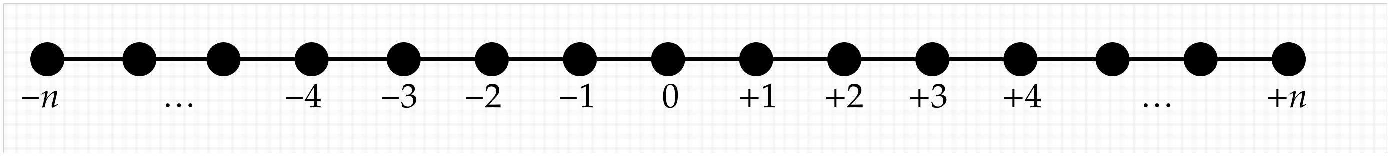
CSE610 Lecture Week 9: Quantum Walks

We first discuss classical random walks on graphs to set the points of comparison. The simplest graph is a path on $2n+1$ nodes, where we might let n go to infinity and/or connect the nodes in a circle:



A walker starts at 0 and flips a fair coin, stepping $+1$ for heads and -1 for tails. Here are some questions to ask about eir location after t such steps:

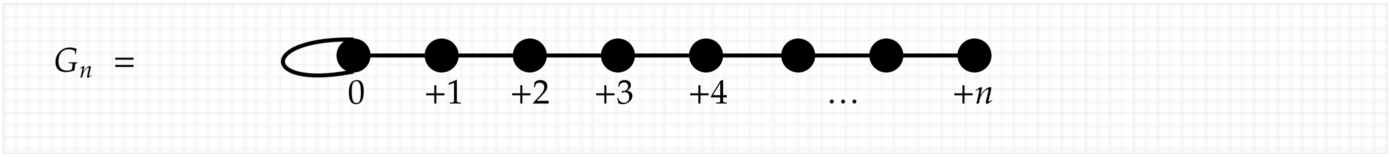
1. What is the expected distance from the origin 0?
2. How about the expected squared distance, and its square root?
3. What is the expected time to reach cell $+n$?
4. How many different nodes would we expect to be visited within t steps?

We can ask these questions also about **biased** cases where the probability of heads is p and tails is $q = 1-p$.

Question 2 is the easiest: the answers are given by the **variance** and **standard deviation** of the **binomial distribution** B_p. The variance of t independent trials is tpq, so the standard deviation is $\sqrt{tpq}$. This equals $\frac{1}{2}\sqrt{t}$ when $p = q = 0.5$. So for $t = 100$ random steps, the standard deviation is 5. We can also say the one-sigma range is ± 5 and the "usual margin of error" is ± 10. (This presumes the graph is large enough to accommodate t steps in any direction.)

Question 1 is a little different. By averaging the absolute displacement rather than its square, the expectation is a little closer to the center: $\frac{\sqrt{t}}{\sqrt{2\pi}}$ rather than $\frac{\sqrt{t}}{2}$. The growth order is the same.

Question 3 is inverse to questions 1 and 2. The growth order is quadratic in n---*if the graph is finite.* There is a surprising, maybe shocking, difference if the graph is actually infinite. Consider:



Here the rule is that at cell 0, heads goes to cell 1 but tails stays on 0. Now let $E[m]$ stand for the expected time to reach cell n when starting from cell m, $0 \leq m \leq n$. By definition, $E[n] = 0$. We recurse, however, from the other end. First, $E[0] = 1 + 0.5E[0] + 0.5E[1]$. This is because we take 1 step and then have a 50% chance of having made no progress and a 50% chance of making a positive step. Next, $E[1] = 1 + 0.5E[0] + 0.5E[2]$. This time, tails does lose a step of progress. Now we can solve for both $E[1]$ and $E[2]$ in terms of $E[0]$:

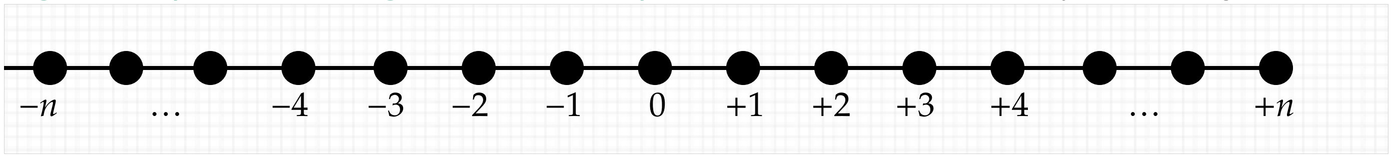
$E[1] \;=\; E[0]-2$
$E[2] \;=\; 2E[1]-E[0]-2 \;=\; E[0]-6$

Now we keep going, using $E[m] \;=\; 1+0.5E[m-1]+0.5E[m+1]$, so
$E[m+1] \;=\; 2E[m] \;-\; E[m-1]-2$, and always substituting values down to $E[0]$:

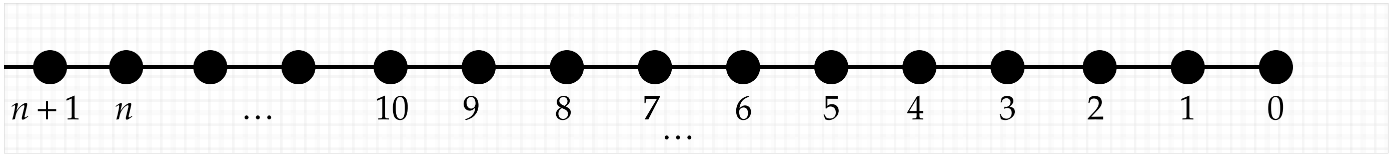
$E[3] \;=\; 2E[2]-E[1]-2 \;=\; 2E[0]-12-E[0]+2-2 \;=\; E[0]-12$;
$E[4] \;=\; 2E[3]-E[2]-2 \;=\; 2E[0]-24-E[0]+6-2 \;=\; E[0]-20$;
$E[5] \;=\; 2E[4]-E[3]-2 \;=\; 2E[0]-40-E[0]+12-2 \;=\; E[0]-30$;
...
$E[n] \;=\; E[0]-\left(n^2+n\right)$.

But $E[n]=0$ from above, so this means $E[0]=n^2+n$ is the expected time to get to cell n. Which again is quadratic.

Digression (Piazza challenge rather than HW): Now, as a curious exercise, try it with the graph



where the only change to the first pic above is adding an edge left of $-n$ to convey that the graph is infinite in that direction. Note that nodes to the right of $+n$ do not matter, and because of this, we can flip the indexing around so that the goal node is 0 and the indexing of other nodes is positive saying how far away from 0 they are:



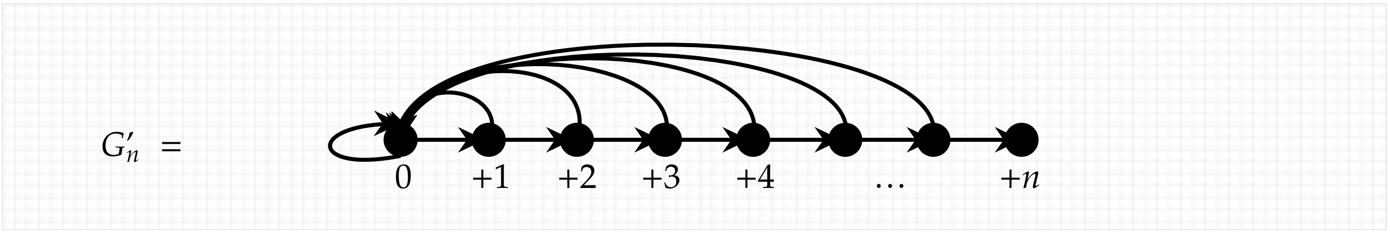
Here we have $E[0]=0$ and $E[1]=1+0.5*0+0.5E[2]$, so $E[2]=2E[1]-2$. And:
$E[2]=1+0.5E[1]+0.5E[3]$, so $E[3] \;=\; 2E[2]-E[1]-2 \;=\; 3E[1]-6$. We actually have the same recursion formula as before: $E[m+1]=2E[m]-E[m-1]-2$, so continuing along:

$E[4] \;=\; 2E[3]-E[2]-2 \;=\; 2\times 3E[1]-2\times 6 \;-\; 2E[1]+2-2 \;=\; 4E[1]-12$
$E[5] \;=\; 2E[4]-E[3]-2 \;=\; 2\times 4E[1]-2\times 12-3E[1]+6-2 \;=\; 5E[1]-20$
$E[6] \;=\; 2E[5]-E[4]-2 \;=\; 2\times 5E[1]-2\times 20-4E[1]+12-2 \;=\; 6E[1]-30$
...
$E[n] \;=\; nE[1]-\left(n^2-n\right)$.

Predicated on the assumption that $E[1]$ has a fixed, finite value (after all, $E[0]=0$), we'll leave you to interpret what this means...this is what one can call a "predicament."

Question 4 builds on question 3. In the case of our path graphs with binomial distribution, as long as $t \ll n^2$, the answer is that the expected number of different nodes visited within t steps is $O\left(\sqrt{t}\right)$. We may care more about the expected number of visits to nodes that are furthest from the starting point. This leads to the larger issue of how fast the initial distribution **diffuses** into the rest of the graph.

If the graph is a directed graph, the expectations can change drastically. The most fiendish such case is when all nodes have a directed edge going back to node 0.



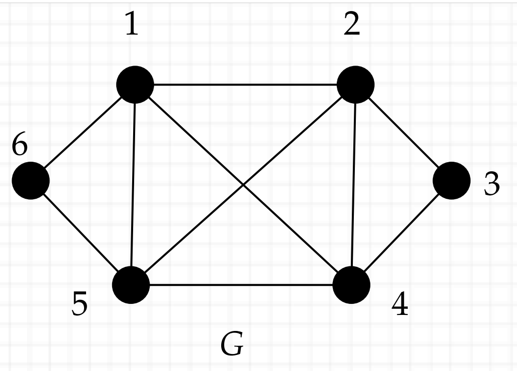
It is still the case that every node (other than $+n$) has two out-edges, one goping right for heads and the one going to cell 0 for tails. Now the only way to reach node n is sometime to get n heads in a row. The expected time to do that is $O\left(2^n\right)$, which is exponential. This huge difference from the undirected case can be chalked up to lack of reversibility. If the edges went both ways, there would also be a chance of getting from 0 to n by a big leap followed by just a few more lucky steps on heads. Quantum systems are *ipso-facto* reversible, so we should not expect the directed case to apply. (We will, however, use directed graphs as a tool to visualize quantum walks.)

Classical Random Walks in General Graphs

The **degree** d_v of a vertex v is the number of edges going out of v, counting any self-loop at v (if allowed) *once*. If $d_v = d$ is the same for all v, then the graph is **d-regular**. It is possible for a graph G to be infinite and yet have d_v be finite for all nodes v; the infinite path graph is an example with $d_v = 2$ for all v. In such cases, G is called **locally finite**. (There is a further breakdown into whether there is a fixed finite bound b such that $d_v \leq b$ for all nodes v, but I do not know a standard name for that condition.)

Definition: The **"standard" classical random walk** on a (locally-)finite graph G has steps that go from a node i to any one of its neighbors j with probability $A[i,j] = 1/d_v$. The matrix A is called the **transition probability matrix** of the walk.

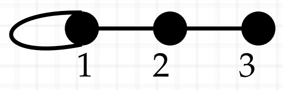
For example, the transition matrix A of the standard walk on the graph G at left is shown at right.



$$A_G = \left[\begin{array}{c|cccccc} p & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0.25 & 0 & 0.25 & 0.25 & 0.25 \\ 2 & 0.25 & 0 & 0.25 & 0.25 & 0.25 & 0 \\ 3 & 0 & 0.5 & 0 & 0.5 & 0 & 0 \\ 4 & 0.25 & 0.25 & 0.25 & 0 & 0.25 & 0 \\ 5 & 0.25 & 0.25 & 0 & 0.25 & 0 & 0.25 \\ 6 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \end{array}\right]$$

The matrix A_G is **row-stochastic**, meaning that the values in each row sum to 1. It is not **column-stochastic**, because G is not regular---for instance, column 2 sums to more than 1. Any d-regular graph G makes A_G **doubly stochastic**---and then A_G is just the ordinary adjacency matrix of G divided by d.

It is possible to define transition probabilities different from those of the standard random walk on G. For instance, we could stipulate that node 1 has only a $\frac{1}{6}$ probability of going to node 2, node 4, or node 5, but a $\frac{1}{2}$ probability of going to 3. Doing similarly with nodes 2, 4, and 5 has the same effect as "regularizing" the graph by tripling each edge to node 3 or node 6, and makes the revised transition matrix A'_G double-stochastic after all. Some graphs are not regularizable by replicating edges or assigning weights, however, including the "lollipop" graph we've used elsewhere:



$$\begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{bmatrix}$$

The problem is that however we replicate or weight the edge between 2 and 3, there will always be more weight at node 2 than at weight 3. Unless we make the weight of the edge $(1,2)$ zero, that is. The alternative is adding a loop to state 3, but that is really a different graph.

We can track the expectations for steps of the quantum walk in the same way we've seen with quantum state vectors, but this time using classical probability not amplitude. The row-vector $x = [1,0,0,0,0,0]$ stands for a probability distribution that is entirely concentrated on node 1. This represents our "walker" starting in a definite position. Unlike with quantum updates, it seems to be more usual to use row vectors and multiply them on the left. Here we get

$$x(1) \;=\; xA_G \;=\; [0, 0.25, 0, 0.25, 0.25, 0.25] \;=\; \tfrac{1}{4}[0,1,0,1,1,1]$$

as the probabilities after one step. To do more steps we just multiply again and again:

$$x(2) \;=\; x(1)A_G \;=\; \left[\tfrac{5}{16}, \tfrac{1}{8}, \tfrac{1}{8}, \tfrac{1}{8}, \tfrac{1}{4}, \tfrac{1}{16}\right] \;=\; \tfrac{1}{16}[5,2,2,2,4,1] \;;$$

$$x(3) = x(2)A_G \;=\; \left[\frac{3}{16}, \frac{15}{64}, \frac{1}{16}, \frac{13}{64}, \frac{11}{64}, \frac{9}{64}\right] \;=\; \frac{1}{64}[12, 15, 4, 13, 11, 9]\;.$$

We could get the same effect by taking powers of A_G. The first row of A_G^2 is $x(2)$, the first row of A_G^3 is $x(3)$, and so on. Is there a distribution x such that $xA_G = x$? If so, then x must be a row-eigenvector of A_G with eigenvalue 1. If G is d-regular then this is obvious: the uniform distribution on the n nodes is stationary. For this particular G, we can skip the route of first diagonalizing A_G and use the symmetry to solve in brute force:

$$[p, p, q, p, p, q]A_G \;=\; [p, p, q, p, p, q]$$

gives the equations $4p + 2q = 1$ and (using just column 3 of the matrix) $0.5p = q$, so $p = 0.2$ and $q = 0.1$. This gives

$$x \;=\; x_G \;=\; \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}\right].$$

Surprised to get fifths out of a matrix with denominators 2 and 4? Well, there are ten edges, and there is a simple formula that works for the standard classical random walk:

$$x_G[u] \;=\; \frac{d_u}{\sum_v d_v}.$$

When there are no self-loops, the denominator is the same as twice the number of edges in the graph. For the lollipop graph G, however, we get denominator 5 not 6: $x_G = [0.4, 0.4, 0.2]$, and to verify:

$$[0.4, 0.4, 0.2]\begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \end{bmatrix} \;=\; [0.2 + 0.2,\ 0.2 + 0.2,\ 0.2] \;=\; x_G\;.$$

Then x_G is called a (the) **stationary distribution** for A_G and represents the long-run expectation of the classical random walk. Here is a remarkable fact:

Theorem. If G is connected, then there is a unique stationary distribution x_G. Moreover, if G is **not** bipartite, then for any input distribution x, $x(t)$ converges to x_G as $t \to \infty$; likewise, the powers A_G^t of A_G converge to the matrix whose rows all equal x_G.

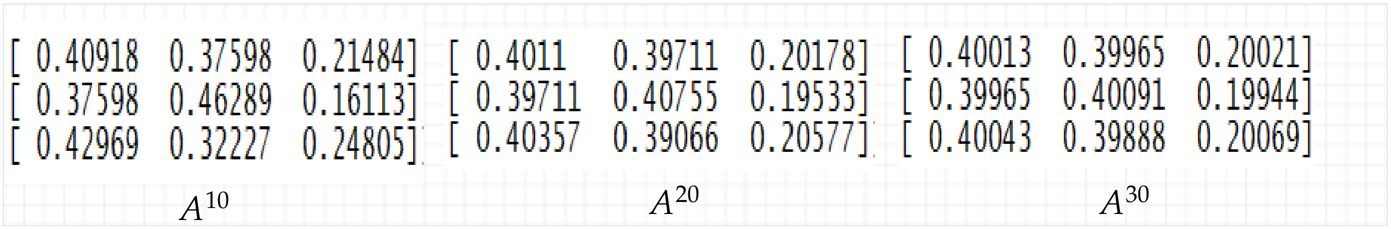
This is a special case of a more-general theorem about **discrete Markov chains**. We can see the convergence happening in our 6-node example: for $t = 10$ the matrix is

```
[ 0.20181   0.19819   0.1014    0.19819   0.20181   0.0986 ]
[ 0.19819   0.20181   0.0986    0.20181   0.19819   0.1014 ]
[ 0.20279   0.19721   0.10222   0.19721   0.20279   0.09778]
[ 0.19819   0.20181   0.0986    0.20181   0.19819   0.1014 ]
[ 0.20181   0.19819   0.1014    0.19819   0.20181   0.0986 ]
[ 0.19721   0.20279   0.09778   0.20279   0.19721   0.10222]
```

and for $t = 20$ it is

```
[ 0.20002   0.19998   0.10002   0.19998   0.20002   0.09998]
[ 0.19998   0.20002   0.09998   0.20002   0.19998   0.10002]
[ 0.20003   0.19997   0.10003   0.19997   0.20003   0.09997]
[ 0.19998   0.20002   0.09998   0.20002   0.19998   0.10002]
[ 0.20002   0.19998   0.10002   0.19998   0.20002   0.09998]
[ 0.19997   0.20003   0.09997   0.20003   0.19997   0.10003]
```

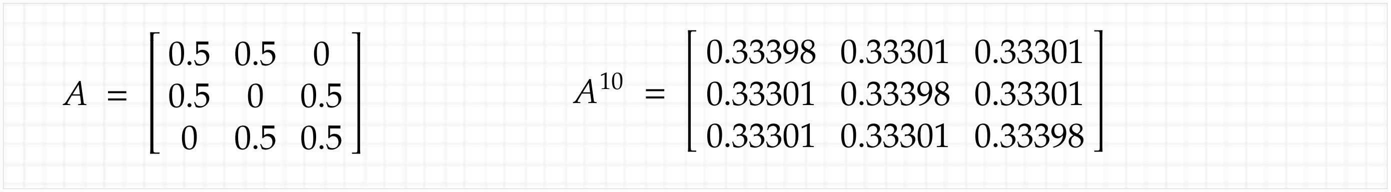
Curiously, the convergence for our lollipop-graph example is not as fast although the graph is smaller. For $t = 10, 20, 30$, respectively, we get:



```
[ 0.40918  0.37598  0.21484]  [ 0.4011   0.39711  0.20178]  [ 0.40013  0.39965  0.20021]
[ 0.37598  0.46289  0.16113]  [ 0.39711  0.40755  0.19533]  [ 0.39965  0.40091  0.19944]
[ 0.42969  0.32227  0.24805]  [ 0.40357  0.39066  0.20577]  [ 0.40043  0.39888  0.20069]
```

A^{10} A^{20} A^{30}

Only on A^{30} is the convergence visually clear (IMHO). If we add a loop at node 3 to make the graph regular, the limiting distribution is uniform and the convergence becomes quite fast again:



$$A = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{bmatrix} \qquad A^{10} = \begin{bmatrix} 0.33398 & 0.33301 & 0.33301 \\ 0.33301 & 0.33398 & 0.33301 \\ 0.33301 & 0.33301 & 0.33398 \end{bmatrix}$$

Finally coming back to our questions 3 and 4, the issue is how fast this convergence goes as a function of n (and distance m from the starting node) in general graphs. Given any $\epsilon > 0$, the **mixing time** is the least t such that the convergence is within ϵ (in each entry of A^t or however measured), and the **mixing rate** expresses this as a function of ϵ (and the size n of the graph). The short answer is that the time is never worse than the $O(n^2)$ that we computed for the path graphs. But it is also generally not better. This is the springboard for interest in quantum walks.

Quantum Walks

One principle of quantum mechanics that we have covered distinguishes between modeling part of a system and the whole system. In a classical random walk we might think of "the system" as the graph G and the current position of the walker, but there is also the source of randomness, which is often called "the coin." We can still represent the system as a composite, but we need to allow (states involving) the two parts to become entangled.

An earlier principle is that any quantum object lives in a "Hilbert Space", so we can postulate a space $\mathbb{H}_G$ for the graph and a space $\mathbb{H}_C$ for the coin. The notion of the walker on the graph will be subsumed by a state that incorporates status info from both systems. Basis elements of the composite system $\mathbb{H}_G \otimes \mathbb{H}_C$ (if you recall discussion in early lectures, it is not the ordinary Cartesian product $\mathbb{H}_G \times \mathbb{H}_C$) will be separable, but their linear combinations can be entangled.

Let $|v\rangle$ represent the current "basic vertex" of the walker, and $|c\rangle$ a basic outcome of the coin. Then $|vc\rangle$ is the joint state of the system. State vectors **a** have entries $\mathbf{a}[vc]$ over all vertices v and outcomes c. There is one immediate and maybe surprising difference from how graphs have been represented before. Instead of one qubit per node, as with graph-state circuits, now we need only as many qubits as it takes to count v from 1 to n. That is only $\ell = \lceil \log_2 n \rceil$ qubits. We thus get simply $\mathbb{H}_G = V$ of size n, as opposed to the 2^n-dimensional Hilbert space underlying n qubits. Likewise, the coin space $\mathbb{H}_C$ becomes just the set C of coin outcomes; if $C = \{H, T\}$ for heads and tails, then we need just one qubit for it. The representation is thus more compact, and the matrix A' for the $\mathbb{H}_G$ side of the space has the same size-order as A_G. This hints that the goings-on within $\mathbb{H}_G$ have a classical skeleton.

If $\mathbb{H}_C$ uses r qubits, then an operator on $\mathbb{H}_G \otimes \mathbb{H}_C$ is given by a $2^{\ell+r} \times 2^{\ell+r}$ matrix $\mathbf{U}$. Even though we are not in the context of density matrices, the traceouts $\mathbf{Tr}_{\mathbb{H}_C}(\mathbf{U})$ and $\mathbf{Tr}_{\mathbb{H}_G}(\mathbf{U})$ are both well-defined. If the coin uses the last r quantum coordinates (in big-endian notation, say), then each entry of the $2^\ell \times 2^\ell$ matrix $A' = \mathbf{Tr}_{\mathbb{H}_C}(\mathbf{U})$ is obtained by summing the 2^r diagonal elements of the corresponding $2^r \times 2^r$ submatrix of $\mathbf{U}$.

Speaking from formal mathematics, I would like to be able to give an "axiomatic" definition of a quantum walk, something like this:

Definition(?) A **quantum walk** on a graph G is defined by a unitary matrix $\mathbf{U}$ acting on a joint space $\mathbb{H}_G \otimes \mathbb{H}_C$ such that $\mathbf{Tr}_{\mathbb{H}_C}(\mathbf{U})$ gives a classical random walk on G.

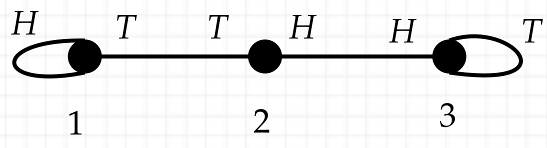
This might need the added requirement that $\mathbf{Tr}_{\mathbb{H}_G}(\mathbf{U})$, which acts on $\mathbb{H}_C$, is unitary. But the intent is that A' after "tracing out the coin" need not be unitary---it could and maybe should be classical. The waffle is in the word "gives". One might think to say that A' **is** a row-stochastic matrix denoting a classical random walk on G, maybe even the standard one, but the mathematical truth is otherwise---and comes with "ample amplitudinal attitude." I still have not found a simple definition in this style online---see below for why it gets trickier.

Instead, what I found in my sources, used for the textbook (in 2014), and what Wikipedia still gives (note the request for expansion dating to Dec. 2009!) is an "operational" definition:

The evolution of a quantum walk in discrete time is specified by the product of two unitary operators: (1) a "coin flip" operator and (2) a conditional shift operator, which are applied repeatedly.

I used a slightly more general form that allows entanglement, but it still feels ad-hoc and is limited to d-regular(izable) graphs. Each non-loop edge (u,v) has two labels in $\{1, \ldots, d\}$, one outgoing from u and incoming to v, the other outgoing from v and incoming from u. Loops have just one label that is both incoming and outgoing. The outgoing labels at a node u denote the actions of a walker on that node from the d possible outcomes of the "coin" (or rather, d-sided die). We need to do all the labels so that not only does each node have d separate outgoing labels, each has d separate incoming labels. For example, if we add another loop to the "lollipop" graph we make it 2-regular and can do the labels with H and T for heads and tails to meet the extra requirement:

$G =$



$$A_G = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{bmatrix}$$

The requirement makes the map $(u,c) \to (v,c)$ one-to-one, where v is the destination for the coin outcome c at vertex u. It is thus a permutation of $V \times C$ (with ordinary Cartesian product), which we can represent both in linear form as

$$\begin{bmatrix} 1H & 1T & 2H & 2T & 3H & 3T \\ 1H & 2T & 3H & 1T & 2H & 3T \end{bmatrix}$$

and as a permutation matrix (which is symmetric because the permutation is self-inverse):

$$\mathbf{P} = \left[\begin{array}{c|cccccc} & 1H & 1T & 2H & 2T & 3H & 3T \\ 1H & 1 & 0 & 0 & 0 & 0 & 0 \\ 1T & 0 & 0 & 0 & 1 & 0 & 0 \\ 2H & 0 & 0 & 0 & 0 & 1 & 0 \\ 2T & 0 & 1 & 0 & 0 & 0 & 0 \\ 3H & 0 & 0 & 1 & 0 & 0 & 0 \\ 3T & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

If we represent the vertices as a **qutrit** then we actually do have proper quantum coordinates (here, in "big-endian" order), though again the vertices are underlying, not separate quantum entities. A single qutrit is acted on by a 3×3 rather than 2×2 unitary matrix, including the 3×3 identity matrix $\mathbf{I}_3$. We can get another 6×6 matrix in the form $\mathbf{Q} = \mathbf{I}_3 \otimes \mathbf{C}$, where $\mathbf{C}$ is a unitary 2×2 matrix acting on $\{|H\rangle, |T\rangle\}$ as the coin space. Whatever matrix we choose for $\mathbf{C}$, the product

$$\mathbf{P} \cdot \mathbf{Q} |\phi\rangle$$

represents first flipping the coin and then moving the walker according to the (superposed) outcome. The six standard basis states for $|\phi\rangle$ are $|vc\rangle$ for $v \in V = \{1,2,3\}$ and $c \in C = \{H,T\}$; these are separable but superpositions of them generally won't be. If we choose the Hadamard matrix $\mathbf{H}$ as the coin matrix, then we get

$$\mathbf{Q} = \mathbf{I}_3 \otimes \mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}, \quad \mathbf{PQ} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}.$$

Then $\mathbf{U} = \mathbf{PQ}$ is the unitary matrix for one step of the quantum walk. When we trace out the coin (summing the pair of colored numbers in each box), we get the 3×3 matrix

$$A = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}.$$

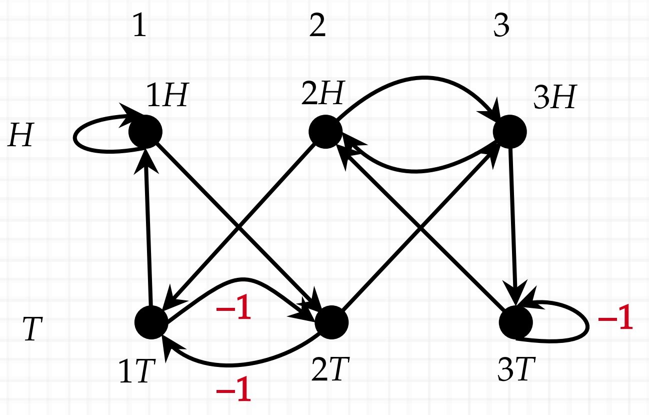
It is not A_G; its rows and columns all happen to add to 0 not 1. Nor is it unitary:

$$A^2 = \frac{1}{2}\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}.$$

The matrix A is Hermitian, but it is not a density matrix because the trace is 0. The matrix A does, however, kind-of represent the classical walk: it would equal A_G *if you squared each entry*. That tempts us to think that $\mathbf{Tr}_C(\mathbf{U})$ **gives amplitudes for the standard classical walk in genera.** But we will see that tracing *the matrix* breaks under powers of $\mathbf{U}$ for reasons that are subtle (meaning, I may not understand them---"I was only following sources").

Let us move on with the "properly quantum" matrix $\mathbf{U} = \mathbf{PQ}$. Its action can be visualized via a similar "maze diagram" to before, but as a grid out of $V \times C$. For a gridpoint (v, c), the c stands for the previous coin flip. Since we are applying $\mathbf{U}$ to column vectors on the right, not row vectors on the left, the point (v', c') is a possible destination if $\mathbf{U}[v'c', vc] \neq 0$. Then we draw a *directed* "maze edge" from
(v, c) to (v', c'). Here is what we get for our "propeller graph"---sticking the normalizing factor into a corner so as not to forget it:

$$\frac{1}{\sqrt{2}}\left[\begin{array}{ccccccc} & 1H & 1T & 2H & 2T & 3H & 3T \\ 1H & 1 & 1 & 0 & 0 & 0 & 0 \\ 1T & 0 & 0 & 1 & -1 & 0 & 0 \\ 2H & 0 & 0 & 0 & 0 & 1 & 1 \\ 2T & 1 & -1 & 0 & 0 & 0 & 0 \\ 3H & 0 & 0 & 1 & 1 & 0 & 0 \\ 3T & 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right]$$

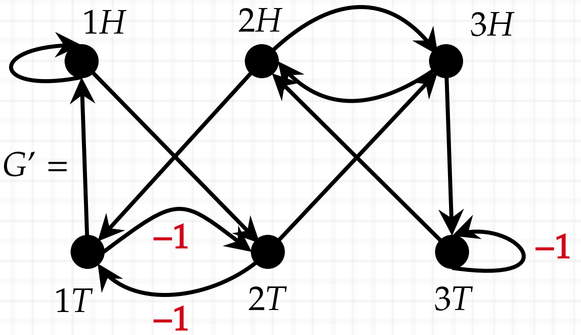


Each node in the "expanded graph" G' has two arrows coming in and two going out, counting the loops as both. The two nodes in each column correspond to the same node in the original graph G. If we start the walker on node 1 in the original graph, now we have a whole spectrum of choices: start with the basis vector $|1H\rangle = [1,0,0,0,0,0]^T$, the vector $|1T\rangle = [0,1,0,0,0,0]^T$, or any superposition of them. If we choose $\sigma_0 = |1H\rangle$, then the state σ_1 of the walk after one step (including the retained memory of the previous coin flip) is given by the first column of $\mathbf{U}$:

$$\frac{1}{\sqrt{2}}[1,0,0,1,0,0]^T = \frac{|1H\rangle + |2T\rangle}{\sqrt{2}}.$$

This is the equal superposition of staying at node 1 via the loop on heads or moving to node 2 on tails. Here are the next two powers:

$G' =$



$$\mathbf{U}^2 = \frac{1}{2}\left[\begin{array}{c|cccccc} 1H & 1 & 1 & 1 & -1 & 0 & 0 \\ 1T & -1 & 1 & 0 & 0 & 1 & 1 \\ 2H & 0 & 0 & 1 & 1 & 1 & -1 \\ 2T & 1 & 1 & -1 & 1 & 0 & 0 \\ 3H & 1 & -1 & 0 & 0 & 1 & 1 \\ 3T & 0 & 0 & 1 & 1 & -1 & 1 \end{array}\right]$$

$$\mathbf{U}^3 = \frac{1}{2\sqrt{2}}\left[\begin{array}{c|cccccc} 1H & 0 & 2 & 1 & -1 & 1 & 1 \\ 1T & -1 & -1 & 2 & 0 & 1 & -1 \\ 2H & 1 & -1 & 1 & 1 & 0 & 2 \\ 2T & 2 & 0 & 1 & -1 & -1 & -1 \\ 3H & 1 & 1 & 0 & 2 & 1 & -1 \\ 3T & 1 & -1 & -1 & -1 & 2 & 0 \end{array}\right]$$

$$\mathbf{U}^4 = \frac{1}{4}\left[\begin{array}{c|cccccc} 1H & -1 & 1 & 3 & -1 & 2 & 0 \\ 1T & -1 & -1 & 0 & 2 & 1 & 5 \\ 2H & 2 & 0 & -1 & 1 & 3 & -1 \\ 2T & 1 & 3 & -1 & -1 & 0 & 2 \\ 3H & 3 & -1 & 2 & 0 & -1 & 1 \\ 3T & 0 & 2 & 1 & 3 & -1 & -1 \end{array}\right]$$

The red 0 at upper left in $\mathbf{U}^3$ comes about because the paths $1H \rightarrow 1H \rightarrow 1H \rightarrow 1H$ and $1H \rightarrow 2T \rightarrow 1T \rightarrow 1H$ have opposite sign and so cancel. This **interference** does not mean that the quantum walker cannot end up at node 1 after 3 timsteps. Ey can do $1H \rightarrow 1H \rightarrow 2T \rightarrow 1T$ with a minus sign that is not cancelled. This does, however, knock down the overall probability. To get the probabilities, we square the entries of the state vector

$$\mathbf{U}^3|1H\rangle = \left[0, \frac{-1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}\right]^T \text{ to get } \left[0, \frac{1}{8}, \frac{1}{8}, \frac{1}{2}, \frac{1}{8}, \frac{1}{8}\right].$$

Now---using that the coin is little-endian---we say we don't care whether the last flip was H or T by adding the adjacent pairs of values to get the probabilities $[\frac{1}{8}, \frac{5}{8}, \frac{1}{4}]$. (The text represents the state vectors as $2 \times n$ grids in order to make clearer which pairs to add.) This means that after three steps, the quantum walker started on node 1, and with the coin happening to lie heads-up before it is Hadamard-ly flipped the first time---the walker is 5 times as likely to be found on node 2 as on node 1. This feels counter-intuitive because 1 has a self-loop: what easier way to stay there could there be?

The numbers in teal show that if we "trace out the coin" in the matrix $\mathbf{U}^3$, then we get zeroes in three of the entries. In full,

$$Tr_C\left(\mathbf{U}^3\right) = \frac{1}{2\sqrt{2}}\begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \end{bmatrix}, \text{ which on squaring entries becomes } \begin{bmatrix} 1/8 & 1/8 & 0 \\ 1/8 & 0 & 1/8 \\ 0 & 1/8 & 1/8 \end{bmatrix}.$$

The issue is not just that this seems to say the walker cannot go from node 1 to node 3 in three steps---we just saw that the probability of that is $\frac{1}{4}$. The larger issue is that a factor of 4 has been lost from the probabilities. This is (IMHO) mysitfying. The reason may be that tracing out the coin from $\mathbf{U}$ was OK because we are modeling ignorance of the prior state of the coin over just one prior maybe-flip. But with $\mathbf{U}^3$, the ignorance is of at least two flips in the actual history, and that may be too much. Adding the pairs of probabilities $\mathbf{U}^3|1H\rangle$ only declared indifference to the one last flip.

The behavior gets quite wild. Do three steps thrice, and you get $\mathbf{U}^9 =$

```
[ 18    4   -1   13   -1   -1]
[ 13    1    4  -18   -1    1]
[ -1   13   -1   -1   18    4]
[  4  -18   -1    1   13    1]
[ -1   -1   18    4   -1   13]
[ -1    1   13    1    4  -18]
```

divided by $4\sqrt{2}$. The next step is $\mathbf{U}^{10} = \frac{1}{32}$ times

```
[31   5   3  -5  -2   0]
[-5  31   0  -2   5   3]
[-2   0  31   5   3  -5]
[ 5   3  -5  31   0  -2]
[ 3  -5  -2   0  31   5]
[ 0  -2   5   3  -5  31]
```

This heavily favors staying on the same node, even if the walker starts on node 2, but taking just 2 more quantum steps gives $\mathbf{U}^{12} = \frac{1}{64}$ times

```
[ 19  33  39 -33   6   0]
[-33  19   0   6  33  39]
[  6   0  19  33  39 -33]
[ 33  39 -33  19   0   6]
[ 39 -33   6   0  19  33]
[  0   6  33  39 -33  19]
```

and the probabilities of being on any (other) node are fairly even again. Whatever, there does not seem to be any convergence to the uniform distribution as for the classical walk on a regular graph.

[At this point, one could go into the six-node graph example, then rejoin the text for the path-graph example (which I connected in a circle to make it regular, but maybe having end loops would be better). One can work through examples with other coin matrices. Is it safe to proclaim as a definition: A **quantum walk** on a graph G is defined by a unitary matrix $\mathbf{U}$ acting on a joint space $\mathbb{H}_G \otimes \mathbb{H}_C$ such that squaring the entries of $\mathbf{Tr}_{\mathbb{H}_C}(\mathbf{U})$ gives a classical random walk on G. ---?]

Simple Python code for the examples in these notes---can play around with it more:

```
import numpy as np
import math
#from numpy import random
np.set_printoptions(precision=5, suppress=True, linewidth=120)

#A = np.array([0,1,0,1,1,1,1,0,1,1,1,0,0,2,0,2,0,0,1,1,1,0,1,0,1,1,0,1,0,1,2,0,0,0,2,0])
#A = A.reshape(6,6)/4.0
#A = np.array([[0.5,0.5,0.0],[0.5,0.0,0.5],[0.0,1.0,0.0]])
#A = np.array([[0.5,0.5,0.0],[0.5,0.0,0.5],[0.0,0.5,0.5]])
A = np.array([1,1,0,0,0,0,0,0,1,-1,0,0,0,0,0,0,1,1,1,-1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,-1])
A = A.reshape(6,6)   #/math.sqrt(2.0)
Ai = np.matmul(A,A)
print(1)
print(A)
print()
print(2)
print(Ai)
print()
for i in range(18):
  Ai = np.matmul(Ai,A)
  print(i+3)
  print(Ai)
  print()
```