CSC 5450 Randomness and Computation

Week 6 : Random Walk

Outline In this and next week we will study random walk in graphs.

Given a graph, the walk starts from a vertex, at each time step the walk moves to a uniformly random neighbor of the current vertex, and repeat.

Some of the basic mathematical questions are :

(1) What is the limiting distribution of the random walk ? (stationary distribution)

(2) How long does it take before the walk approaches the limiting distribution ? (mixing time)

(3) Starting from a vertex s, what is the expected number of steps to first reach t ? (hitting time)

(4) How long does it take to reach every vertex at least once ? (cover time)

Today we will mostly focus on question (3) and its applications, and question (1) and (4).

Next week we will focus on question (2).

Plan

- Random walk on a line : 2-SAT, 3-SAT
- Random walk on directed graphs : stationary distribution, pagerank, perfect matching
- Random walk on undirected graphs : electrical network, hitting time, cover time

---

2-SAT [MU 7.1.1]

Given a formula with two variables in each clause, there is a deterministic linear time algorithm to determine if there is a satisfying assignment or not.

We consider an algorithm for 2-SAT based on random walk (known as WalkSAT) :

```
random walk 2-SAT
  ① start from an arbitrary assignment
  ② Repeat up to $200n^2$ times, terminate if all clauses are satisfied
      ⓐ Choose an arbitrary clause that is not satisfied
      ⓑ choose a random variable in the clause and switch its value
  ③ Return a satisfying assignment if it is found ; otherwise return "unsatisfiable".
```

Clearly the algorithm is correct when it returns a satisfying assignment .

We thus focus on the case that there is a satisfying assignment but the algorithm didn't find it.

How to analyze the algorithm as analyzing random walk? What random variables do we consider?

A natural choice is to consider the number of clauses satisfied by the current assignment, but it is not easy to analyze as it can vary considerably from step to step.

Instead, we consider how "close" is the current assignment to a satisfying assignment (assuming it exists).

Let S be a satisfying assignment.

Let A_i be the assignment in the i-th step of the algorithm.

Let X_i be the number of variables that have the same value in A_i and S.

If $X_i = n$, then $A_i = S$ and we win.

We will keep track of X_i when a satisfying assignment is not found yet.

First, if $X_i = 0$, then $\Pr(X_{i+1} = 1 \mid X_i = 0) = 1$.

Suppose $1 \le X_i \le n-1$. In an unsatisfied clause, since S is a satisfying assignment, there must be a variable in the clause that has different values in A_i and S.

Since we pick a random variable in that clause, with prob $\ge \frac{1}{2}$ we pick such a variable and "correct" its value.

Therefore, $\Pr(X_{i+1} = j+1 \mid X_i = j) \ge \frac{1}{2}$ and $\Pr(X_{i+1} = j-1 \mid X_i = j) \le \frac{1}{2}$.

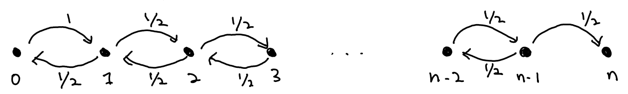
To model it as a random walk problem, we consider a pessimistic version of the stochastic process.

$$\Pr(Y_{i+1} = 1 \mid Y_i = 0) = 1$$
$$\Pr(Y_{i+1} = j+1 \mid Y_i = j) = 1/2$$
$$\Pr(Y_{i+1} = j-1 \mid Y_i = j) = 1/2$$

Clearly, the expected time for Y to reach n is not faster than X to reach n.

We will give an upper bound on the expected time for Y to reach n.

This can be thought of as a random walk on a line.



Let h_j be the expected number of steps to reach n when starting from j.

Then $h_j = \frac{1}{2}(h_{j-1}+1) + \frac{1}{2}(h_{j+1}+1) = \frac{h_{j-1}+h_{j+1}}{2} + 1 \iff h_j - h_{j+1} = h_{j-1} - h_j + 2$

To compute h_j we just need to solve the following system of linear equations.

$$h_n = 0$$
$$h_j - h_{j+1} = h_{j-1} - h_j + 2$$
$$h_0 - h_1 = 1 \quad (\text{because } h_0 = h_1 + 1)$$

By induction, we see that $h_j - h_{j+1} = 2j+1$.

We want to determine $h_0 = h_0 - h_n = \sum_{i=0}^{n-1} (h_i - h_{i+1}) = \sum_{i=0}^{n-1} (2i+1) = 2\left(\frac{(n-1)\cdot n}{2}\right) + n = n^2$. $\square$

So, if we run the algorithm for $2n^2$ steps, by Markov's inequality, the probability of not finding a satisfying assignment (if there exists one) is at most $1/2$.

Therefore, by running for $200n^2$ steps, the failure probability is at most $1/2^{100}$.

Clearly, the algorithm can be implemented in polynomial time.

3-SAT (MU 7.1.2)

The same approach can be applied for 3-SAT, giving a faster exponential time algorithm for it.

The main difference is the probability of "moving up" is only $1/3$.

Using the same notation,

$$h_n = 0$$

$$h_j = \frac{2h_{j-1}}{3} + \frac{h_{j+1}}{3} + 1 \iff h_j - h_{j+1} = 2(h_{j-1} - h_j) + 3$$

$$h_0 - h_1 = 1$$

By induction, $h_j - h_{j+1} = 2^{j+2} - 3$. Summing up as before, $h_j = 2^{n+2} - 2^{j+2} - 3(n-j)$.

In particular $h_0 = 2^{n+2} - 3n - 4$, which is more than the brute-force algorithm trying all 2^n assignments.

To improve it there are two observations:

① We can find a better initial assignment. In particular, if we pick a random initial assignment, then the expected number of matches with S is $n/2$.
(this alone is not enough, $h_{n/2}$ is not much smaller than h_0.)

② Since the random walk drifts toward 0, if the algorithm runs longer, then it is more likely that it is well below $n/2$. In this case, we should restart from a random initial assignment instead of keep running it.

These ideas lead to the fastest algorithm for 3-SAT known some time ago.

Schöning 3-SAT algorithm

① Repeat up to N times, terminating if all clauses are satisfied

ⓐ Start with a <u>random</u> initial assignment

ⓑ Repeat up to <u>3n</u> steps, terminating if all the clauses are satisfied

- switch the value of a random variable in an unsatisfied clause

② Return a satisfying assignment if it is found; otherwise return "unsatisfiable".

Suppose we start with an (unsatisfied) assignment with j mismatches with S.

In $3j$ steps, it reaches n if there are j steps "down" and $2j$ steps "up", which happens with probability $\binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j}$

This is a lower bound on the probability of reaching n in $3n$ steps.

To have a good estimate of it we use Stirling's formula:

Stirling's formula $\quad \sqrt{2\pi m}\left(\frac{m}{e}\right)^m \le m! \le 2\sqrt{2\pi m}\left(\frac{m}{e}\right)^m$

Using it, $\binom{3j}{j} = \frac{(3j)!}{(2j)!\,(j)!} \ge \frac{\sqrt{2\pi(3j)}}{4\sqrt{2\pi(2j)}\sqrt{2\pi(j)}}\left(\frac{3j}{e}\right)^{3j}\left(\frac{e}{2j}\right)^{2j}\left(\frac{e}{j}\right)^j$

$= \frac{\sqrt{3}}{8\sqrt{\pi}\sqrt{j}}\left(\frac{27}{4}\right)^j = \frac{c}{\sqrt{j}}\left(\frac{27}{4}\right)^j$ for $c = \frac{\sqrt{3}}{8\sqrt{\pi}}$ (a constant)

Therefore, starting with an assignment with j mismatches with S, the probability of reaching n is at least $\binom{3j}{j}\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j} \ge \frac{c}{\sqrt{j}}\left(\frac{27}{4}\right)^j\left(\frac{2}{3}\right)^j\left(\frac{1}{3}\right)^{2j} = \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j$

So, the probability of the algorithm to find a satisfying assignment in $3n$ steps is

$$\ge \sum_{j=0}^{n} \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j \cdot \Pr(\text{a random assignment has } j \text{ mismatches with } S)$$

$$= \sum_{j=0}^{n} \frac{c}{\sqrt{j}}\left(\frac{1}{2}\right)^j \cdot \binom{n}{j}\left(\frac{1}{2}\right)^n$$

$$\ge \frac{c}{\sqrt{n}}\left(\frac{1}{2}\right)^n \sum_{j=0}^{n}\binom{n}{j}\left(\frac{1}{2}\right)^j$$

$$= \frac{c}{\sqrt{n}}\left(\frac{1}{2}\right)^n\left(1+\frac{1}{2}\right)^n \quad \text{(by the binomial theorem)}$$

$$= \frac{c}{\sqrt{n}}\left(\frac{3}{4}\right)^n$$

In $3n$ steps, the success probability p is at least $\frac{c}{\sqrt{n}}\left(\frac{3}{4}\right)^n$.

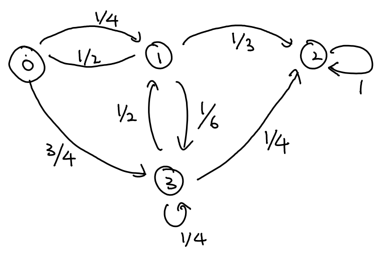
Thus, the expected number of times until a success is at most $\frac{\sqrt{n}}{c}\left(\frac{4}{3}\right)^n$.

Since each time takes $3n$ steps, the expected running time is $O\left(n^{1.5}\left(\frac{4}{3}\right)^n\right)$. □

This is not much slower than the current fastest algorithm, with randomized running time $O(1.308^n)$ [1].

<u>**Markov Chain**</u> (MU 7.1-7.3, MR 6.2, [2])

We analyze the general problem of random walk on a directed graph.



each vertex corresponds to a state

an arc corresponds to the transition prob. from state i to state j.

We can also formulate the problem as a matrix problem

$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1/4 & 0 & 3/4 \\ 1 & 1/2 & 0 & 1/3 & 1/4 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 1/2 & 1/4 & 1/4 \end{array}$$

Let $X(t)$ be the state at time t.

Let $p_i(t)$ be the probability of being in state i at time t.

e.g. $\vec{p}_0 = (1, 0, 0, \dots, 0)$ if the walk starts at state 0 in time 0,

or $\vec{p}_0 = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$ if the walk starts at a random state in time 0.

By the definition it follows that $\vec{p}_{t+1} = \vec{p}_t P$, and more generally $\vec{p}_{t+m} = \vec{p}_t P^{m-1}$

This random process is called a Markov chain because

$$\Pr(X(t)=a_t \mid X(t-1)=a_{t-1}, X(t-2)=a_{t-2}, \dots, X_0=a_0) = \Pr(X_t=a_t \mid X_{t-1}=a_{t-1}) = P_{a_{t-1}, a_t}$$

<u>**States**</u> (MU 7.2, Chapter 4 in [2])

We will assume the Markov chain is finite throughout this notes.

A Markov chain is <u>irreducible</u> if the corresponding graph is strongly connected

(i.e. there is a directed path from i to j and from j to i for every pair $i, j \in V$)

In other words, for every i, j, there exists an n s.t. $\Pr(X(m+n)=s_j \mid X(m)=s_i) > 0$

The <u>period</u> of state i is $d(s_i) = \gcd\{n \geq 1 : P^n_{i,i} > 0\}$.

State i is <u>aperiodic</u> if $d(s_i)=1$.

A Markov chain is aperiodic if all states are aperiodic; otherwise it is periodic.

(e.g. random walk in a bipartite graph is periodic since $d(s_i)=2$ for all i.)

<u>**Theorem**</u> (Thm 4.1, Cor 4.1 of [2])

For any finite, irreducible, aperiodic Markov chain, there exists $N < \infty$ s.t.

$(P^n)_{i,j} > 0$ for all i,j and for all $n \geq N$.

Stationary distribution (MU 7.3, chapter 5 of [2])

Stationary distribution (MU 7.3, chapter 5 of [2])

A stationary distribution of a Markov chain is a probability distribution $\vec{\pi}$ s.t.

$$\vec{\pi} = \vec{\pi} P.$$

Informally, $\vec{\pi}$ is a "steady" state or an "equilibrium" state or a "fixed-point" state, as

$$\vec{\pi} = \vec{\pi} P^t \text{ for any } t \geq 0.$$

Intuitively, given any finite, irreducible, aperiodic Markov chain, if we run it long enough, then we will completely forget about the history and converge to the same distribution.

Given two probability distribution $\vec{p} = (p_1, p_2, \dots, p_n)$ and $\vec{q} = (q_1, q_2, \dots, q_n)$, the total variation distance is defined as $d_{TV}(\vec{p}, \vec{q}) = \frac{1}{2} \sum_{i=1}^{n} |p_i - q_i|$

We say that $\vec{p}(t)$ converges to $\vec{q}$ if $\lim_{t \to \infty} d_{TV}(\vec{p}(t), \vec{q}) = 0$.

One more definition: the hitting time is defined as $T_{i,j} = \min\{n \geq 1 : X_n = j \mid X_0 = i\}$.

The mean hitting time (or just hitting time) is defined as $h_{i,j} = E[T_{i,j}]$.

The following theorem is known as the fundamental theorem of Markov chain.

Theorem For any finite, irreducible and aperiodic Markov chain:

① There exists a stationary distribution $\vec{\pi}$. (Thm 5.1 of [2])

② Any initial distribution $\vec{p}(0)$ will converge to $\vec{\pi}$. (Thm 5.2 of [2])

③ There is a unique stationary distribution. (Thm 5.3 of [2])

④ $\pi_i = \lim (P^t)_{i,i} = \frac{1}{h_{i,i}}$.

Proof outline

Step ①: Define $p_i = \sum_{n=0}^{\infty} \Pr(X_n = s_i, T_{1,1} > n)$.

In words, this is the expected number of visits to state i before returning to 1.

Let $\vec{\pi} = (\pi_1, \pi_2, \dots, \pi_n) = \left(\frac{p_1}{h_{1,1}}, \frac{p_2}{h_{1,1}}, \frac{p_3}{h_{1,1}}, \dots, \frac{p_n}{h_{1,1}}\right)$.

Then, with some calculations (see Thm 5.1 of [2]), this is a probability distribution ($\sum_{i=1}^{n} \pi_i = 1$) and it is stationary.

Intuitively, any walk can be broken into intervals s.t. each interval starts from state 1 and ends in state 1, and so this probability distribution reflects the long term behavior of the Markov chain.

Step ②: Informally, when two chains "meet" each other (i.e. have the same state), then they behave the same and you cannot distinguish them afterwards, and so they will have the same distribution.

This can be made formal by using the <u>coupling argument</u>, which we will go into details next time, and we will also come back to this convergence theorem.

Since there exists N s.t. $(P^n)_{i,j} > 0$ for all i,j and all $n \geq N$ by the above theorem, we know that two chains will meet with probability 1 and this will imply part ②.

Step ③ Any initial distribution will converge to $\vec{\pi}$. In particular another stationary distribution $\vec{\pi}'$ must converge to $\vec{\pi}$. But $\vec{\pi}$ and $\vec{\pi}'$ won't change over time since they are stationary, this means that $\vec{\pi}$ must be equal to $\vec{\pi}'$.

Step ④ Using the proof of ①. $\pi_1 = \frac{p_1}{h_{1,1}} = \frac{1}{h_{1,1}}$ because $p_1 = 1$ by definition.

If we start from state i, by ① we will have $\pi_i = \frac{1}{h_{i,i}}$.

But starting from which state doesn't matter by ③, thus we have $\pi_i = \frac{1}{h_{i,i}}$.

Intuitively, by the definition of $h_{i,i}$, we expect vertex i to be visited with probability $\frac{1}{h_{i,i}}$, and thus π_i should be equal to $1/h_{i,i}$. □

You can also see MU for a different proof of this fundamental theorem.

<u>PageRank</u> ([3])

Suppose we have a directed graph describing the relationships between a set of homepages, i.e. there is an arc from page i to page j if page i links to page j.

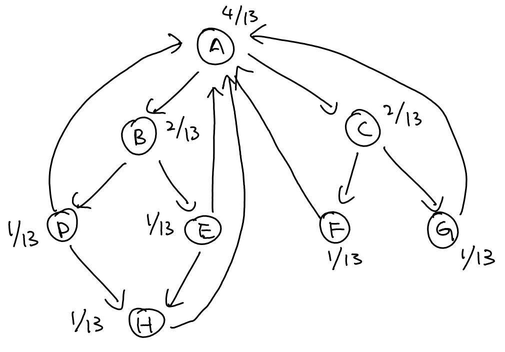
We want an algorithm to "rank" how important a page is.

Intuitively, a page being linked by many other pages is important, and a page being linked by an important page is important.

This motivates the following algorithm:

PageRank : - initially every page has PageRank value $1/n$.

- in each step, each page divides its current PageRank value equally to its outgoing links and sends these equal shares to the pages that it points to. Each page updates its new PageRank value to be the sum of the shares it receives.



the "equilibrium" PageRank values.

It is not difficult to see that the equilibrium PageRank values are equal to the probabilities in the stationary distribution of the random walk when each entry P_{ij} in P is equal to $1/d^{out}(i)$ where $d^{out}(i)$ is the outdegree of i in the graph, since the PageRank values and the stationary probabilities satisfy the same equations: $\text{pagerank}^{(t+1)}(v) = \sum_{u:\, uv\in E} \text{pagerank}^{(t)}(u) \quad \forall v \iff \overrightarrow{\text{pagerank}}^{(t+1)} = \overrightarrow{\text{pagerank}}^{(t)} \cdot P$.

Therefore, we know that if the graph is strongly connected and aperiodic, then the "equilibrium" pagerank values are unique regardless of the initial distribution.

In practice, the directed graph may not satisfy that condition, and the following modified process is used: fix a number $s>0$, divide s fraction of its pagerank value to its neighbors, divide $1-s$ fraction of its pagerank to all nodes evenly.

This is equivalent to the random walk that with probability s, go to a random neighbor, and with probability $1-s$, go to a random vertex. Then the resulting graph is strongly connected and aperiodic, and hence a unique stationary distribution exists.

Having a unique "equilibrium" pagerank value shows that the ranking only depends on the graph structure, but not on the initial pagerank values.

As a consequence, one can use a random walk to estimate the PageRank value (e.g. to empirically observe $h_{i,i}$). This gives a simple distributed algorithm to estimate the PageRank value.

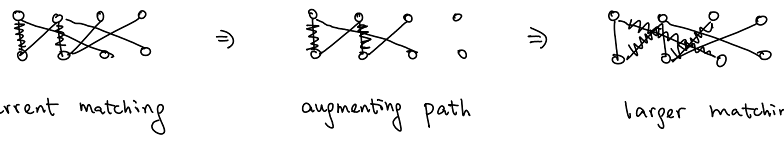
Perfect Matching in Regular Bipartite Graphs ([4])

The bipartite matching problem is to find a maximum number of vertex disjoint edges in a bipartite graph. By Hall's marriage theorem (see wiki), it is well-known that a regular bipartite graph always has a perfect matching (i.e. a bipartite graph with $2n$ vertices has a matching of size n).

There is a recent result showing how to find a perfect matching in such graphs in $O(n\log n)$ time. Note that this is sublinear time when the graph has much more than $n\log n$ edges (e.g. $\Omega(n^2)$ edges). The algorithm and the analysis are very elegant.

The traditional approach is to repeatedly find an augmenting path to enlarge the matching.

Pictorically,



An augmenting path is a path $v_1 - v_2 - \dots - v_{2\ell+1}$, where $v_{2i-1} - v_{2i}$ is an edge not in the current matching, $v_{2i} - v_{2i+1}$ is an edge in the current matching, and v_1 and $v_{2\ell+1}$ are unmatched vertices.

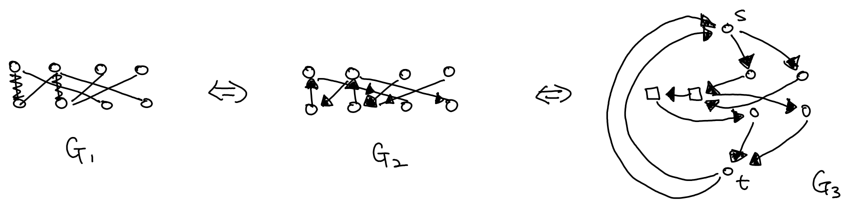
<u>Theorem</u> The current matching is maximum if and only if there is no augmenting path.

<u>proof idea</u> One direction is easy: If there is an augmenting path, then we can use it to enlarge the matching.

Another direction is to show that if the current matching M is not maximum, then there is an augmenting path, by considering the union of a larger matching M^* and M. $\square$

So, the maximum bipartite matching problem can be reduced to at most n subroutines of finding an augmenting path, which can be done in $O(m)$ time by a BFS (breadth first search).

The new idea is to replace BFS by random walk. Let me explain the algorithm by pictures.



G_1 is the original undirected bipartite graph with a matching M.

G_2 is the directed graph where each edge in the matching M is pointing upward, while every other edge (not in the matching) is pointing downward.

G_3 is the directed graph obtained from G_2, by contracting each edge in the matching M into a single (square) node. The source s has $d^{out}(v)$ edges to every unmatched vertex v on top, and every unmatched vertex u in bottom has $d^{in}(u)$ edges to the sink t. And t has $d^{in}(t)$ edges to s.

It is not difficult to show that ① G_1 has an augmenting path

It is not difficult to show that ① G_1 has an augmenting path

$\Leftrightarrow$ ② G_2 has a directed path from a top unmatched vertex to a bottom unmatched vertex

$\Leftrightarrow$ ③ G_3 has a cycle from s to s.

Also, it is not difficult to verify that G_3 is an Eulerian directed graph (i.e. the indegree is equal to the outdegree for every vertex) if G_1 is a regular bipartite graph.

So, if we do a random walk in G_3, then the expected time to find an augmenting path is equal to the expected hitting time $H_{s,s}$ in G_3.

Recall that $H_{s,s} = 1/\pi_s$, where π_s is the probability of being in s in the stationary distribution, which is easy to compute in Eulerian directed graph.

<u>Claim</u> In Eulerian directed graph, the stationary distribution is $\pi_v = \frac{d^{out}(v)}{m}$ when edge uv is traversed with probability $\frac{1}{d^{out}(u)}$.

<u>Proof</u> $$\pi_v = \sum_{u: uv \in E} \pi_u P_{u,v} = \sum_{u: uv\in E} \frac{d^{out}(u)}{m} \cdot \frac{1}{d^{out}(u)} = \frac{d^{in}(v)}{m} \overset{\text{Eulerian}}{=} \frac{d^{out}(v)}{m} = \pi_v.$$

Since $\sum_v \pi_v = \sum_v \frac{d^{out}(v)}{m} = 1$, this is the unique stationary distribution. □

Therefore, $H_{s,s} = m/d^{out}(s)$.

In the i-th iteration when there are only i edges in the matching, $d^{out}(s) \geq (n-i)\cdot d$ assuming G_1 is a d-regular graph.

So, in the i-th iteration, $H_{s,s} \leq dn/d(n-i) = n/(n-i)$.

Therefore, the total running time is $\sum_{i=0}^{n-1} n/(n-i) = O(n \log n)$.

With appropriate data structures, the total complexity of the algorithm is $O(n \log n)$.

<u>**Undirected graphs**</u> (MR 6.3)

In an undirected graph, $P_{uv} = 1/d(u)$ for any $u,v \in V$.

It is not difficult to show that the corresponding Markov chain is periodic if and only if the graph is bipartite, since an odd cycle can be used to show that $(P^t)_{i,j} > 0$ for a large enough t.

The (unique) stationary distribution is easy to compute.

<u>Claim</u> $\pi_v = d(v)$

proof The same as above for Eulerian directed graph. ●

We are interested in understanding the following quantities:

- hitting time $h_{i,j}$
- commute time $C_{i,j} = h_{i,j} + h_{j,i}$
- cover time : the first time when all vertices are visited at least once.

Interestingly, there are close connections between these quantities and electrical networks.

So let us take a small detour in electrical networks.

Electrical networks [MR 6.4]

A resistance electrical network is an undirected graph, where each edge has a resistance.

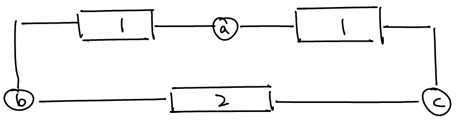
The flow of current in such networks is governed by two rules:

Kirchhoff's law : the sum of incoming currents is equal to the sum of outgoing currents.

Ohm's law : the voltage across a resistor is equal to the product of the resistance and the current through it.

e.g. if one unit of current is injected into node b and removed from node c,

then the electric flow is as follows: half an ampere in bc, ba, and ac.



The effective resistance between u and v is the voltage difference between u and v when one ampere is injected into u and removed from v.

Let $R_{u,v}$ denote the effective resistance between u and v.

Theorem For any u,v in G, the commute time $C_{uv} = 2m R_{uv}$ where $m = |E(G)|$.

proof For $v \in V$, let $N(v)$ be the set of neighbors of v, and $d(v) = |N(v)|$.

Let ϕ_{xv} be the voltage difference between x and v, if $d(x)$ amperes are injected into each $x \in V$, and 2m amperes are removed from v.

this is negative if the current is going from w to x

By Kirchoff's law, $d(x) = \sum_{w \in N(x)}$ (current going through xw)

$= \sum_{w \in N(x)}$ (voltage of x - voltage of w) (by Ohm's law)

$= \sum_{w \in N(x)} (\phi_{xv} - \phi_{wv})$ (because ϕ_{xv} = voltage of x - voltage of v)

On the other hand, for all $x \in V - v$, $h_{xv} = \sum_{w \in N(x)} \frac{1}{d(x)} (1 + h_{wv})$.

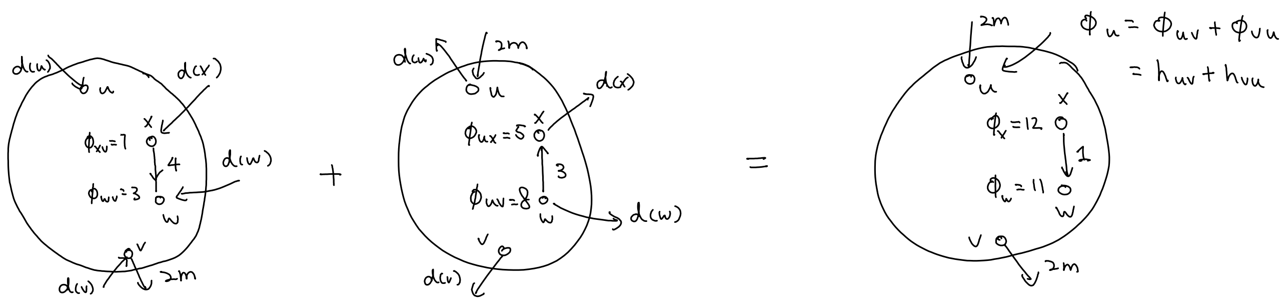
This is just equivant to $d(x) = \sum_{w \in N(x)} (h_{xv} - h_{wv})$.

Since this system of linear equations has a unique solution (why? because this is diagonal dominant and the graph is connected), we must have $\phi_{xv} = h_{xv}$ for all x.

By the same argument, h_{xu} is the potential difference between x and u, if $d(x)$ amperes are injected into each $x \in V$, and $2m$ amperes are removed from u.

If we reverse this flow (by changing sign on each edge), h_{xu} is the potential difference between u and x (note the order changed) when $2m$ amperes are injected into u and $d(x)$ amperes are removed from each $x \in V$.

Adding these two flows (the flow where amperes are injected into every node and removed from v, and the flow where amperes are injected into u and removed from v) together edge by edge and vertex by vertex, this results in a flow with $2m$ amperes injected into u and $2m$ amperes injected into v (check that it satisfies Kirchhoff's law and Ohm's law) where the incoming currents are equal to the outgoing currents at every other node. Pictorically,



The voltage difference between u and v in this new flow is just the sum of the two flows, which is equal to $h_{u,v} + h_{v,u}$, which is equal to $C_{u,v}$.

If we scale down everything (the voltages and the currents) by a factor of $2m$, then there is one unit of flow from u to v.

The voltage difference between u and v in this scaled-down flow is $\frac{C_{u,v}}{2m}$, and by definition this is the effective resistance between u and v, hence $C_{u,v} = 2m R_{u,v}$. $\square$

Once we have established the connection to electrical networks, we can use intuitions from Physics to prove bounds on different quantities.

Corollary For any edge uv, $C_{uv} \leq 2m$.

Proof The effective resistance between u and v is at most one (by Ohm's law). $\square$

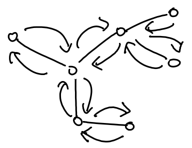
Alternative Proof (MU Lemma 7.15) $\frac{2|E|}{d(u)} = h_{u,u} = \frac{1}{d(u)} \sum_{w \in N(u)} (1 + h_{w,u})$

$\Leftrightarrow 2|E| = \sum_{w \in N(u)} (1 + h_{w,u}) \Rightarrow h_{w,v} < 2|E| \Rightarrow C_{w,v} < 4|E|.$ $\square$

Theorem The cover time of a connected graph is at most $2m(n-1)$.

Proof Let T be a spanning tree of G.

Consider a walk that goes through T where each edge in T is traversed once in each direction, e.g.



Then this is a walk that visites every vertex at least once.

So the cover time of G is bounded by the expected length of this walk, which is at most $\sum_{uv \in T} (h_{uv} + h_{vu}) = \sum_{uv \in T} C_{uv} \leq (n-1) \cdot 2m$,

where the last inequality follows by the corollary above. $\square$

For the complete graph with n vertices, the cover time is $O(n \log n)$ (why? similar to coupon collector), but the above bound only gives $O(n^3)$.

The following is a much better estimate of the cover time.

Theorem Let $R(G) = \max R_{u,v}$ and $C(G)$ be the cover time of G.

Then $mR(G) \leq C(G) \leq 2e^3 mR(G) \ln n + n$

Proof Let $R(G) = R_{uv}$. Then we know that $2m R_{uv} = C_{uv} = h_{uv} + h_{vu}$.

Therefore $C(G) \geq \max\{h_{uv}, h_{vu}\} \geq C_{uv}/2 = mR_{uv}$, hence the lower bound.

For the upper bound, note that the maximum hitting time is at most $2mR(G)$, regardless of the starting vertex.

So, if the random walk runs for $2e^3 m R(G)$ steps, by Markov's inequality, a vertex is not covered with probability at most $1/e^3$.

If the random walk runs for $2e^3 m R(G) \cdot \ln n$ steps, then this probability decreases to $1/n^3$.

By union bound, some vertex is not visited in $2e^3 m R(G) \cdot \ln n$ steps is of probability at most $1/n^2$. When this happens, we just use the bound in the above theorem that $C(G) \le n^3$.

Combining, we have $C(G) \le (1-\frac{1}{n^2}) \cdot 2e^3 m R(G) \cdot \ln n + (\frac{1}{n^2}) \cdot n^3 \le 2e^3 m R(G) \cdot \ln n + n$

Finally, from Physics we have the following result called the Rayleigh's short-cut principle, which says that the effective resistance never increases when we decrease the resistance of some edge, and the effective resistance never decreases if we increase the resistance of some edge.

Application of Rayleigh's principle : Suppose that G has p edge-disjoint paths from s to t, each of length at most l, then $R_{st} \le l/p$.

Proof: Increase the resistance of every other edge (i.e. the edges not in the paths) to infinity, the effective resistance from s to t could only increases, but even in this modified network the effective resistance is at most l/p (by direct calculations), hence the upper bound. □

Graph connectivity (MR 6.6)

If we want to test whether there is a path from s to t in an undirected graph, we can simply run a random walk with $2n^3$ steps. Since the cover time is at most n^3, the random walk will hit t with probability at least $1/2$ by the Markov's inequality. This algorithm has the feature that it uses only $O(\log n)$ space, but still runs in polynomial time. □

Next time we will see another application to generate a random spanning tree efficiently.

References

[1] Hertli. 3-SAT faster and simpler - unique-SAT bounds for PPSZ hold in general

[2] Häggström. Finite Markov chains and algorithmic applications

[3] Easley, kleinberg. Networks, Crowds, and Markets, chapter 14.

[4] Goel, krapralov, khanna. Perfect matchings in $O(n\log n)$ time in regular bipartite graphs.