A whirling tour of chip-firing and rotor-routing

Jim Propp
U. Mass. Lowell

(based on articles in progress by Ander Holroyd, Lionel Levine, and Jim Propp; slides available at `jamespropp.org/tour.pdf`)

presented in honor of Peter Winkler
June 8, 2007

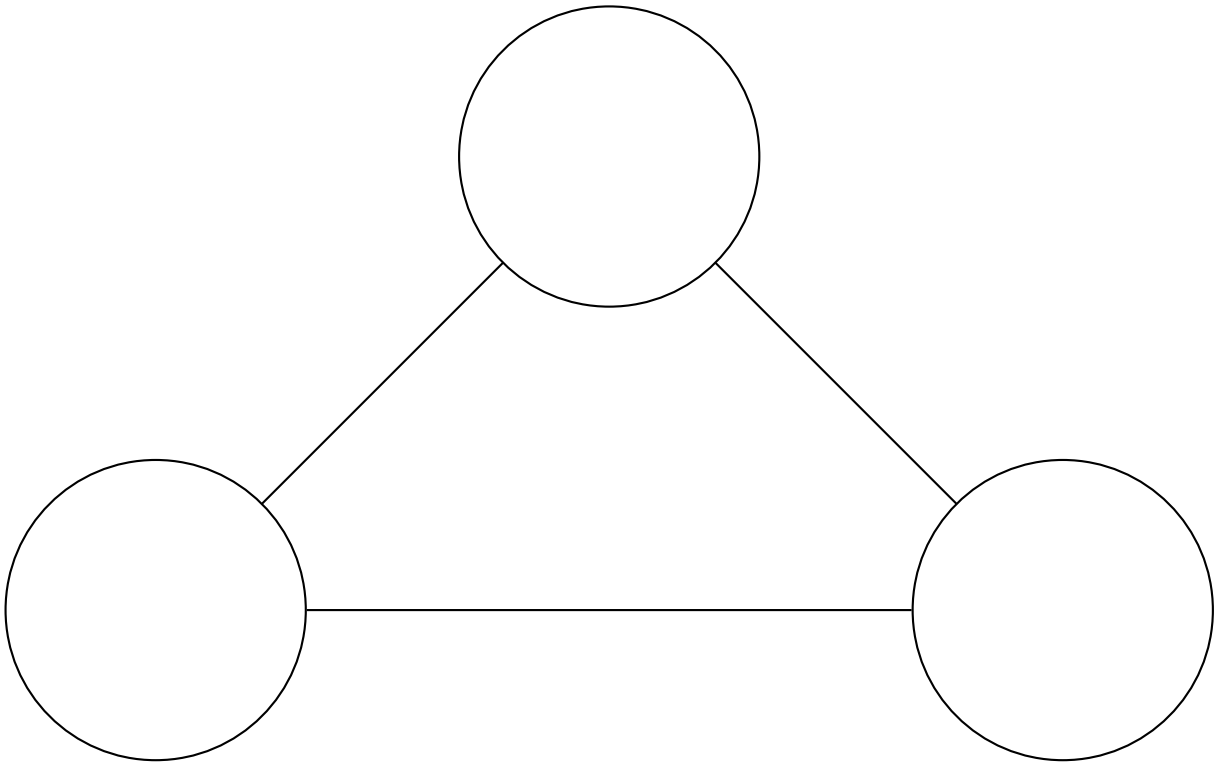
I. Questions and answer(s)

Question 1: Hitting time
(see e.g. #45: Maximum hitting time for random walks on graphs, G. Brightwell and P. Winkler, J. Random Structures and Algorithms 1:3 (1990), pp. 263-276)

A particle does a random walk on a connected graph G, starting at the vertex s; at each stage the particle chooses randomly from the $\deg(v)$ neighbors of the vertex v it currently occupies to decide which neighbor of v to go to next, until it finally arrives at the vertex t.

The *hitting time* from s to t (a random variable) is the number of steps the particle takes before it first reaches t.

Warm-up: The expected hitting time from one vertex to another in the triangle graph

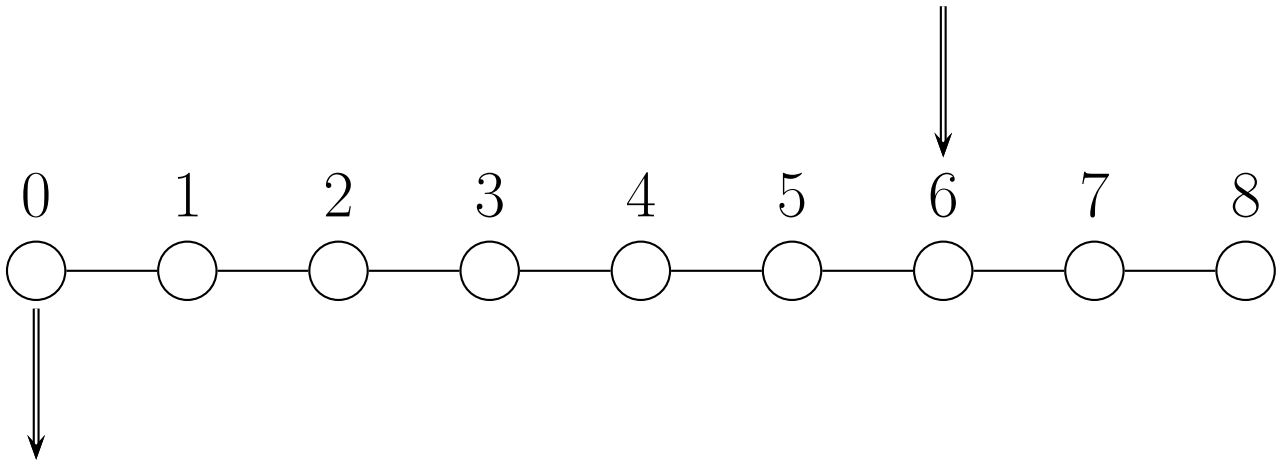


is ...

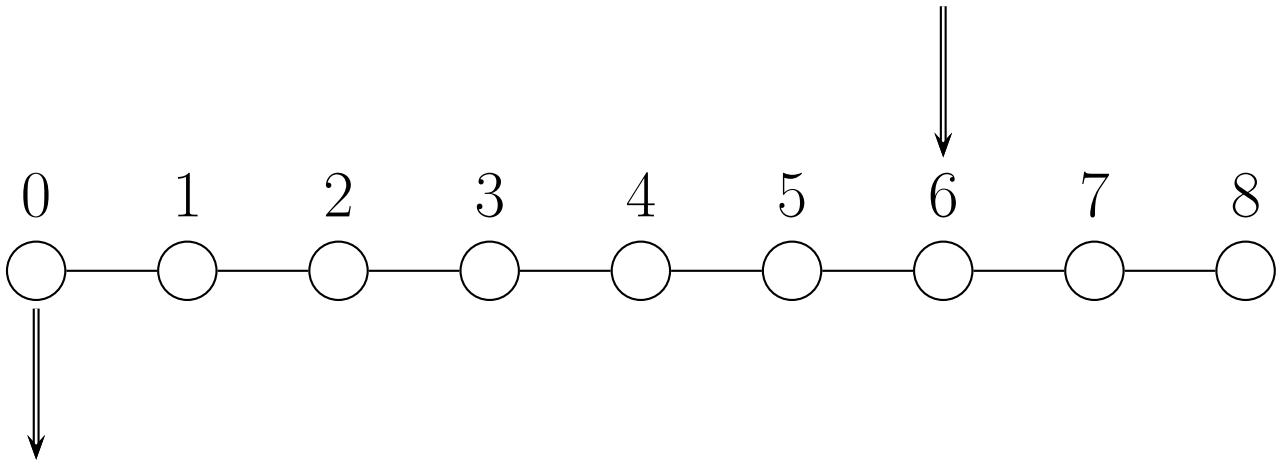
Warm-up: The expected hitting time from one vertex to another in the triangle graph is 2.

(This is just a geometric random variable with parameter $p = 1/2$, since at each stage you have a chance of $1/2$ of getting to the target vertex if you aren't there already.)

Question: What's the expected hitting time from 6 to 0 for random walk on the graph $G = \{0, 1, 2, ..., 8\}$ with an edge between vertices i and j precisely when $|i - j| = 1$)?



Question: What's the expected hitting time from 6 to 0 for random walk on the graph $G = \{0, 1, 2, ..., 8\}$ with an edge between vertices i and j precisely when $|i - j| = 1$)?



Answer: 60

Question 2: Chip-firing
(see e.g. #77: Mixing of random walks and other diffusions on a graph, L. Lovász and P. Winkler, Surveys in Combinatorics, 1995, P. Rowlinson (editor), 1995, pp. 119-154, London Math. Soc. Lecture Note Series 218).

Let G be a connected finite graph with designated source s and target t, with varying numbers of chips at its vertices.

If a vertex $v \neq t$ has $\deg(v)$ or more chips on it, we may "fire" the vertex; that is, we may remove $\deg(v)$ chips from it and distribute them evenly among the neighbors of v. If no such vertex v exists, we call the chip-configuration "stable".

Warm-up: Let G be the triangle graph. Put 4 chips on the $s$ource vertex s, 0 chips on the $t$arget vertex t, and 1 chip on the $u$ther vertex u.

s	t	u
4	0	1
2	1	2
0	2	3
1	3	1

Note that we get the same final state via another path:

s	t	u
4	0	1
2	1	2
3	2	0
1	3	1

"Abelian" (or Church-Rosser) property: If a chip-configuration on a finite connected graph with at least one target-node is not stable, then it can be made stable by repeated chip-firings, and the final configuration does not depend on the choice of which vertex to fire when.

Let $C_{\max}$ be the configuration in which each vertex $v \neq t$ has $\deg(v) - 1$ chips.

Question: If we start in the state $C_{\max}$ of the path-graph $G = \{0, 1, 2, ..., 8\}$ and add a chip at 6, with absorption at 0, how many chip-motions will occur before the system becomes stable?

"Abelian" (or Church-Rosser) property: If a chip-configuration on a finite connected graph with at least one target-node is not stable, then it can be made stable by repeated chip-firings, and the final configuration does not depend on the choice of which vertex to fire when.

Let $C_{\max}$ be the configuration in which each vertex $v \neq t$ has $\deg(v) - 1$ chips.

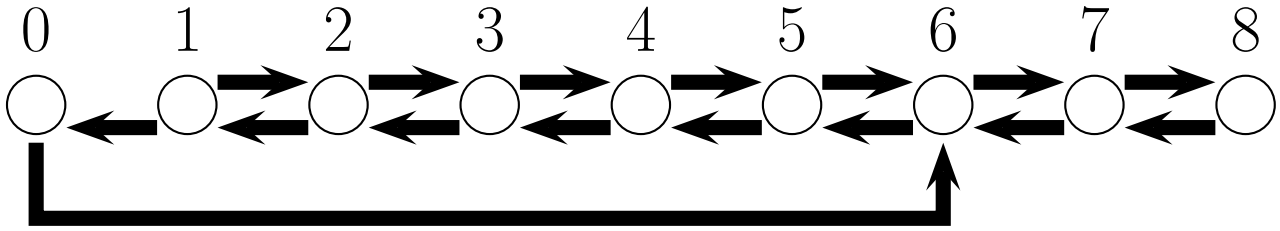
Question: If we start in the state $C_{\max}$ of the path-graph $G = \{0, 1, 2, ..., 8\}$ and add a chip at 6, with absorption at 0, how many chip-motions will occur before the system becomes stable?

Answer: 60

Question 3: Unicyclic sub-digraphs
(see e.g. Chip firing and the chromatic polynomial, by Norman Biggs and Peter Winkler; Centre for Discrete and Applicable Mathematics, technical report LSE-CDAM-97-03 (1997))

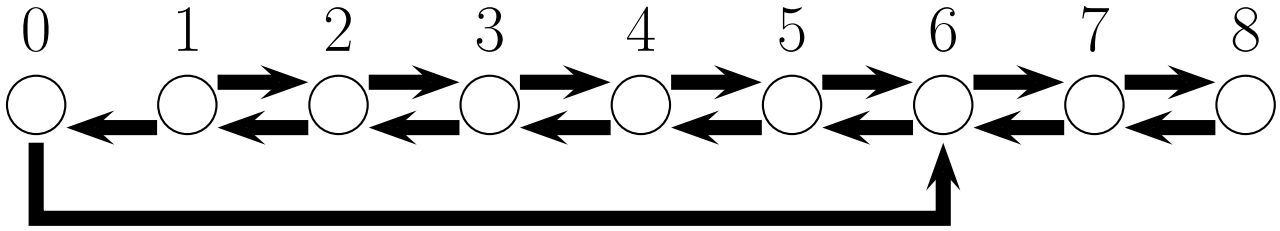
Given a directed graph D, say that a pair (v, U) consisting of a vertex v of D and a spanning subdigraph U of D is a *unicycle with root* v iff
(1) every vertex has out-degree 1 in U,
(2) U has a unique directed cycle, and
(3) v lies on that directed cycle.

Question: Let D be the digraph with vertex set $\{0, 1, 2, ..., 8\}$, with an arc from n to $n-1$ for $n = 1, 2, ..., 8$ and from n to $n+1$ for $n = 1, ..., 7$, and with an extra arc from 0 to 6.



There is a unique unicycle rooted at 0; how many other unicycles are there?

Question: Let D be the digraph with vertex set $\{0, 1, 2, ..., 8\}$, with an arc from n to $n-1$ for $n = 1, 2, ..., 8$ and from n to $n+1$ for $n = 1, ..., 7$, and with an extra arc from 0 to 6.



There is a unique unicycle rooted at 0; how many other unicycles are there?

Answer: 60

Question 4: Whirling tours

(see e.g. #112: On playing golf with two balls, I. Dumitriu, P. Tetali and P. Winkler, SIAM J. Disc. Math. 16:4 (2003), pp. 604-615)

Let T be any tree, possibly with loops. Fix a target vertex t, and let s be any other vertex. Order the edges (including loops) incident to each $u \neq t$ arbitrarily subject to the condition that the edge on the path from u to t must be the last edge incident to u. Now walk from s by choosing each exiting edge in round-robin fashion, in accordance with the edge-order at the current vertex, until t is reached. Call such a walk a "whirling tour".

Question: What is the length of a whirling tour on $G = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ that starts at $v = 6$ and ends at $t = 0$?

(Note: It turns out that the length of the tour in a tree T does not depend on the way the edges emanating from the individual vertices are ordered.)

Question: What is the length of a whirling tour on $G = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ that starts at $v = 6$ and ends at $t = 0$?

(Note: It turns out that the length of the tour in a tree T does not depend on the way the edges emanating from the individual vertices are ordered.)

The tour goes

67876567876545678765345678765432
345678765432123456787654321 0

Question: What is the length of a whirling tour on $G = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ that starts at $v = 6$ and ends at $t = 0$?

(Note: It turns out that the length of the tour in a tree T does not depend on the way the edges emanating from the individual vertices are ordered.)

The tour goes

6787656787654567876543456787654323456787654321234567876543210

Answer: 60

Question 1: Hitting time (Winkler, 1990)

Question 2: Chip-firing (Winkler, 1995)

Question 3: Unicyclic sub-digraphs (Winkler, 1997)

Question 4: Whirling tours (Winkler, 2003)

Question 5: How old is the person we are honoring today?

Question 6: Why do all these questions have the same answer?

Coincidence? ...

Evidence that it's not a coincidence:

Question 1: $60 = 2+4+6+8+10+12+12+6$ (expected number of times each vertex is visited under random walk)

Question 2: $60 = 2+4+6+8+10+12+12+6$ (number of chips that leave each vertex during relaxation)

Question 3: $60 = 2+4+6+8+10+12+12+6$ (number of unicycles rooted at each vertex)

Question 4: $60 = 2+4+6+8+10+12+12+6$ (number of times each vertex is visited during the whirling tour)

II. Chip-firing and hitting times

Let G be a connected finite graph with designated source s and target t.

Following Dhar, define an elementary excitation-relaxation operation as the operation that adds a single chip at s to a stable configuration and then performs chip-firing until the configuation becomes stable again.

This operation had been studied earlier by Arthur Engel. (A slight variant was later studied by Norman Biggs.)

Theorem (Engel): Let $C_{\max}$ be the chip configuration in which each vertex $v \neq t$ has $\deg(v) - 1$ chips on it. If we perform the elementary excitation-relaxation operation at s enough times (m, say), the chip configuration $C_{\max}$ will recur. If M denotes the number of chip-motions that occur during this process, then the ratio M/m (the average number of chip-motions per excitation-relaxation operation) is equal to the expected hitting time from s to t.

Example: The triangle graph.

As we saw in the warm-up to Question 1, the expected hitting time from s to t is 2.

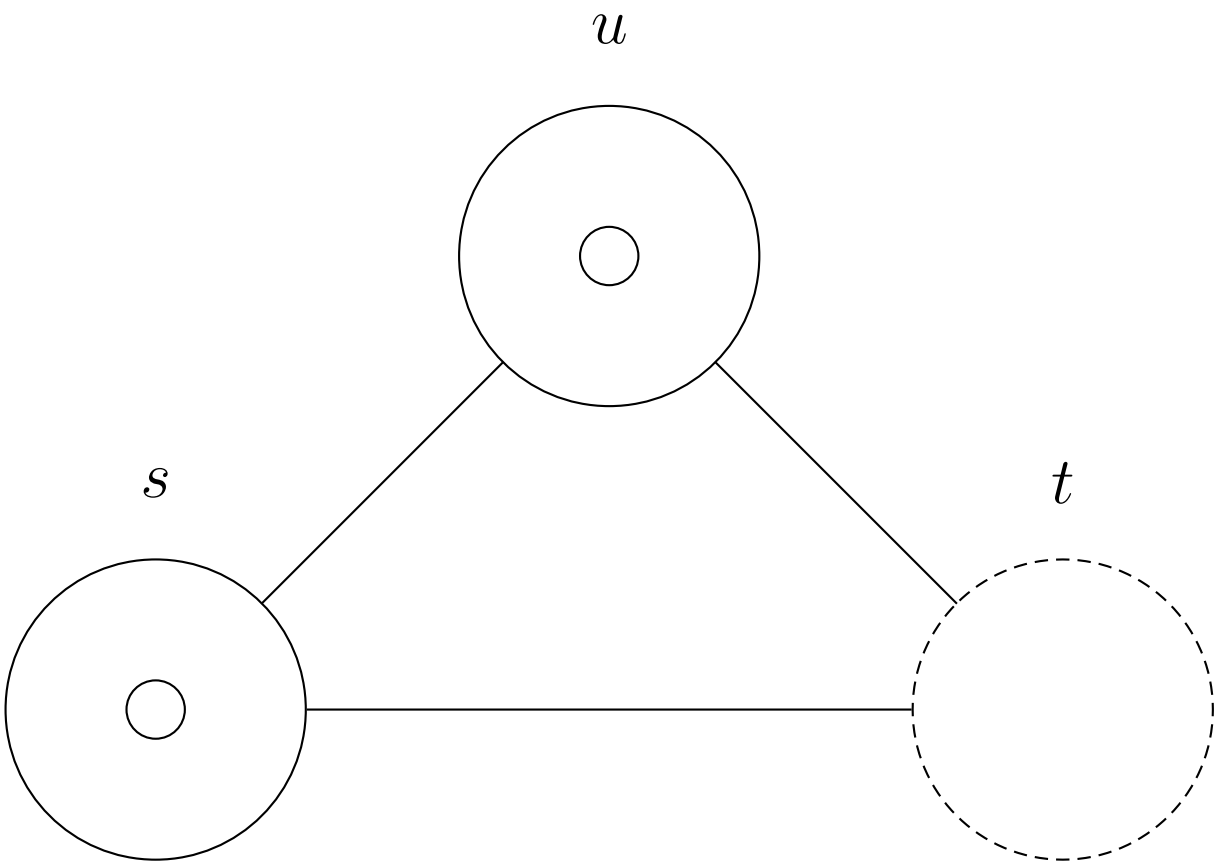
Using the work we did in the warm-up to Question 2, we can check that when we add 3 chips to the system, the number of chip-motions is 6.

This checks: $6/3 = 2$.

(N.B.: If we add chips one at a time at s, the number of chip-motions cycles between the values 4, 2, and 0, with period 3.)

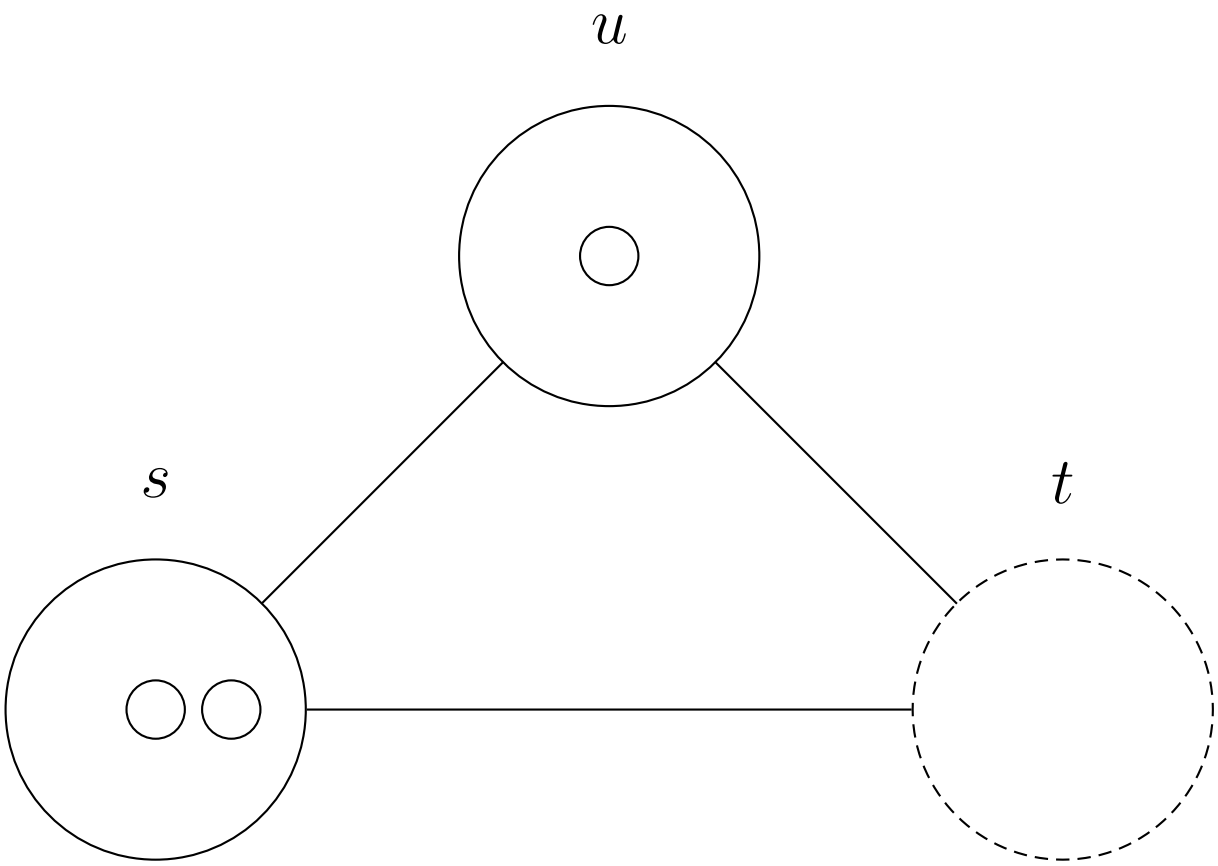
Let's see this in detail:

Computing the expected hitting time via chip-firing:



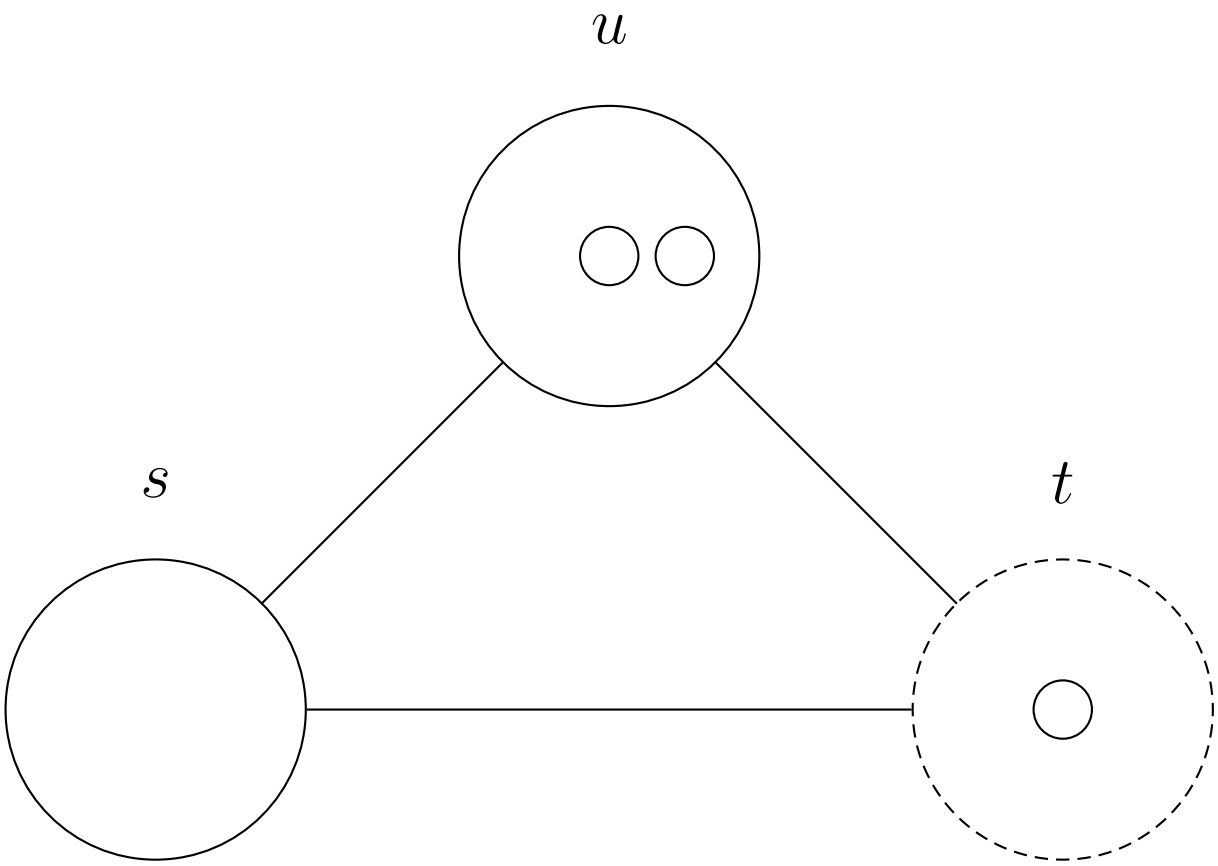
Moves so far: 0

Computing the expected hitting time via chip-firing:



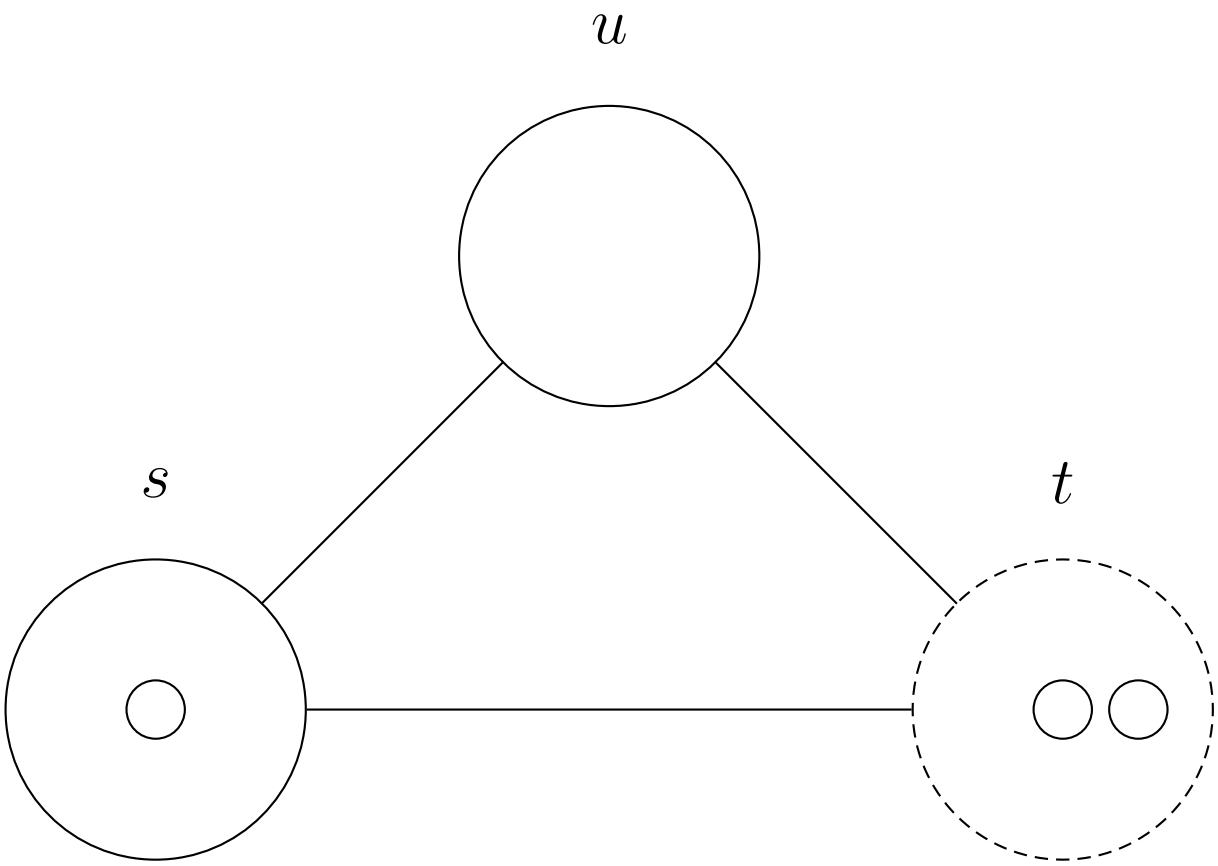
Moves so far: 0

Computing the expected hitting time via chip-firing:



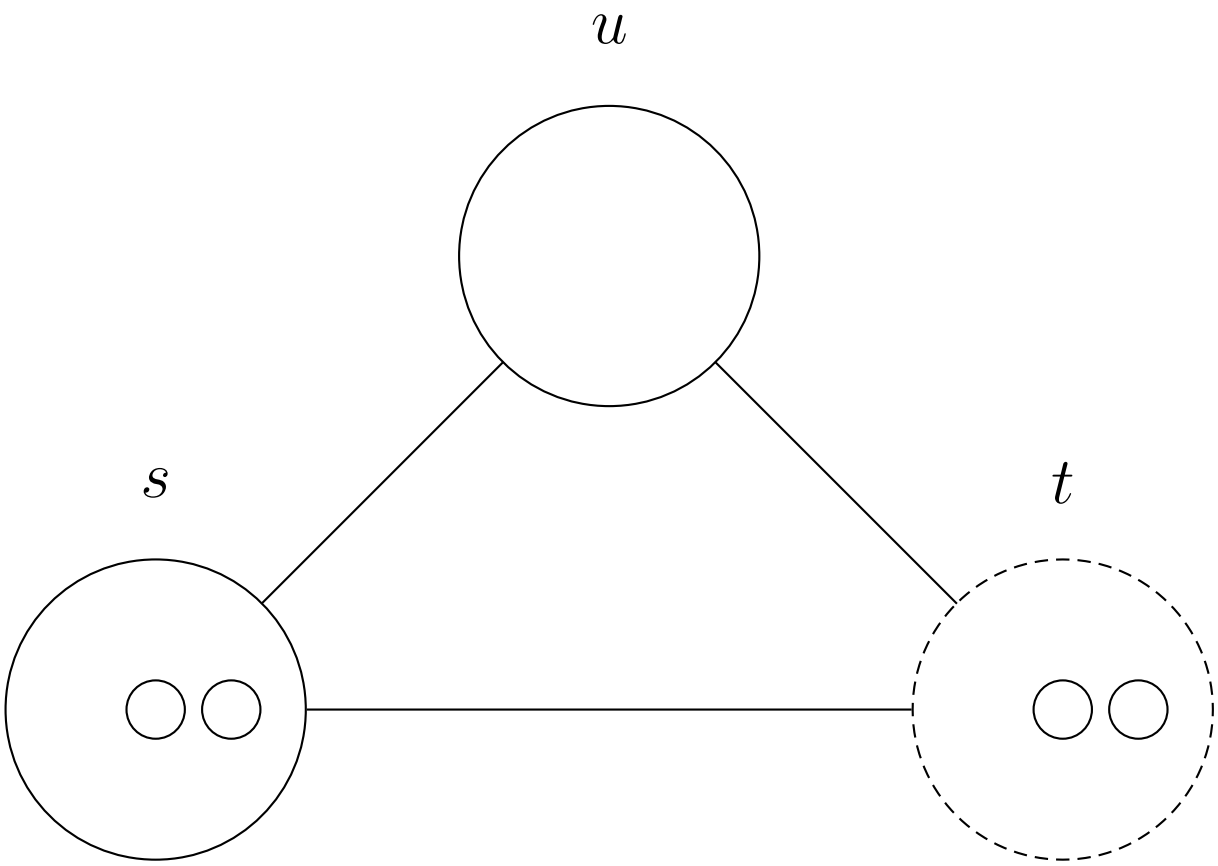
Moves so far: 2

Computing the expected hitting time via chip-firing:



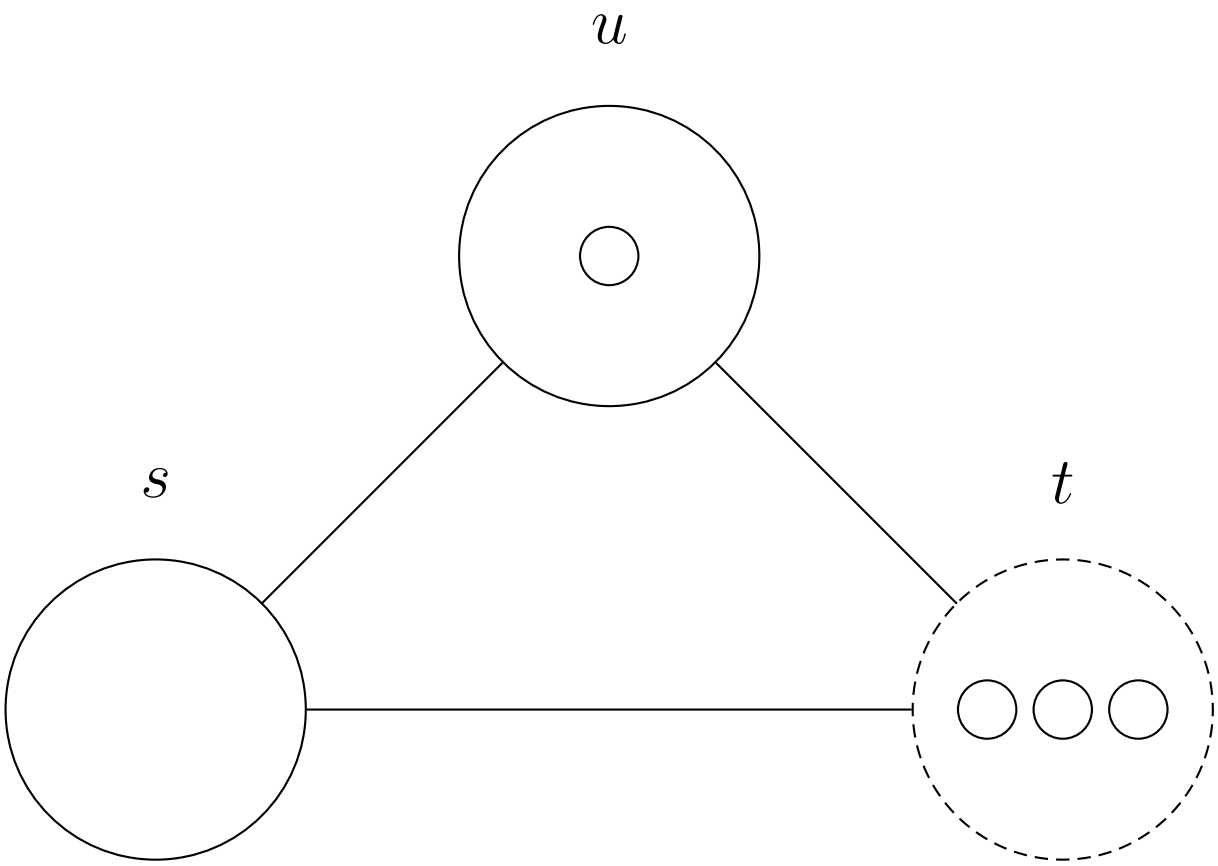
Moves so far: 4 (end of first stage)

Computing the expected hitting time via chip-firing:



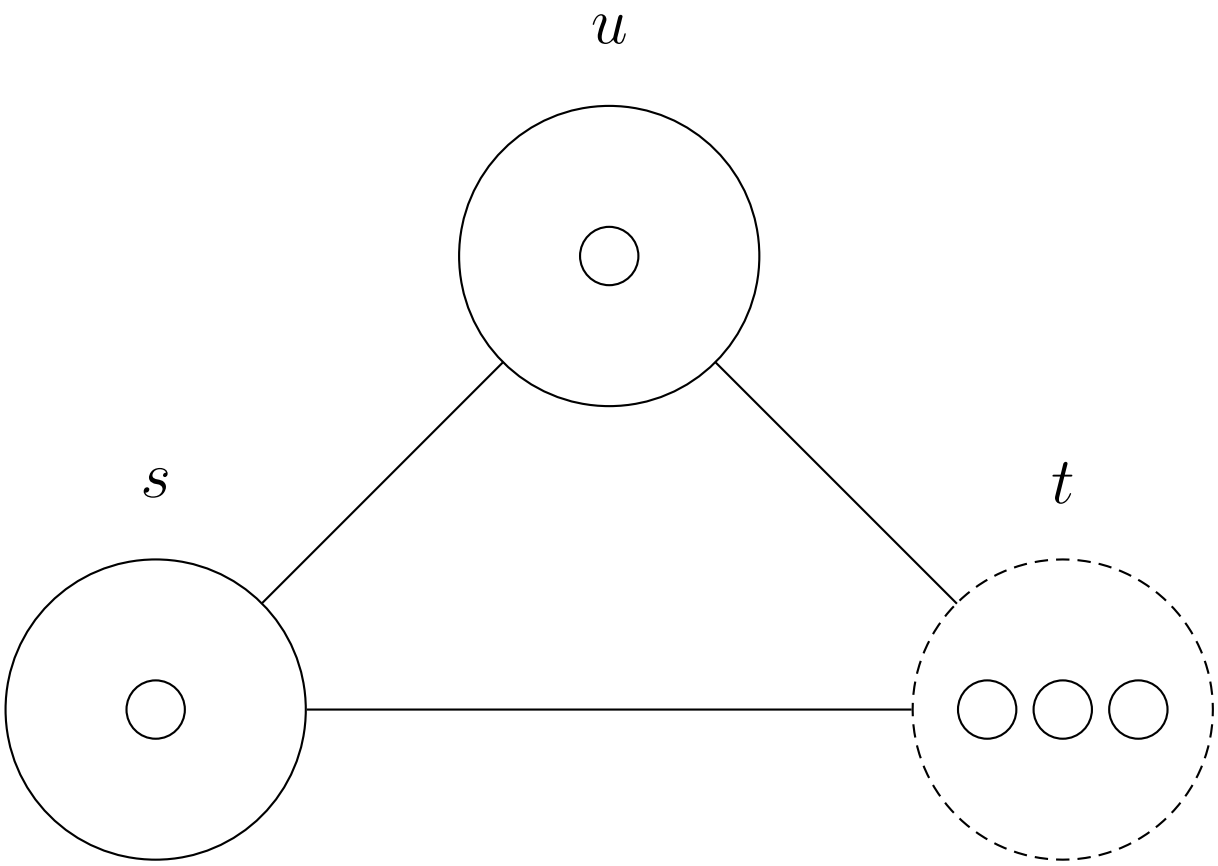
Moves so far: 4 (end of first stage)
Moves so far: 0

Computing the expected hitting time via chip-firing:



Moves so far: 4 (end of first stage)
Moves so far: 2 (end of second stage)

Computing the expected hitting time via chip-firing:



Moves so far: 4 (end of first stage)
Moves so far: 2 (end of second stage)
Moves so far: 0 (end of third stage)

Why must $C_{\max}$ recur?

A sketch of a proof (first found by Doyle; rediscovered by Björner and Lovász):

(1) If we ignore the occupancy of the target-vertex, *some* state S must recur (since there are only finitely many states).

(2) Whatever sequence of firings brings S back to itself must also bring $C_{\max}$ back to itself (since a vertex capable of firing remains capable of firing if it acquires some extra chips).

Why does M/m converge to the expected hitting time?

Intuition: By the abelian property, we get the same result if, we instead adding one chip, relaxing the system, adding another chip, relaxing the system, etc., some large number N times, we load N chips into the system at the start and then relax all of them.

So suppose we put a huge number N of chips on s and start to relax the system. If we follow the destiny of a randomly chosen chip, it very nearly does a random walk, with deviations from true randomness that become negligible as N goes to infinity.

Real proof: Doyle uses detailed balance.

Here's a sketch of an economical proof: When a chip is at location u its value is $h(u)$ dollars, where $h(u)$ is the expected hitting time from u to the target.

For simplicity, suppose there are vertices v, w with $p_{u,v} = p_{u,w} = 1/2$.

Then

$$h(u) = 1 + (h(v) + h(w))/2$$

so

$$2h(u) = 2 + h(v) + h(w).$$

Moving chips costs you money (1 dollar per chip move), and you end up broke ($h(t) = 0$), so the amount of money you started out with ($h(s)$ per chip) equals the amount you spent on transporting chips (the average number of moves per chip).

When the graph is a tree, the number of chips m that we need to add to get back to $C_{\max}$ is 1; why?

Certain stable chip-configurations C (like $C_{\max}$) have the property that, given any stable chip-configuration C', it's possible to get from C' to C via elementary excitation-relaxation operations. We call them "critical" configurations.

As Dhar observed, the critical configurations form a group under the operation of amalgamating chips (on a vertex-by-vertex basis) and then relaxing the system until it's stable again. This is the "critical group" of the graph G with target t, first analyzed by Gabrielov.

The identity element of this group does not always have a simple representative; e.g., if G is the 198-by-198 square grid with a single extra vertex t adjacent to the boundary vertices, the identity element is as shown in `jamespropp.org/sandpile-identity.pdf` (lifted from "Cellular automata and self-organized criticality" by Michael Creutz; see arXiv: `hep-lat/9611017`).

Fact: The critical group is the cokernel of the Laplacian of G (more precisely, the cokernel of the minor obtained by deleting the row and column associated with t).

Example: For the triangle graph, the Laplacian is

$$\begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

whose minor

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

has cokernel isomorphic to the 3-element group.

Fact: The order of the critical group equals the number of spanning trees of G.

(This follows from the earlier fact by the Matrix Tree Theorem; Biggs and Winkler gave a bijective proof.)

Wake-up call: What is the critical group of a tree?

How many spanning trees does a tree have?

In the case where G is itself a tree, the order of the critical group is 1.

This implies that if you start with the critical configuration $C_{\max}$ and perform a single elementary excitation-relaxation operation anywhere (using any vertex other than the target as the source), you get back $C_{\max}$ immediately. Hence $m = 1$.

Engel developed chip-firing in the 1970s as a pedagogical tool (the "stochastic abacus") for allowing fourth graders to solve problems in discrete probability theory. Fourth-graders can't do algebra (let alone linear algebra), so they need a method of simulation that mimics the system they're studying (random walk) to the extent of giving the correct answer to questions like "How long does it take on average for the mouse to reach the cheese?"

Chip-firing was re-invented in the 1980s as a toy model of self-organized criticality; in that context, it's called the abelian sandpile model.

Chip-firing was also re-invented by Joel Spencer in the 1980s, in connection with "pusher-and-chooser" games.

III. Rotor-routers and hitting times

A different way to derandomize hitting-time problems (and many other kinds of problems in discrete probability) is via rotor-routers.

Rotor-router walk was originally invented in the 1990s as yet another kind of self-organizing system (not for its relevance to random walk), under the name "Eulerian walkers model"; it was re-invented by me a few years later as an outgrowth of my interest in Engel's work as well as work of Diaconis and Fulton.

For the case of a particle doing a walk on a connected graph G, with no source or target, assign some cyclic ordering to the edges emanating from each vertex v. The rotor-router rule is that when the particle visits a particular vertex for the nth time, it leaves along the nth edge (interpreting n modulo the degree of v).

Represent the state of the system by a collection of arcs, one for each vertex, where the arc at v indicates the direction travelled by the particle the last time it visited v. (If the particle hasn't visited v yet, the arc points in the direction of the "0th edge" in the cyclic ordering of edges incident to v, modulo the degree of v.)

The rotor-router rule can be re-stated as follows: At each step, the arc at the current vertex v ROTATES, and the particle at v is ROUTED in the direction indicated by the new arc.

Easy claim: Once every vertex has been visited, the arcs form a unicycle rooted at the current vertex. This remains true forever after.

Proof: If w lies on a cycle that doesn't contain the current vertex, then you can show that some number of steps after its most recent visit to w, the particle was at w again, which is a contradiction.

Claim: Once the system has left behind its transient (non-unicyclic) states and entered into the recurrent (unicyclic) basin of attraction, its behavior becomes periodic, and the frequency with which each vertex is visited is proportional to its steady-state probability under random walk.

Generalizing this to infinite graphs, or biased walks, or Markov chains in which the transition probabilities need not be rational, is a subject of ongoing research.

It has the potential to be useful, because such "quasirandom" methods of simulating random systems can sometimes give us the information we want without all the noise inherent in random simulation.

Areas where rotor-routing seems to be especially promising are models of aggregation and erosion. (Maybe percolation too!)

`jamespropp.org/quasirandom.html` has lots of links about this.

Some problems from this nascent theory make for fun puzzles with a variety of different levels of difficulty, because of the way in which solving a problem about the long-term behavior of a rotor-router system calls for the solver to find quantities associated with the system that are invariant throughout the system's evolution.

These invariants often are associated with non-standard systems for representing numbers. In these systems (e.g., a base-two system with the digits 0, 1, and 2), there is more than one way to represent a given number. Indeed, the state of the rotors can be seen as a succession of different representations of the invariant quantity associated with the system.

For one fun example, see p. 82 of Winkler's book of puzzles *Mathematical Puzzles: A Connoisseur's Collection.*

For another example, see Michael Kleber's Intelligencer article "Goldbug Variations" at `http://people.brandeis.edu/~kleber/Papers/rotor.pdf`.

Variants of rotor-routing make for fun reachability puzzles (given a specified starting configuration, how to you get to a specified target configuration?), as in the children's toy Think a Dot, in which marbles are routed by flip-flops.

Here's an example of a harder "puzzle":

Theorem (Propp): If a particle does N stages of rotor-router walk on Z^2, starting from $(0, 0)$ and getting absorbed when it arrives at either (0,0) or (1,1), then the number of times it gets absorbed at (1,1) ("success") differs from pN by at most a constant times $\log N$, where $p = \pi/8$ is the probability of success under random walk.

Indeed, $\log N$ may be overly pessimistic: simulation (see e.g. `jamespropp.org/rotor-router-1.0/`) shows that, for over *half* of the integers N between 1 and 10^4, the number of successes in the first N stages of rotor-walk is the *closest integer* to pN (that is, $\pm \log N$ can be replaced by $\pm 1/2$).

With random simulation, the discrepancy would be $\sqrt{N}$ instead of $\log N$.

With chip-firing, the discrepancy would be $N/\log N$.

Unlike chip-firing, rotor-routing applies to simulation of random walk in which transition-probabilities are irrational.

But today we're talking about hitting time for (unbiased) random walk on finite graphs.

To use rotor-routers for computing hitting times, let particles leave the system at the target and then re-enter at the source.

Define a stage as the process of the particle leaving the source, arriving at the target, and then returning to the source, and define the length of the stage as the number of steps minus 1 (we don't want the transition from the target to the source to count as a step).

Once each vertex has been visited, the state of the system is always a unicycle, and at the moment when the system arrives at the target (the end of one stage and the start of the next), it is a unicycle rooted at the target.

Theorem (Levine and Propp): Once the system becomes unicyclic, its behavior enters an infinite loop.

Theorem (Holroyd and Propp): During one period of the infinite loop, the average duration of the stages is equal to the expected hitting time from s to t for random walk on G.

Example: The triangle graph, with rotors initially pointing from s to u and from u to s.

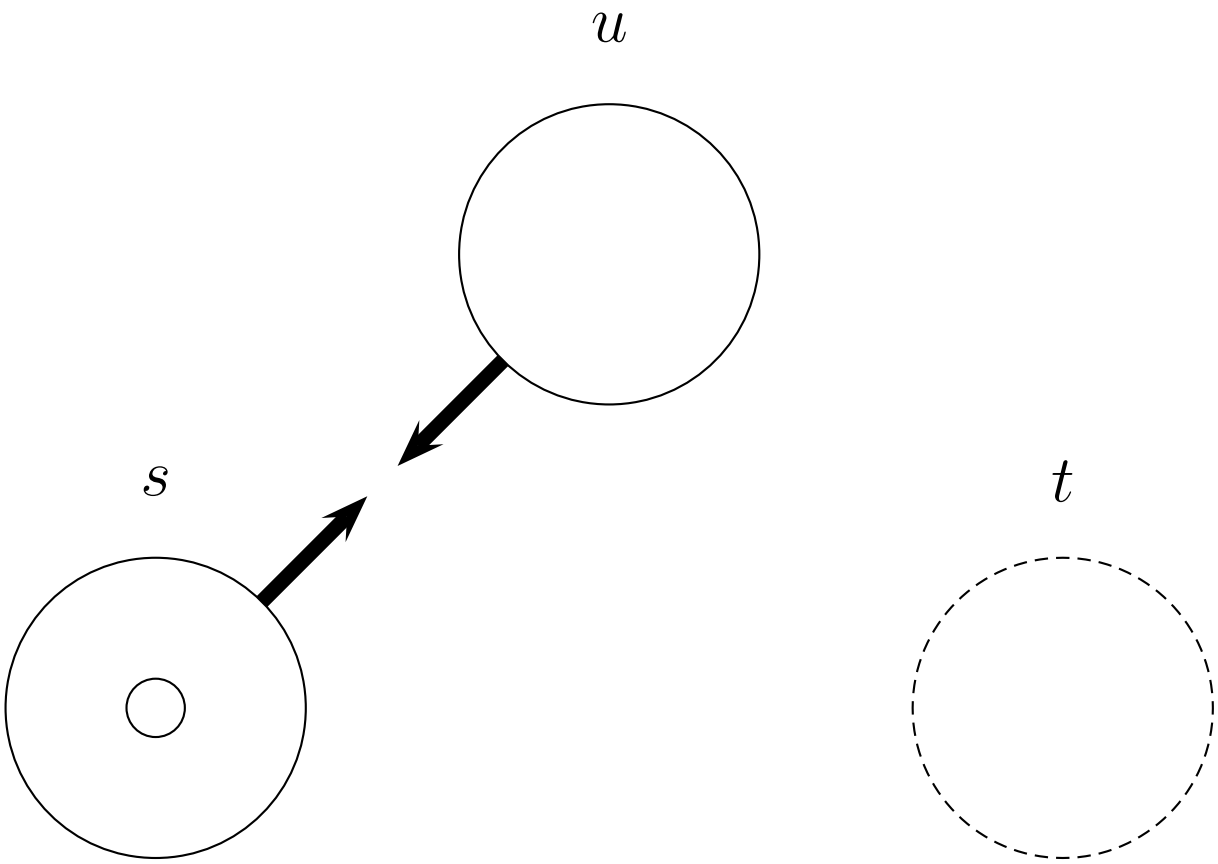
The stages have lengths 1,2,1,3,1,3,1,3,....

Once the system finishes with its transient behavior, it enters a period of length 2, where the number of steps taken by each particle to get from s to t alternates between 1 and 3.

Over one period, the average duration of a stage is $(1 + 3)/2 = 2$.

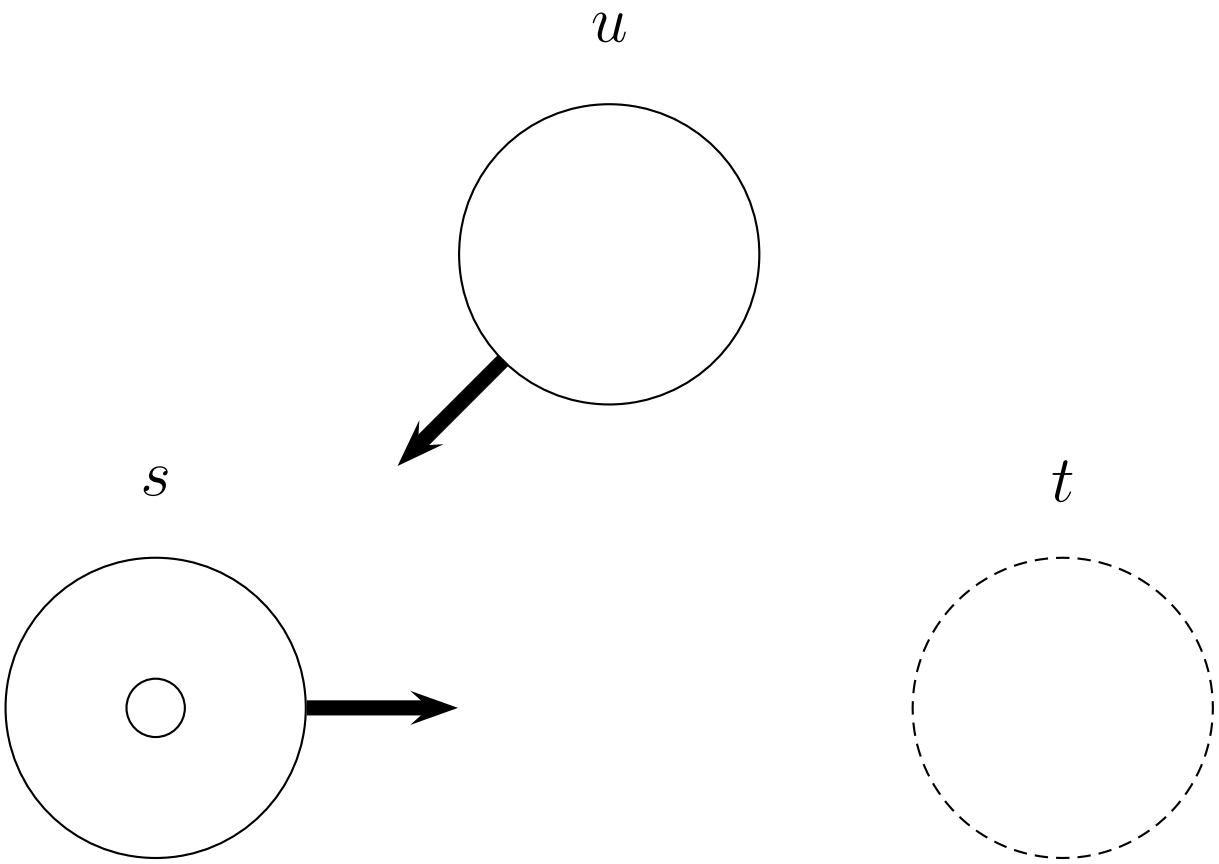
Let's see this in detail:

Rotor-routing on a triangle:



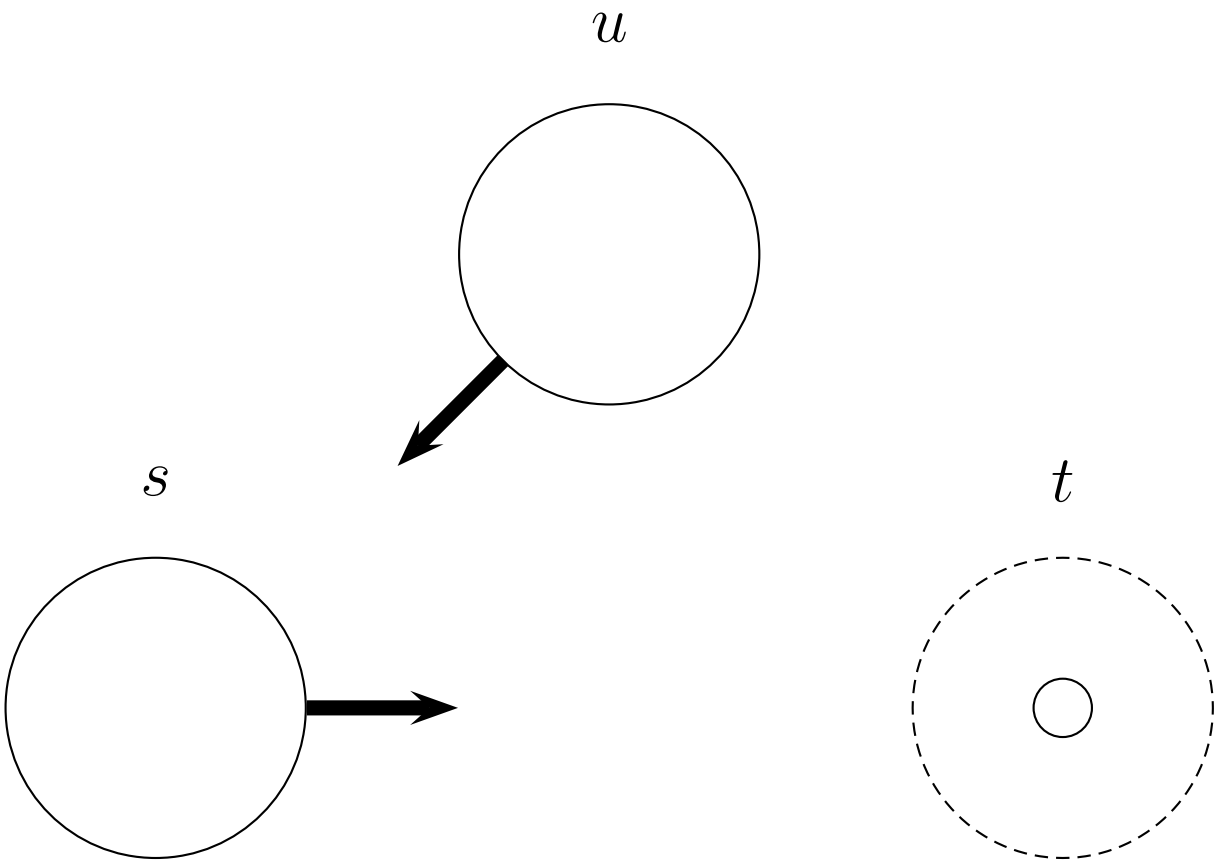
Steps so far: 0

Rotor-routing on a triangle:



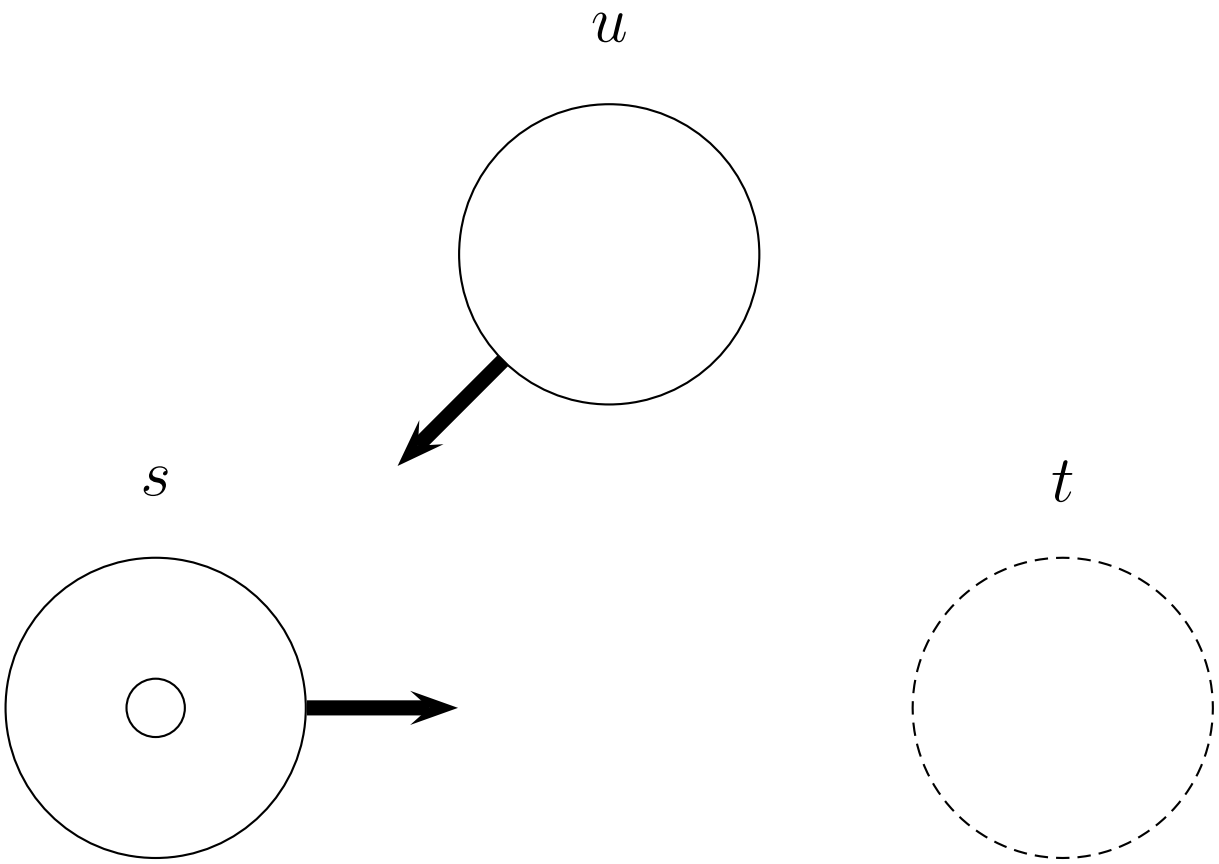
Steps so far: 0

Rotor-routing on a triangle:



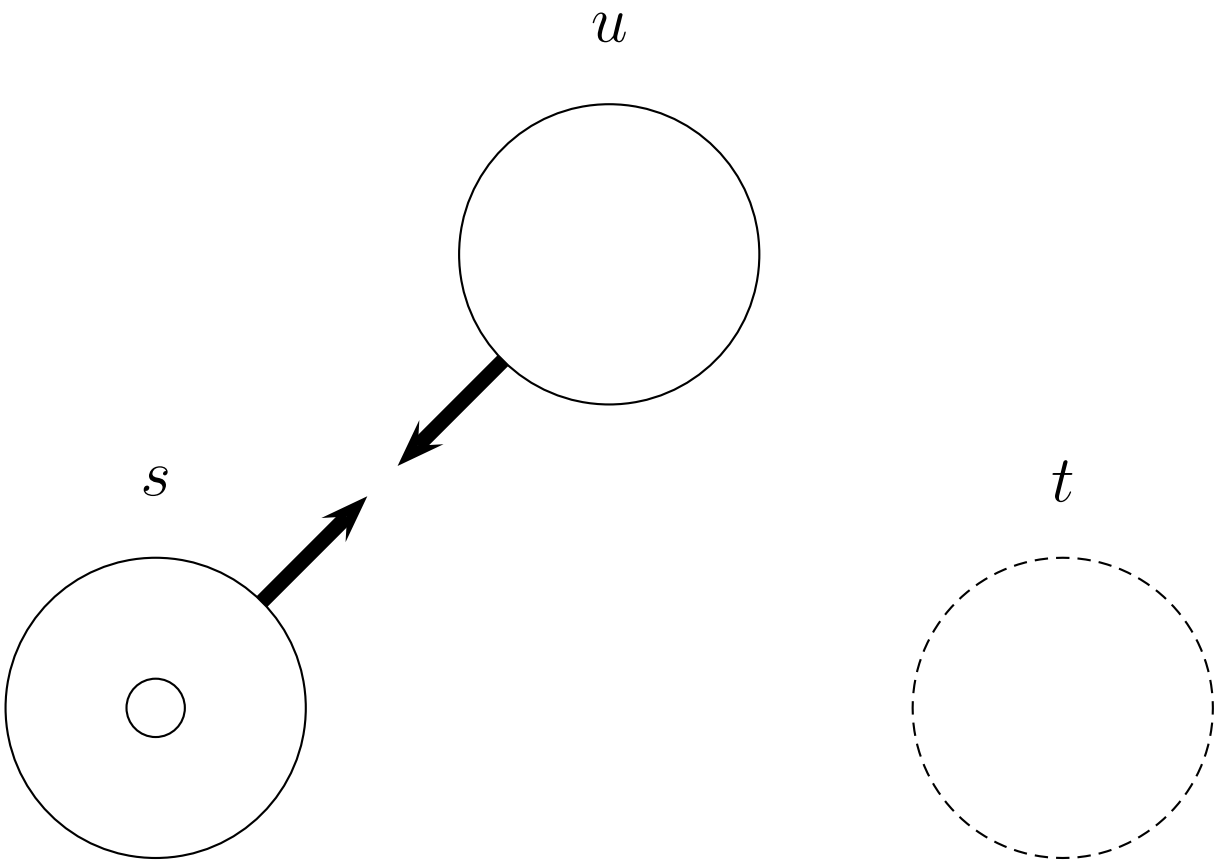
Steps so far: 1 (end of first stage)

Rotor-routing on a triangle:



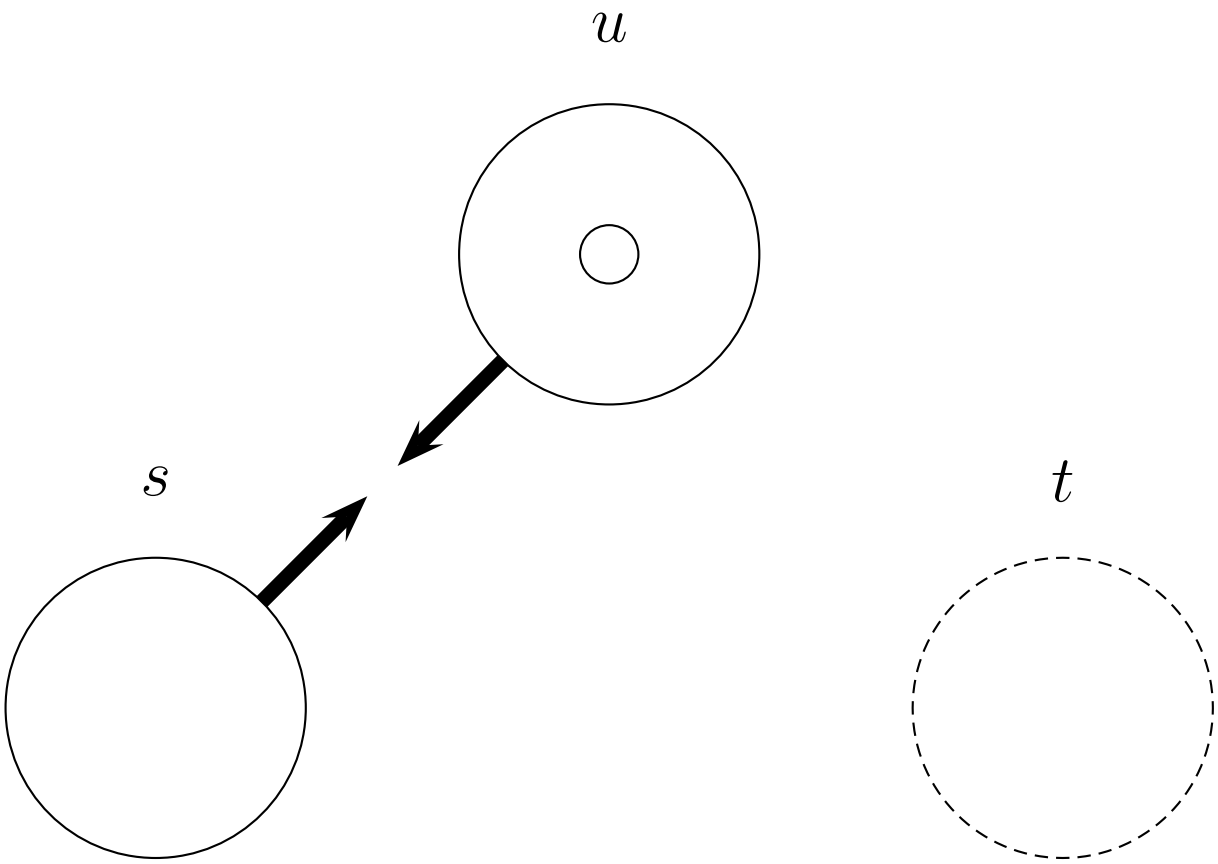
Steps so far: 1 (end of first stage)
Steps so far: 0

Rotor-routing on a triangle:



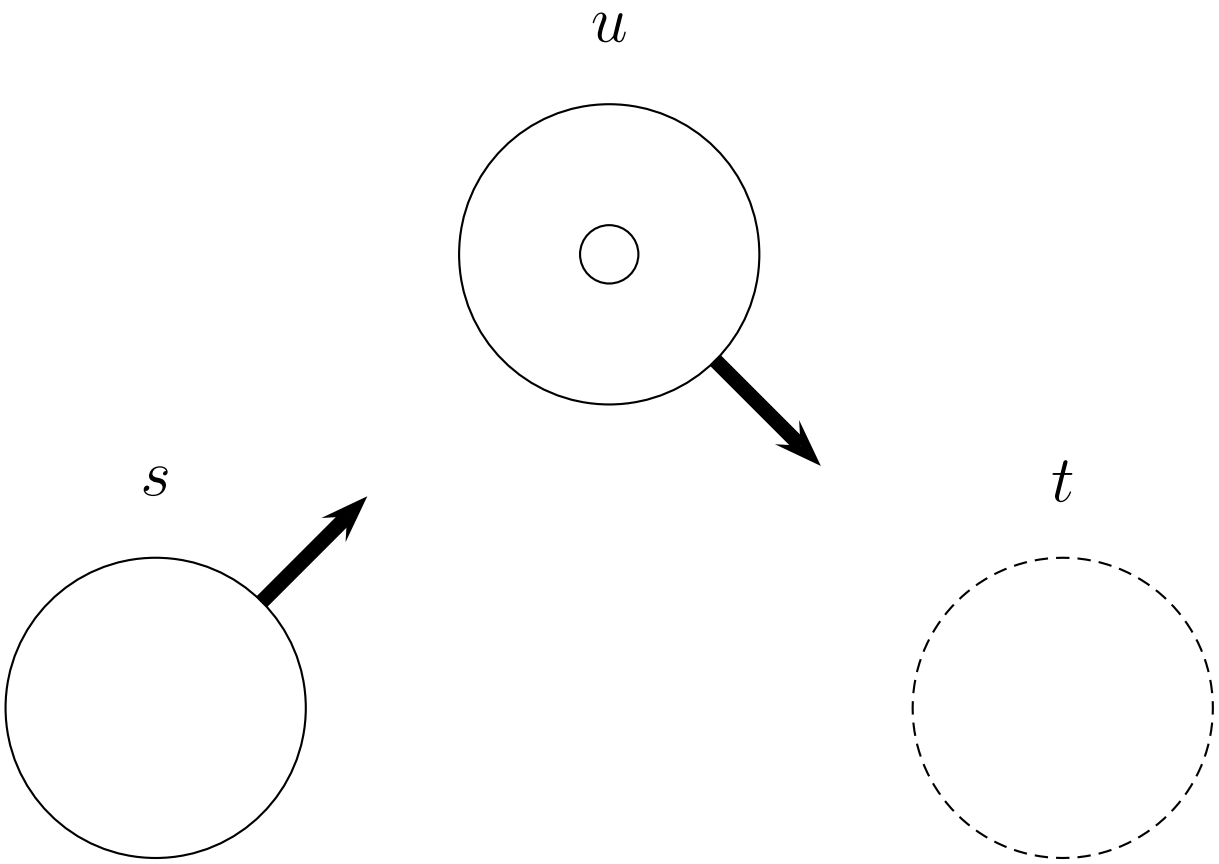
Steps so far: 1 (end of first stage)
Steps so far: 0

Rotor-routing on a triangle:



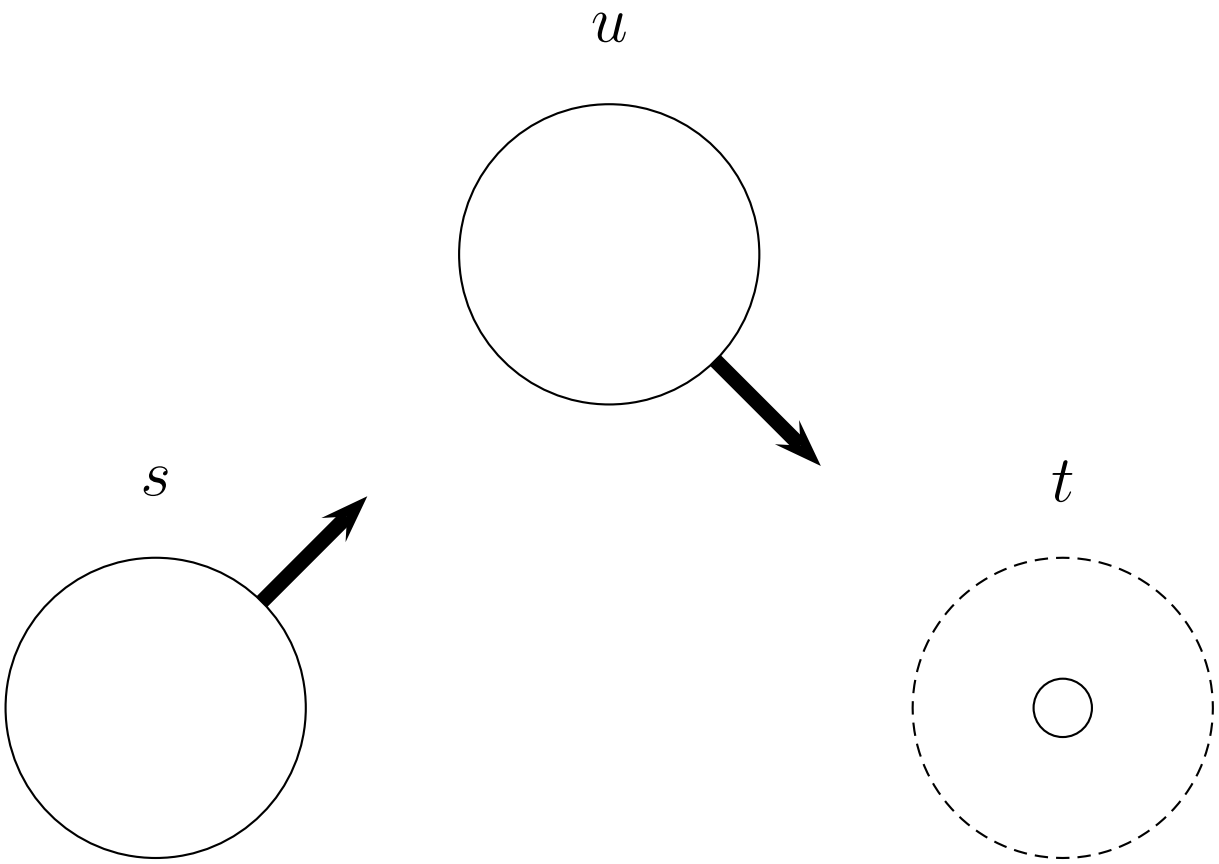
Steps so far: 1 (end of first stage)
Steps so far: 1

Rotor-routing on a triangle:



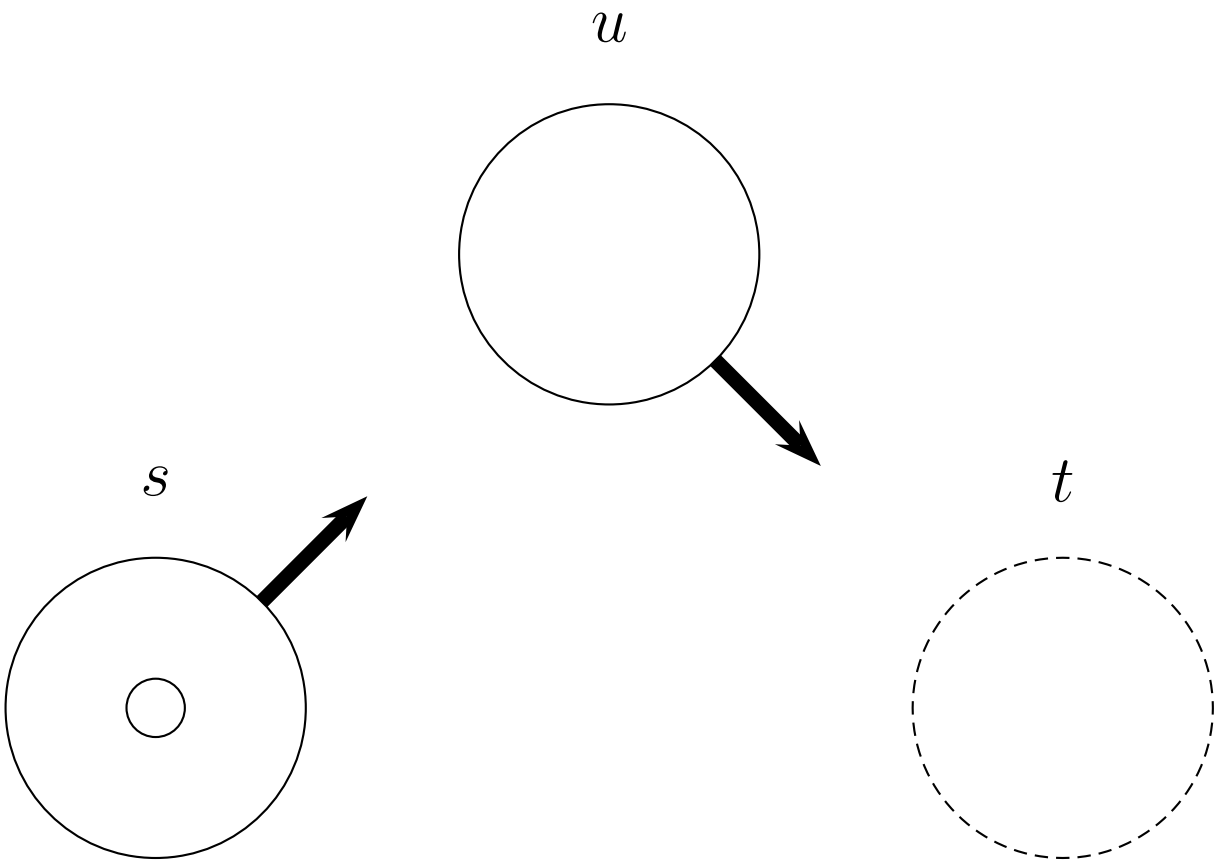
Steps so far: 1 (end of first stage)
Steps so far: 1

Rotor-routing on a triangle:



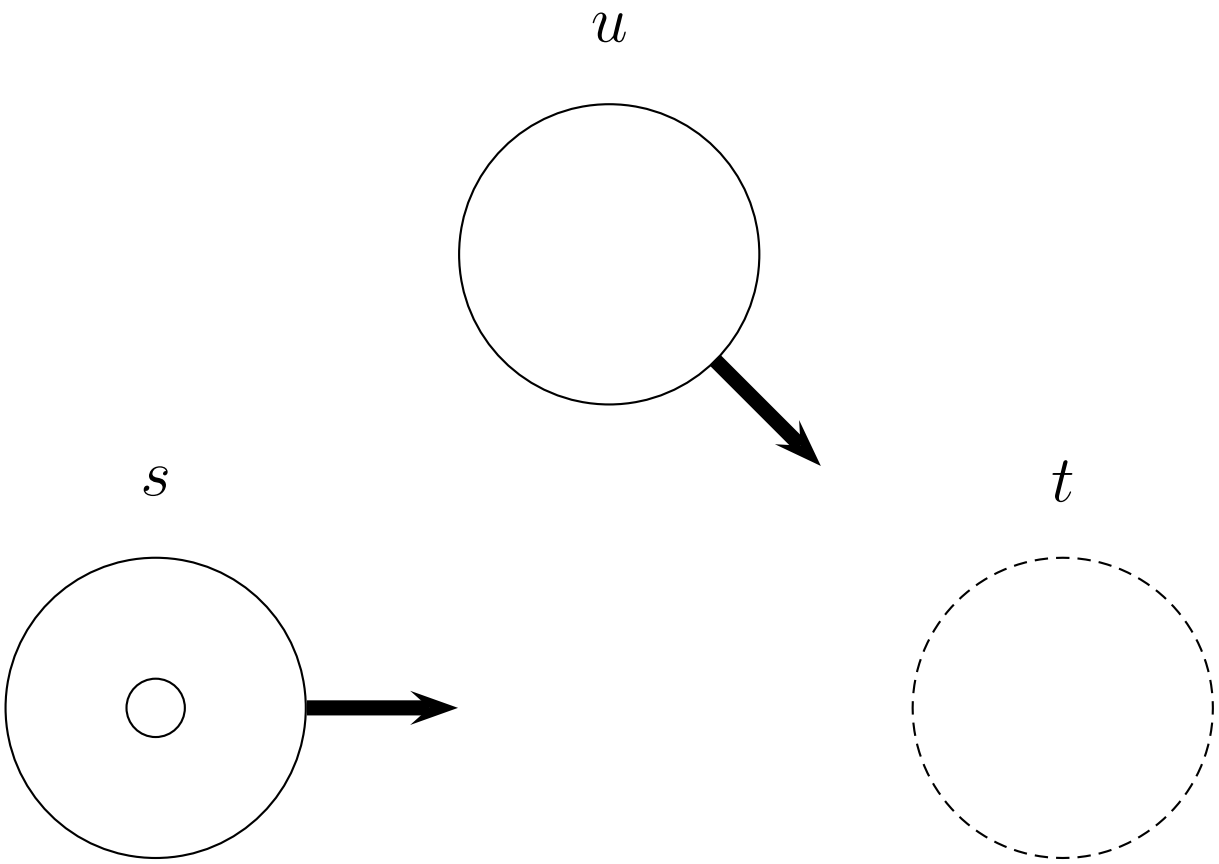
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)

Rotor-routing on a triangle:



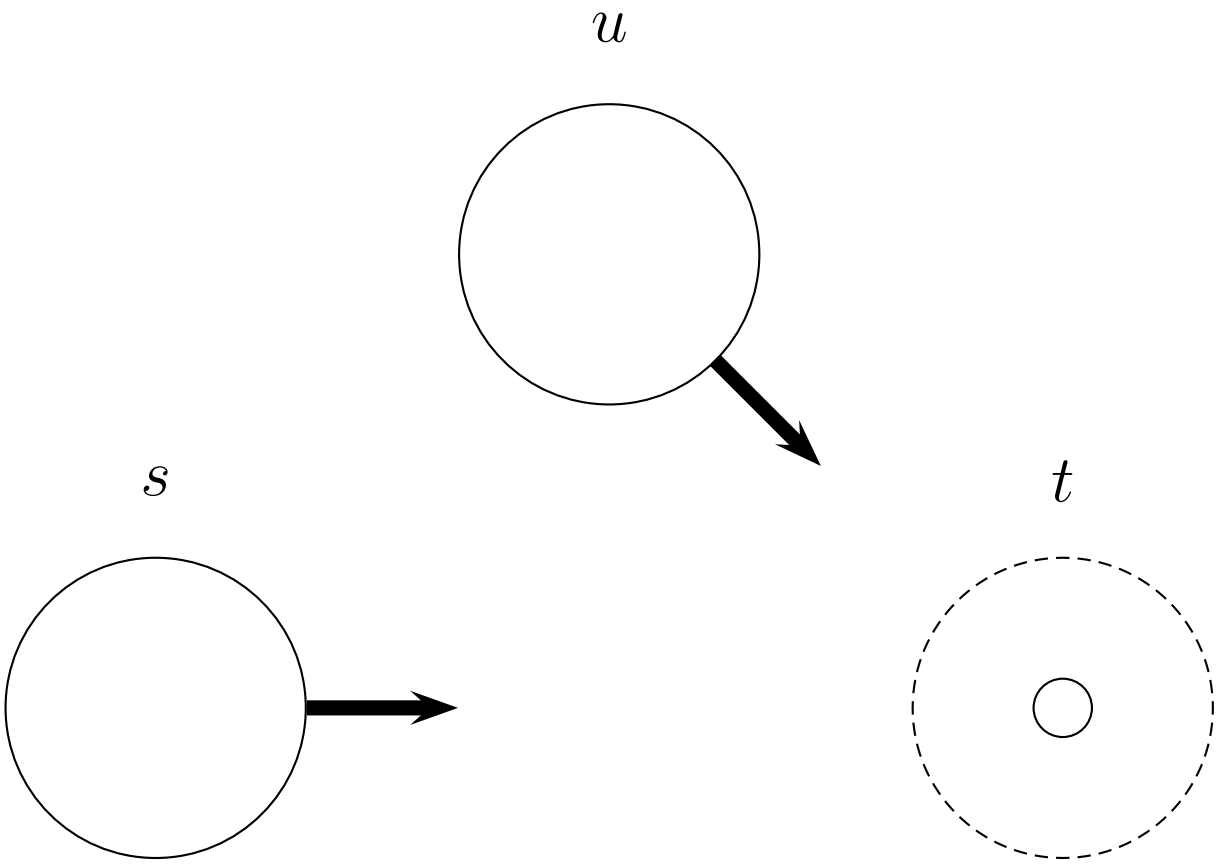
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 0

Rotor-routing on a triangle:



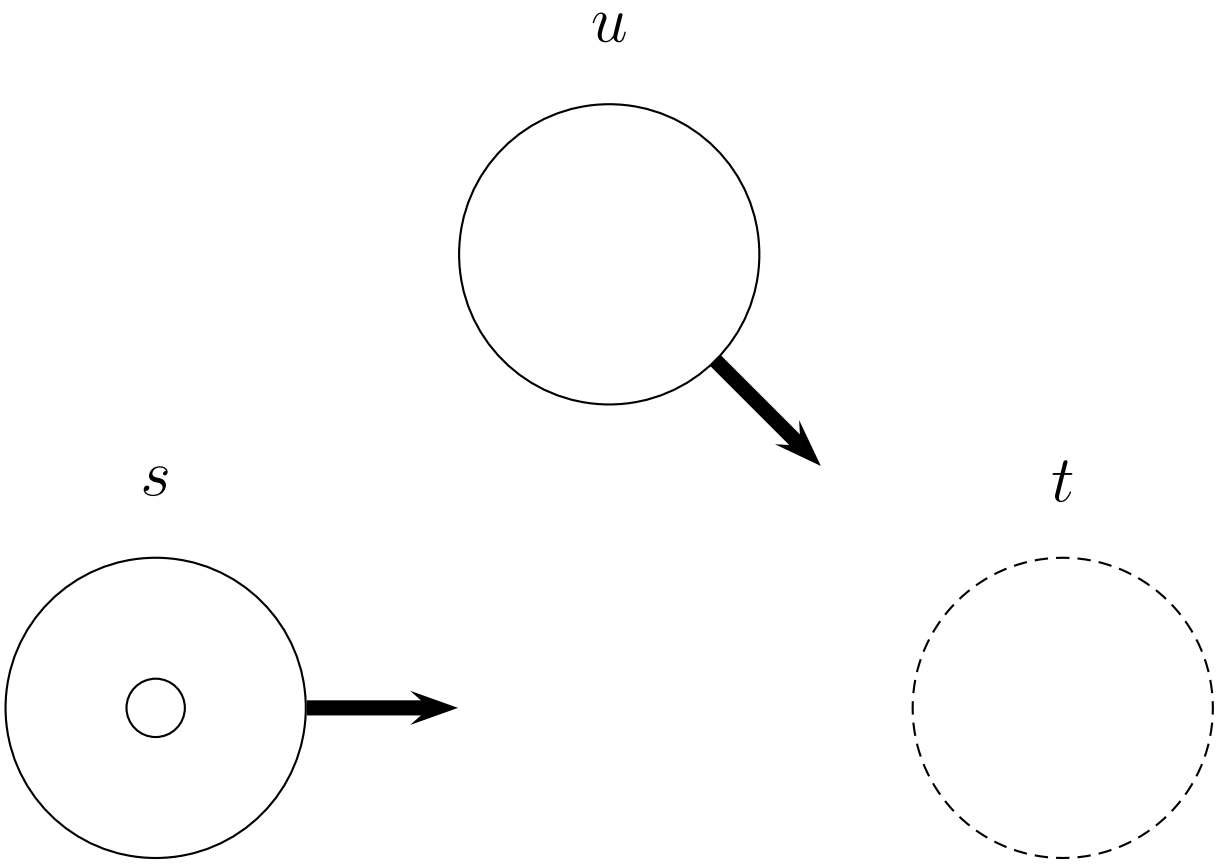
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 0

Rotor-routing on a triangle:



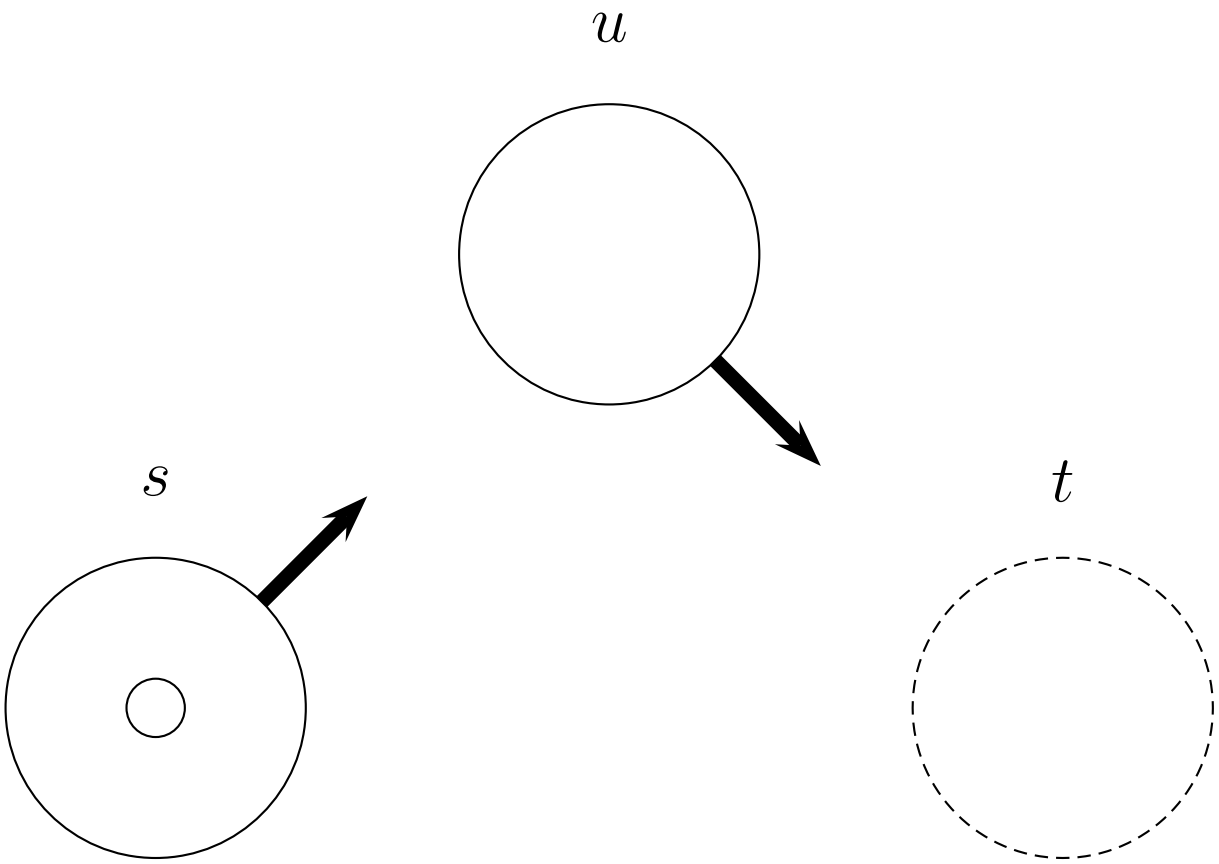
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)

Rotor-routing on a triangle:



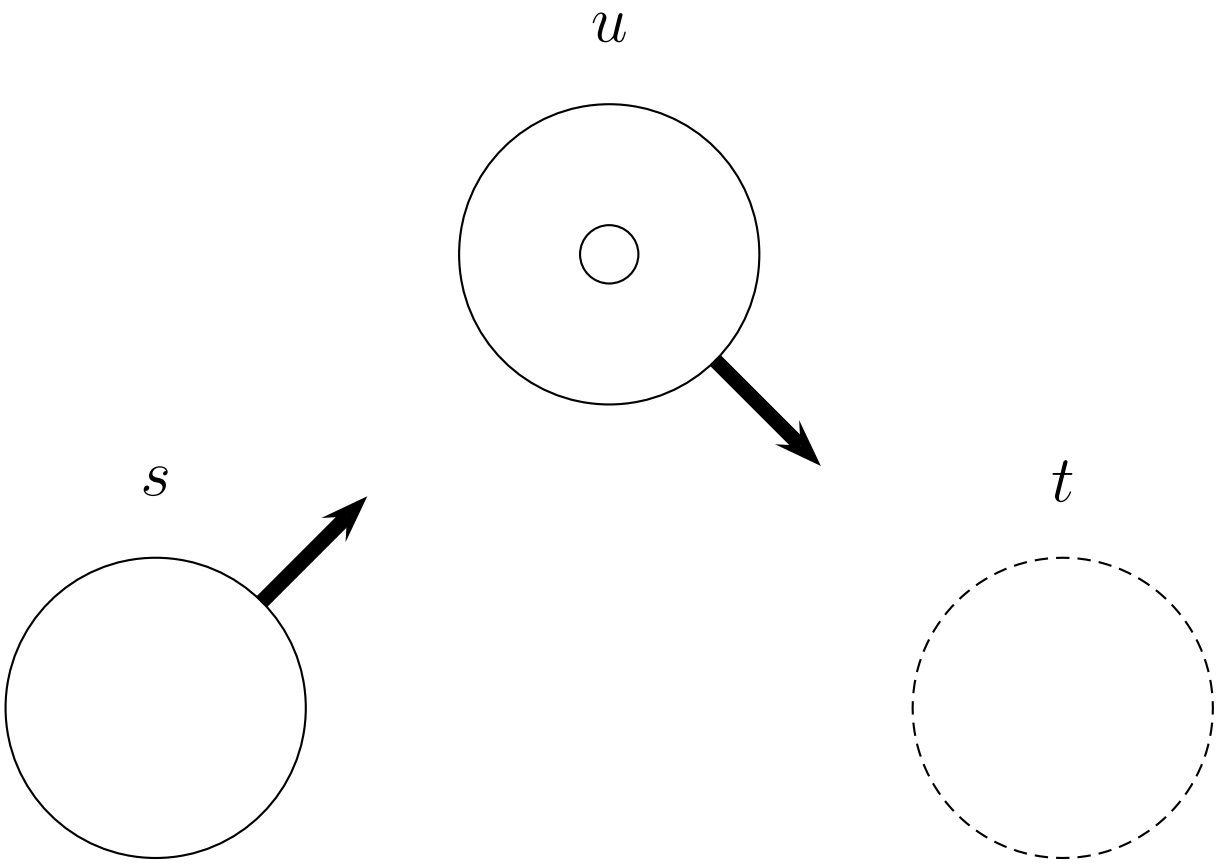
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)
Steps so far: 0

Rotor-routing on a triangle:



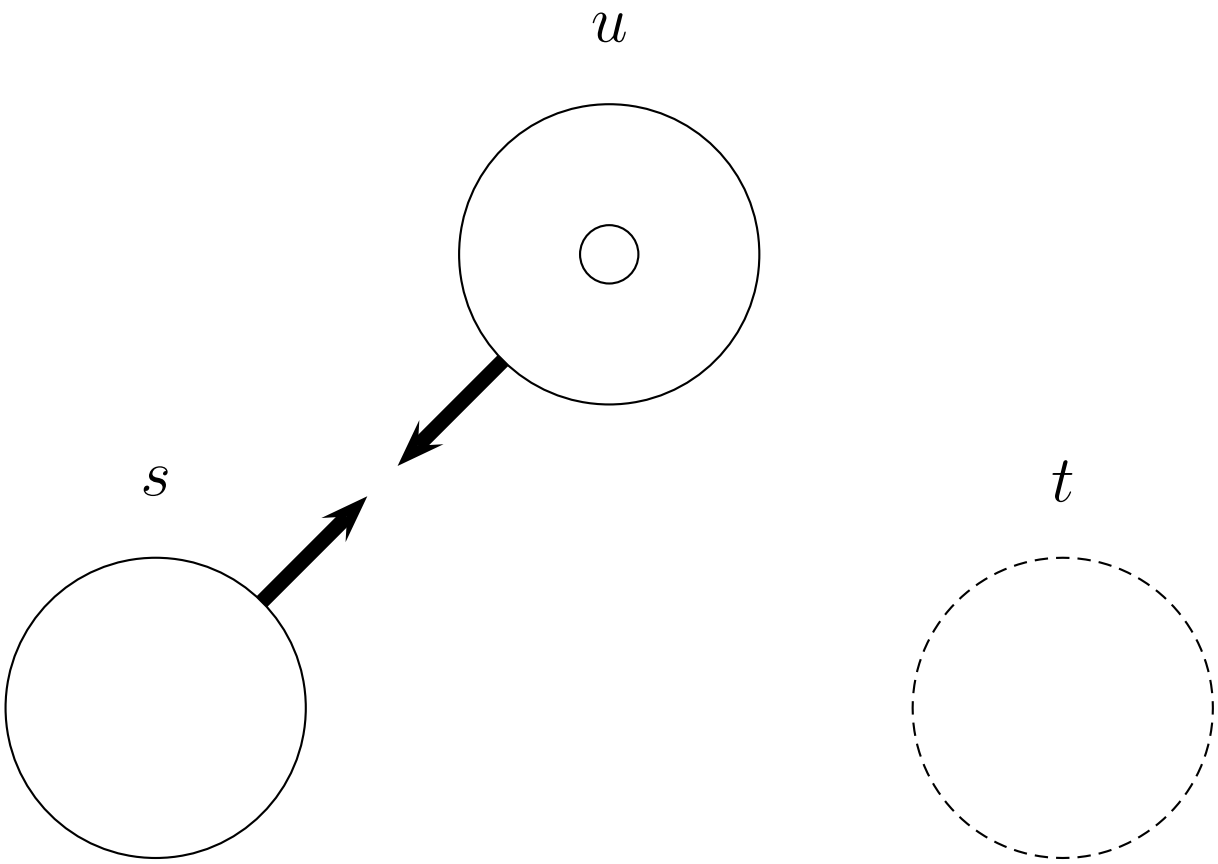
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)
Steps so far: 0

Rotor-routing on a triangle:



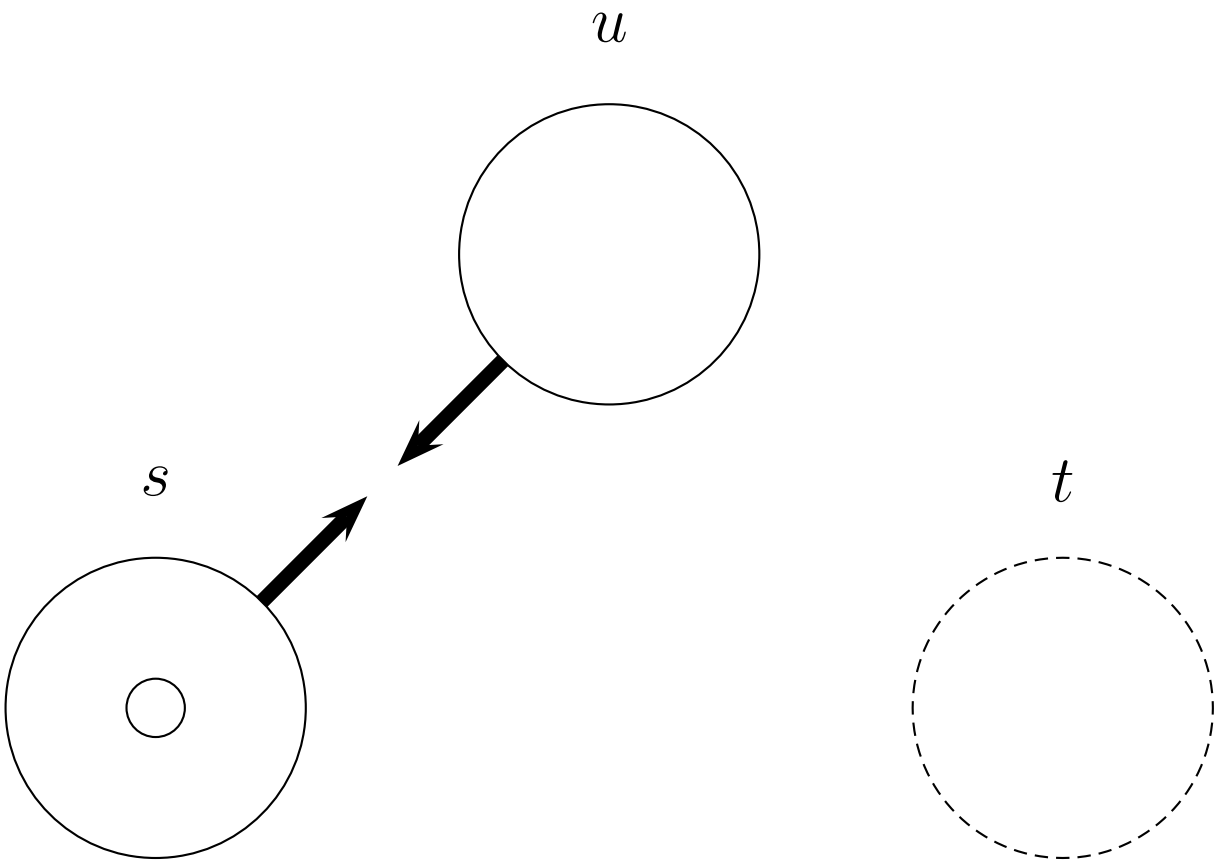
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)
Steps so far: 1

Rotor-routing on a triangle:



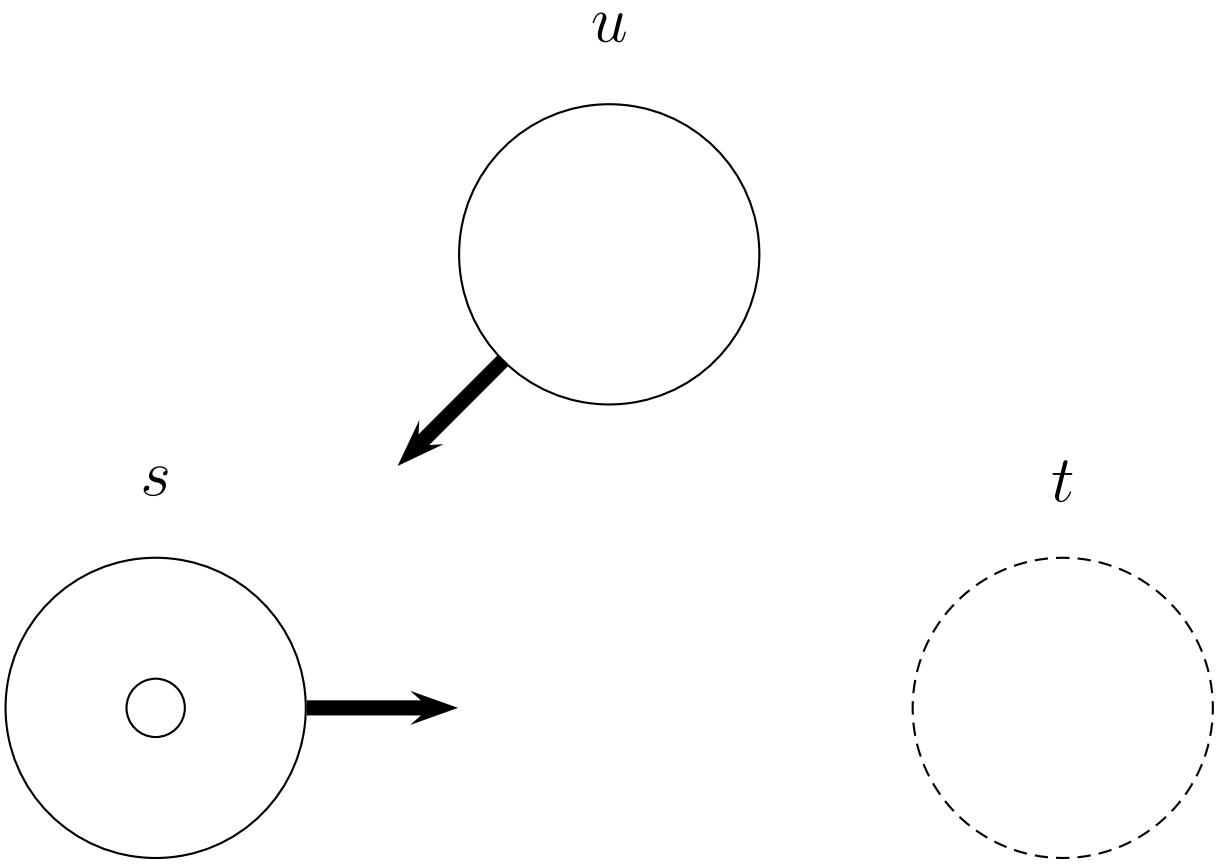
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)
Steps so far: 1

Rotor-routing on a triangle:



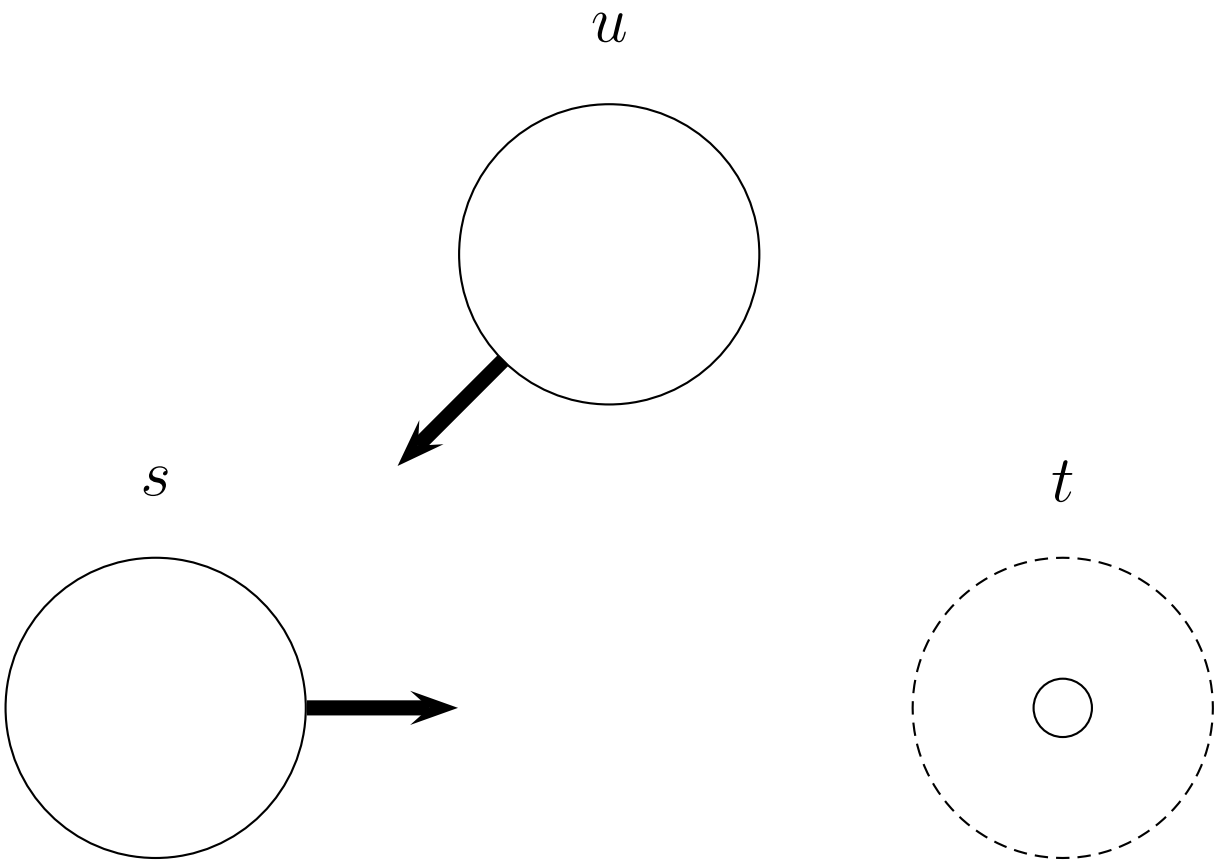
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)
Steps so far: 2

Rotor-routing on a triangle:



Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)
Steps so far: 2

Rotor-routing on a triangle:



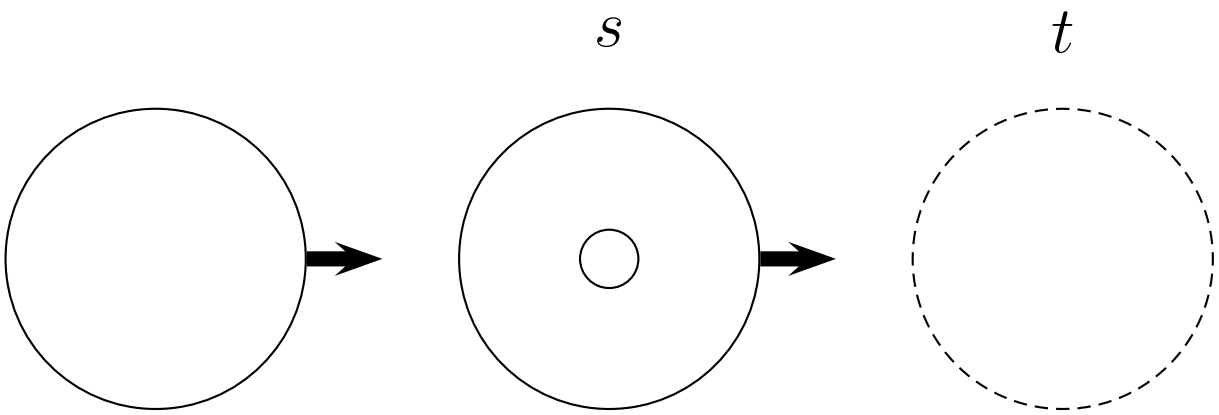
Steps so far: 1 (end of first stage)
Steps so far: 2 (end of second stage)
Steps so far: 1 (end of third stage)
Steps so far: 3 (end of fourth stage)

Theorem (Levine and Propp): The elementary excitation-relaxation operators for rotor-routing (defined in the obvious way) generate an abelian group, and this group is isomorphic to the critical group for chip-firing.

Corollary: When G is a tree, the number of stages in the period is just 1.

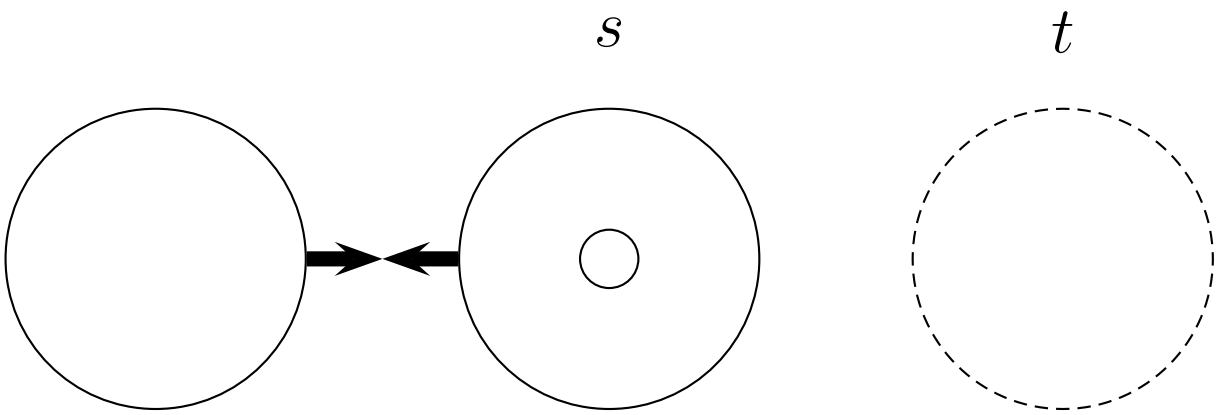
Example: The path of length 3, with rotors initially pointing towards the target.

Rotor-routing on a path of length 3:



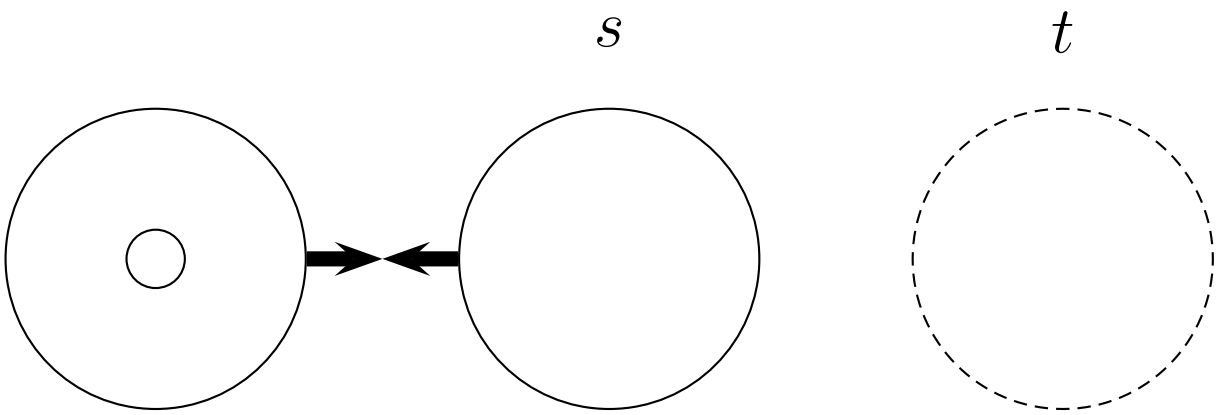
Steps so far: 0

Rotor-routing on a path of length 3:



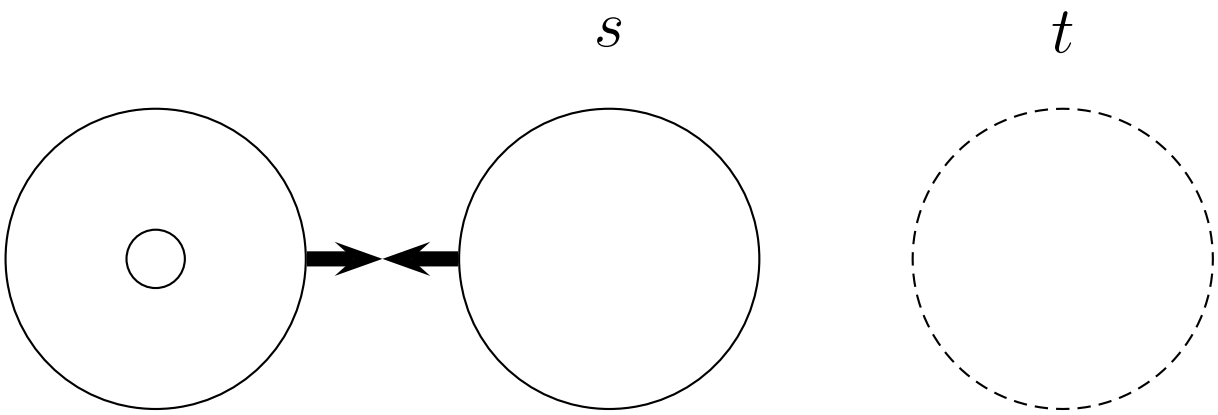
Steps so far: 0

Rotor-routing on a path of length 3:



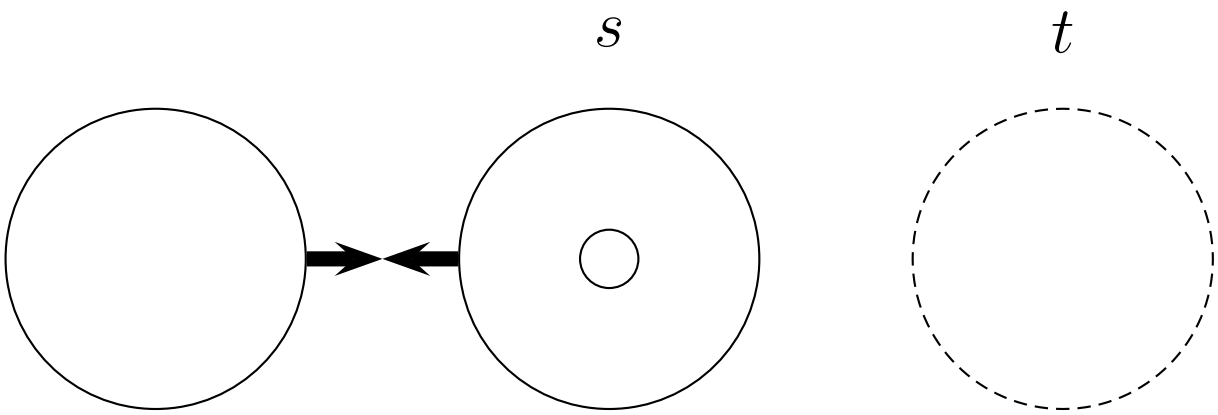
Steps so far: 1

Rotor-routing on a path of length 3:



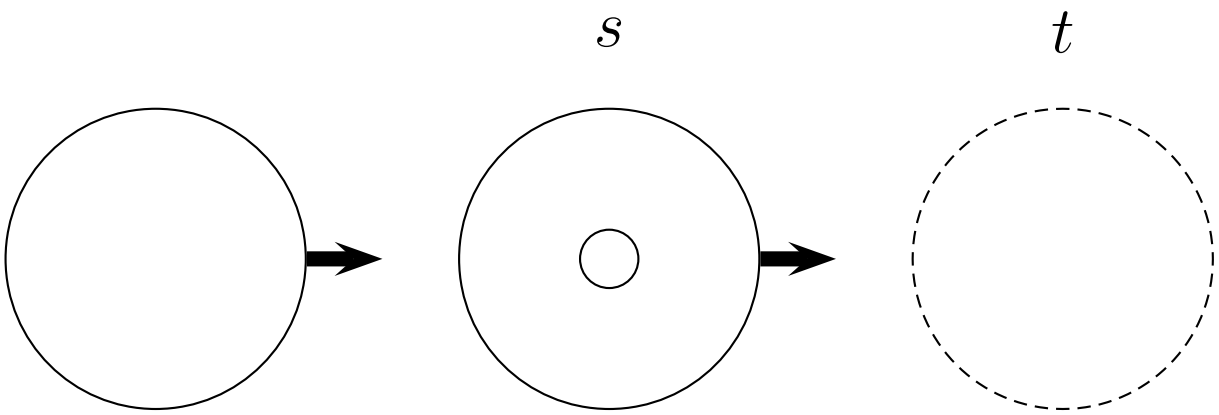
Steps so far: 1

Rotor-routing on a path of length 3:



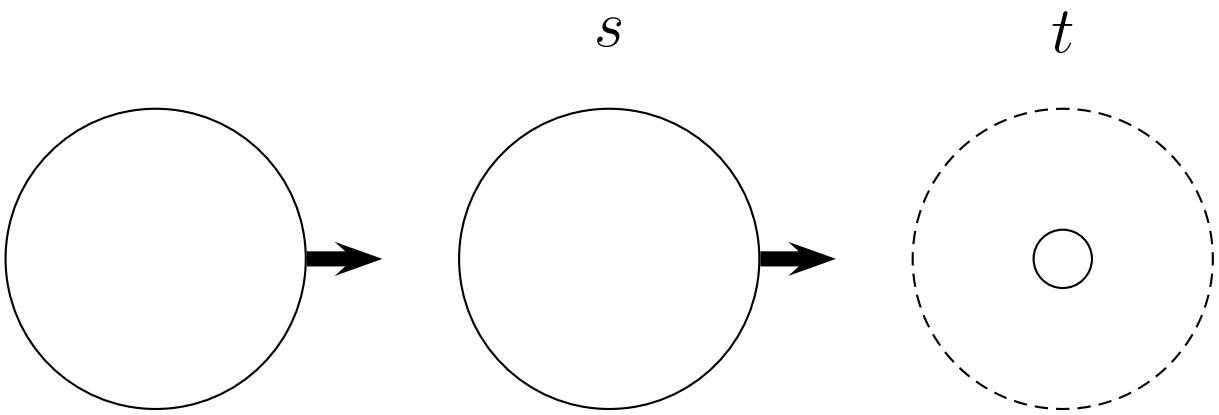
Steps so far: 2

Rotor-routing on a path of length 3:



Steps so far: 2

Rotor-routing on a path of length 3:



Steps so far: 3

Total number of steps: 3

Note that this is just the whirling tour of Dumitriu, Tetali and Winkler on the path of length 3.

This holds in full generality.

Moreover, the states of the whirling tour are precisely the unicycles.

We disregard the unicycle rooted at 0 because (for hitting time) we disregard the transition from the target back to the source (remember where we subtracted 1 in defining the length of a stage).

IV. Higher moments

Suppose a particle does random walk forever, starting from s. Let T_1 be the random time until the particle first hits target t_1 and T_2 be the random time until the particle first hits target t_2. $\text{Exp}(T_1)$ is just the expected hitting time for t_1 and $\text{Exp}(T_2)$ is just the expected hitting time for t_2. We can use chips (or if we prefer, rotors) not just to compute $\text{Exp}(T_1)$ and $\text{Exp}(T_2)$ but also $\text{Exp}(T_1 T_2)$.

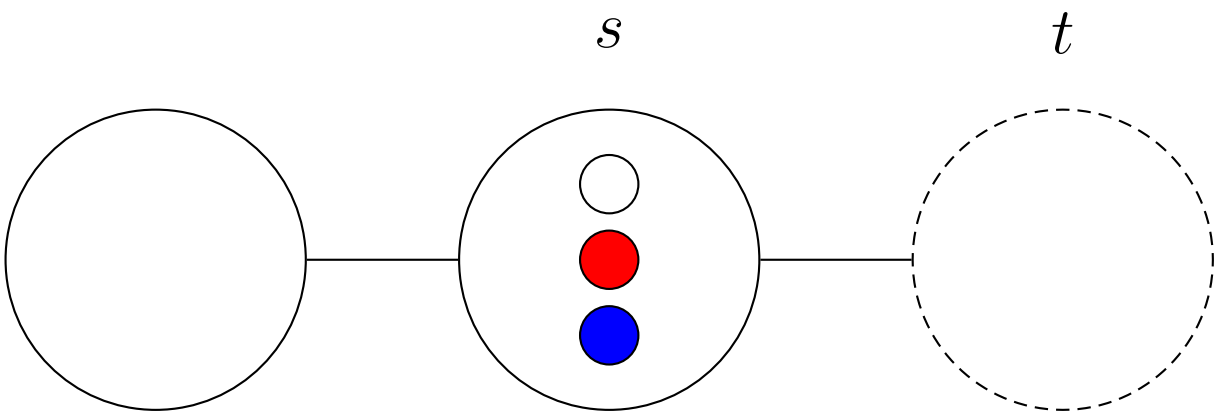
Chip-firing method (Propp): Add white chips to the system at s and remove them at t_1 and t_2. Every time you move a white chip, create a new blue chip and a new red chip at the vertex the white chip has moved to. Blue and red chips separately undergo chip-firing dynamics of the usual kind, except that blue chips only get absorbed at t_1 while red chips only get absorbed at t_2.

(For rotor-routing, have a white rotor, a blue rotor, and a red rotor at each vertex.)

Example: The path of length 3, with $t_1 = t_2$ on one end and s in the middle.

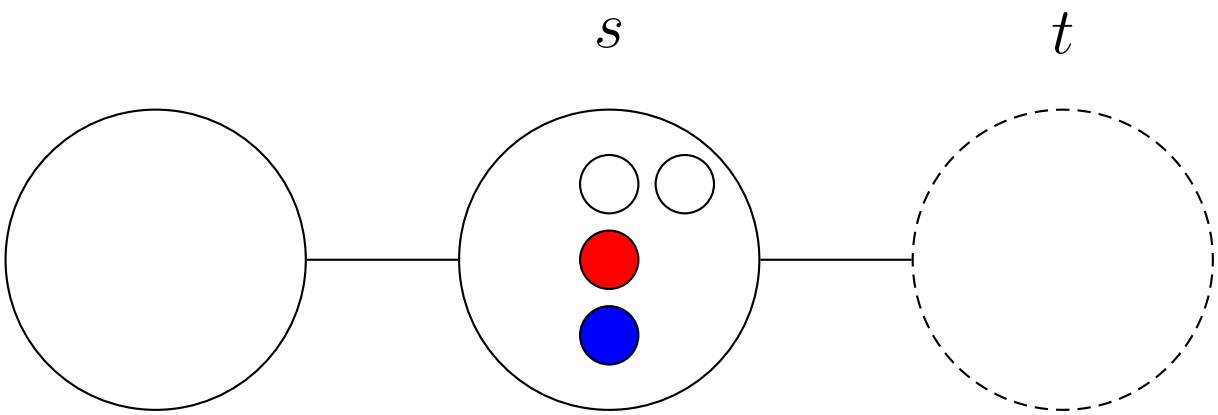
The expected hitting time from s to t is 3.
The expected squared hitting time from s to t is ...

Computing the expected square of the hitting time via multicolored chip-firing:



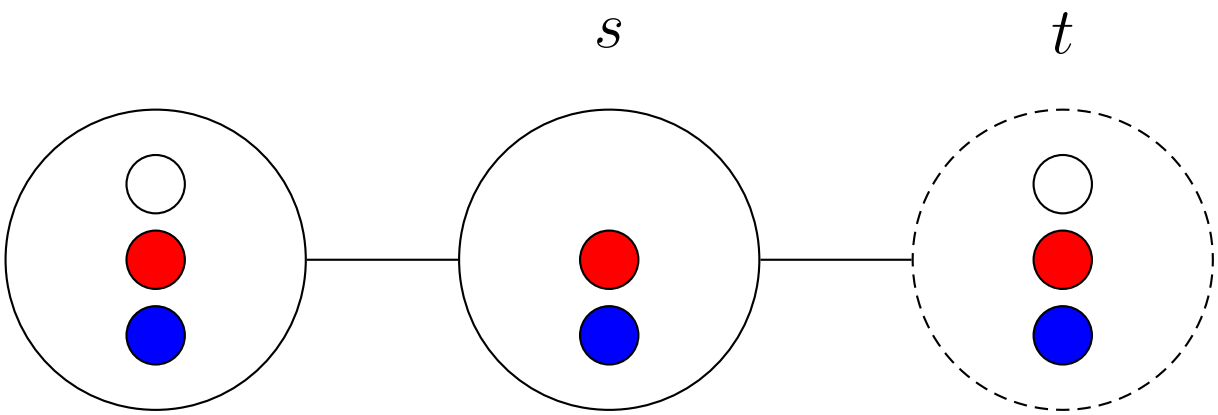
Moves so far: 0

Computing the expected square of the hitting time via multicolored chip-firing:



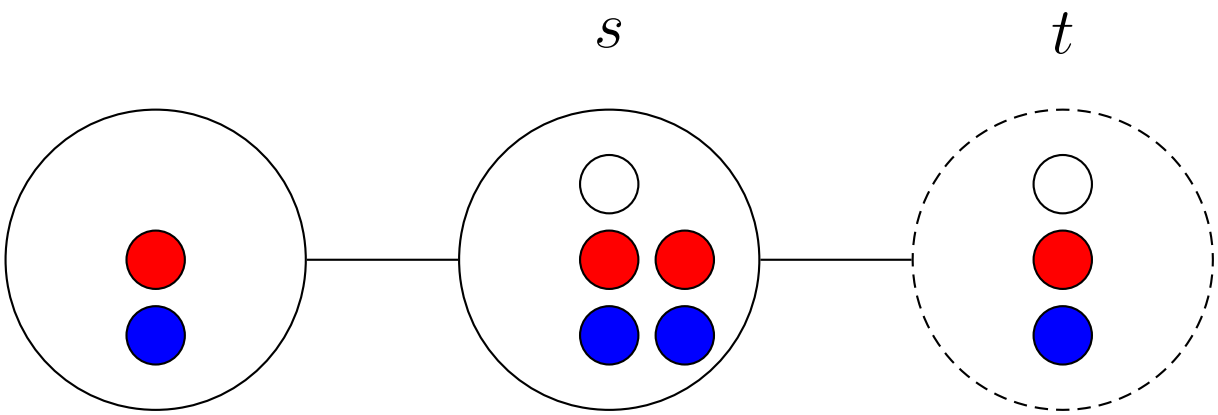
Moves so far: 0

Computing the expected square of the hitting time via multicolored chip-firing:



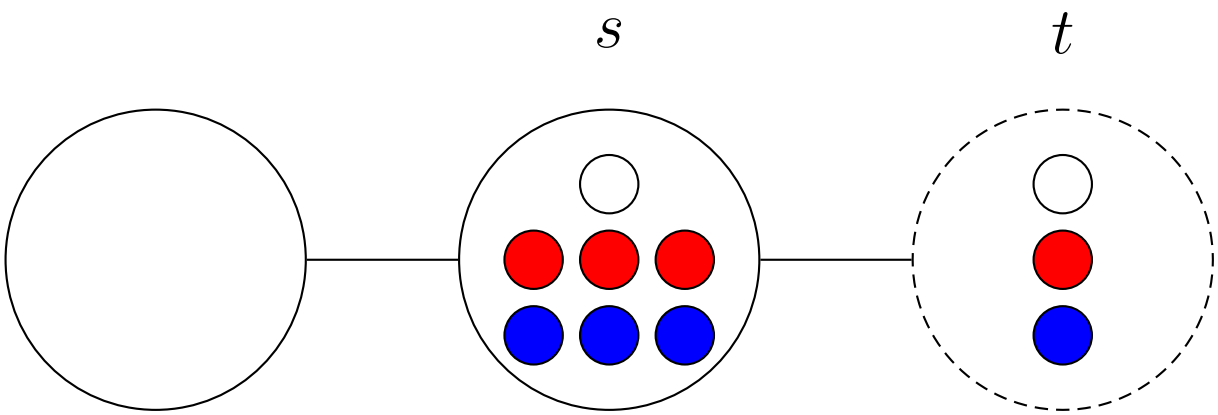
Moves so far: 2

Computing the expected square of the hitting time via multicolored chip-firing:



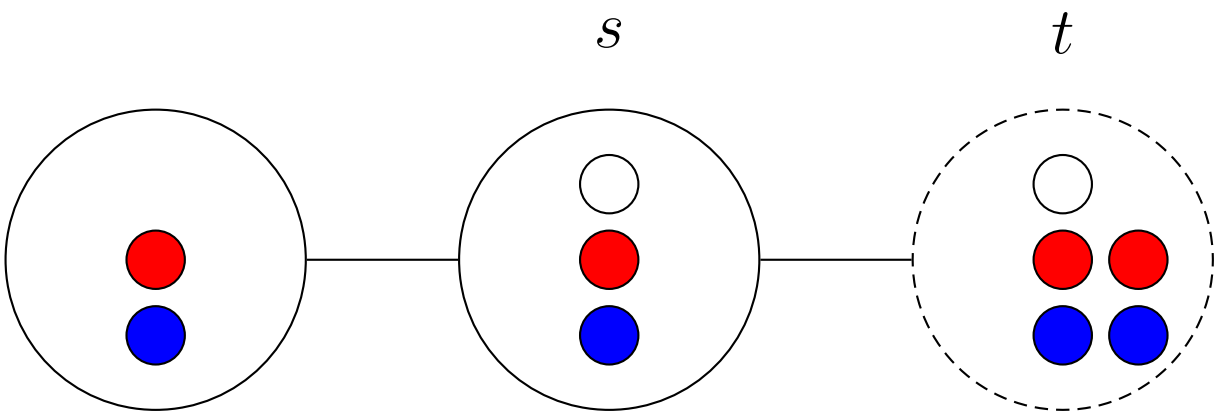
Moves so far: 3

Computing the expected square of the hitting time via multicolored chip-firing:



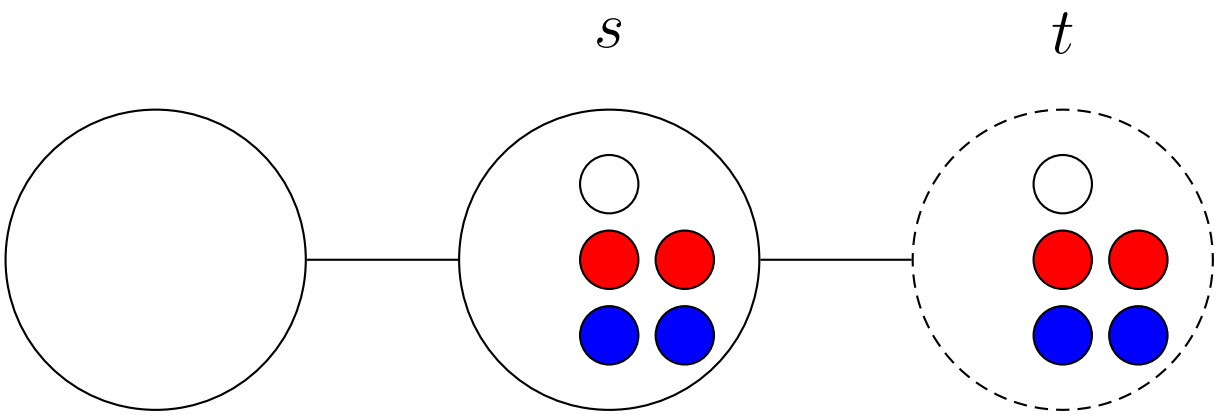
Moves so far: 5

Computing the expected square of the hitting time via multicolored chip-firing:



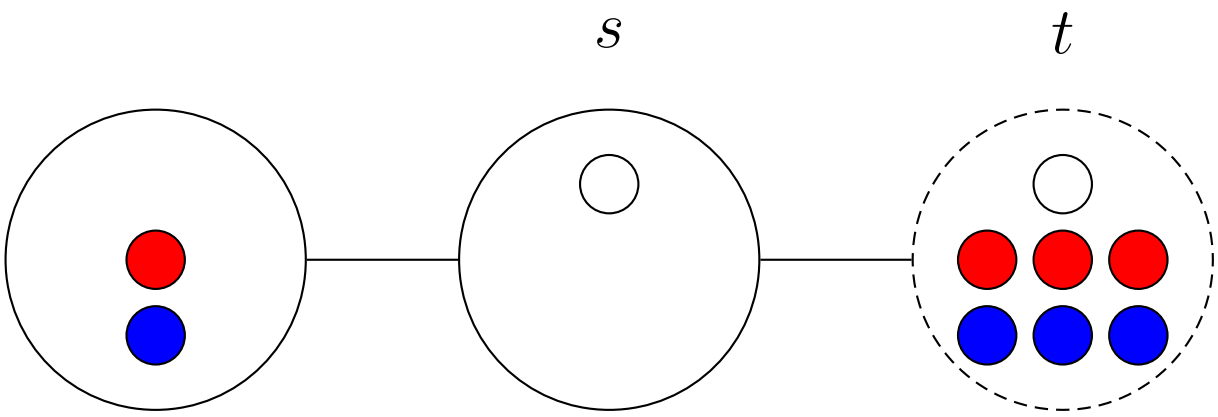
Moves so far: 9

Computing the expected square of the hitting time via multicolored chip-firing:



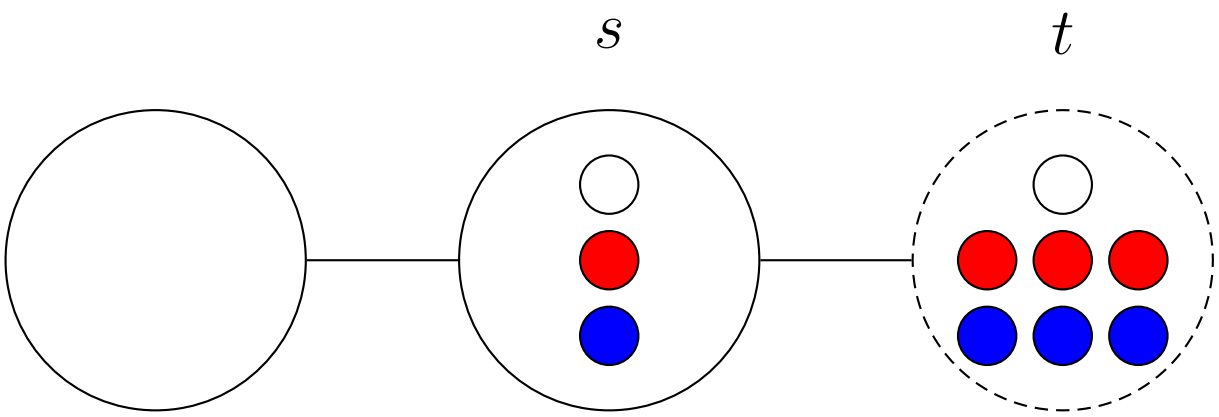
Moves so far: 11

Computing the expected square of the hitting time via multicolored chip-firing:



Moves so far: 15

Computing the expected square of the hitting time via multicolored chip-firing:



Moves so far: 17

The expected squared hitting time from s to t is 17.

It's not hard to use this set of ideas to show that for any tree, the expected square of the hitting time from s to t is an integer (easy direct proofs also exist).

Example: Let G be the two-vertex multigraph with vertices s, t, with a self-loop at s and an edge connecting s to t. We'll use ordinary chip-firing on G to compute the expected value of the geometric random variable with $p = 1/2$, and use multi-colored chip-firing on G to compute the expected value of the square of this geometric random variable.

Example: Let G be the two-vertex multigraph with vertices s, t, with a self-loop at s and an edge connecting s to t. We'll use ordinary chip-firing on G to compute the expected value of the geometric random variable with $p = 1/2$, and use multi-colored chip-firing on G to compute the expected value of the square of this geometric random variable.

In honor of Pete's love of (productive!) provocation, we leave the solution to the amusement of the reader.

Exercise: Let G be a finite connected graph with vertices s, t. Design a chip-firing scheme to compute the expected square of the number of visits to t in between visits to s under random walk.

Exercise: Extend the approach to the study of higher moments of such random variables.

A mathematician named Pete

Limerick contest:

A mathematician named Pete

...

Present submissions to me by lunch tomorrow; the winner will be announced by the end of the conference.

slides available at

`jamespropp.org/tour.pdf`