm chooses a clause where both variables are incorrect, it will *always* move towards the 0 node. Thus the probability the algorithm moves toward 0 is at least $\frac{1}{2}$. While this is different from the random walk we studied previously, it only biases the results in favor of shorter running times.

Thus the probability of the random walk proceeding from i to $i-1$, and hence closer to a satisfying assignment, is at least 1/2. Because of this, we can use the value of the hitting time we developed for line graphs earlier. Hence the number of iterations we need to perform in expectation before finding a satisfying assignment is $< 2n^2$. Running for n^3 steps gives us a high probability result via Markov's inequality.