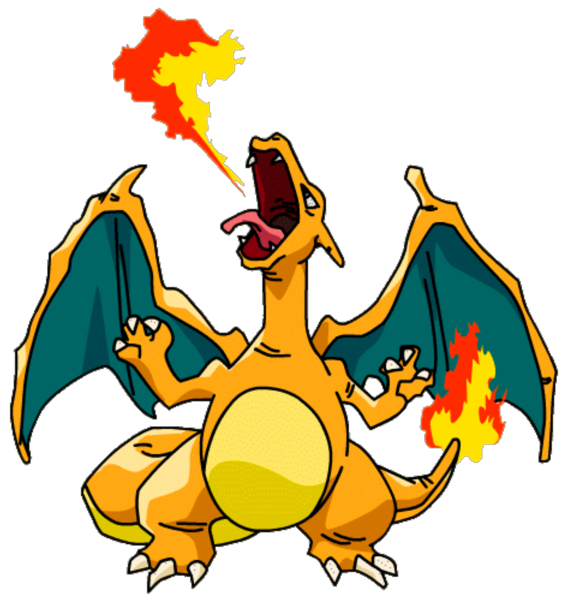
Probability and Recurrences

Admin

- **Assignment 8** is out due next Wed; based on this week of lectures
  - **Lots of new definitions**---helpful to review notes and examples from class before/while attempting HW
- Assignment 9 will be the last one you turn in 🎉
- Assignment 10 will consists of practice problems
  - Help prepare you for finals; review all topics
- Find out about CS courses next year: **Preregistration Info Session at 3.15pm**
- **Health day** next week 🎉🎉🎉

Recap: Last Lecture

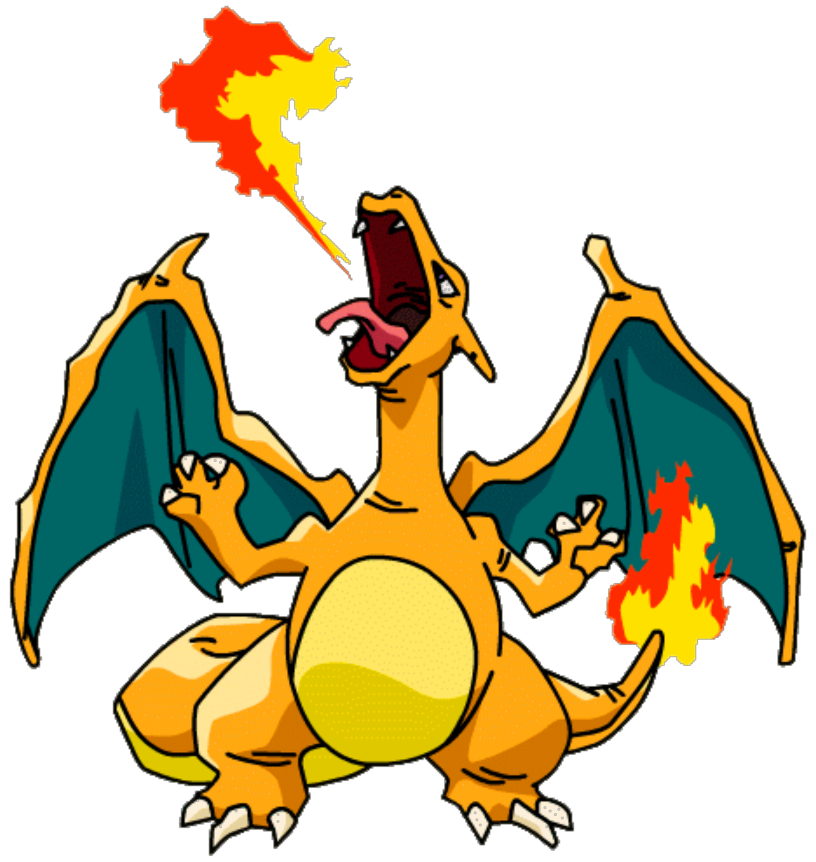
- Expectation, law of total expectation and linearity of expectation
- Common tricks to compute expectation:
  - Break down into indicator RVs, or random variables we already know how to compute expectation of and use LoE
  - Condition random variable on an event and form a recurrence
- n Bernoulli trials when probability of success is p
  - Expectation of number of successes?
  - Expected number of trials until first success?
- nth Harmonic number?
  - $1 + 1/2 + 1/3 + \ldots + 1/n = \Theta(\log n)$



Coupon/Pokemon Collector Problem

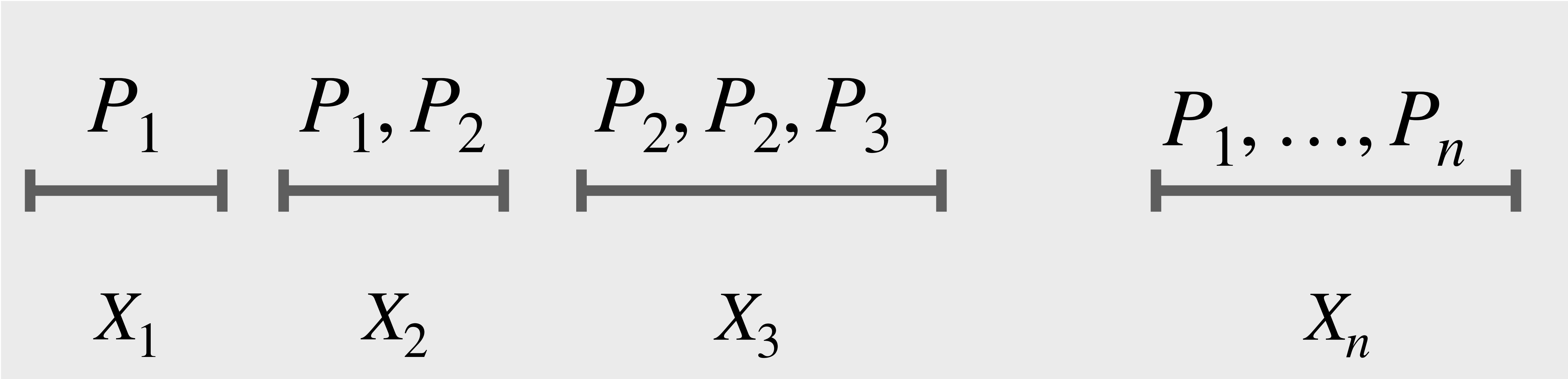
Gotta' Catch 'Em All

- Suppose there are n different types of Pokemon cards
- In each trial we purchase a pack that contains **one** Pokemon card
- We repeat until we have at least one of each type of card, how many packs does it take in expectation to collect all?
- Let X be the r.v. equal to the number of packs bought until you first have a card of each type.
- **Goal**: compute $E[X]$
- We break X into smaller random variables
- **Idea**: we make progress every time we get a card we don't already have

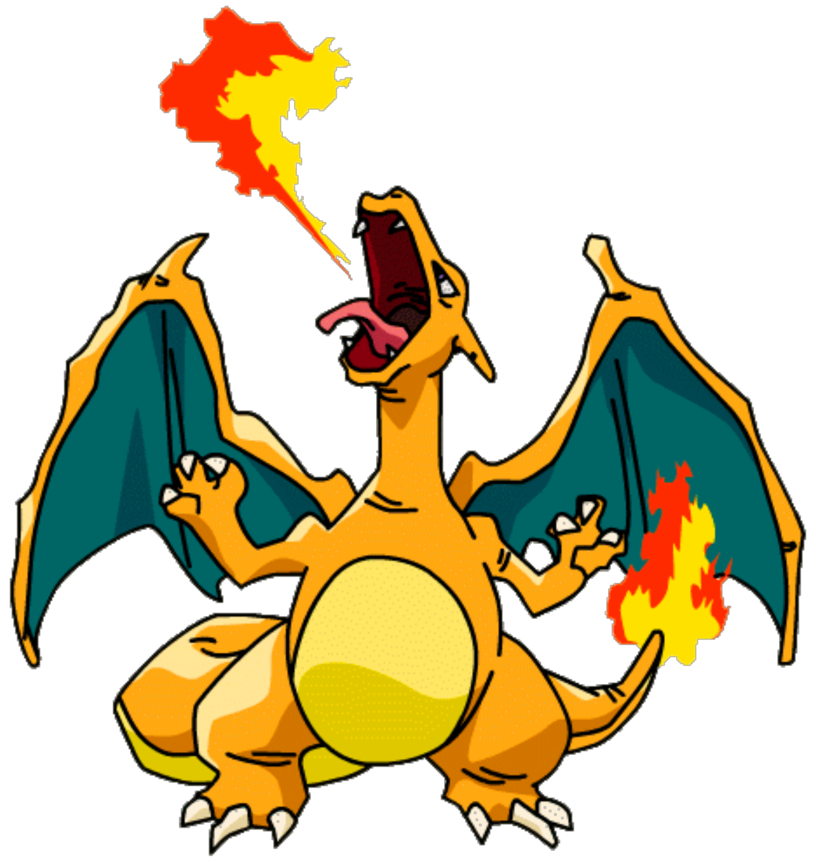


Pokemon Collector Problem

- Let X_i denote the "length of the ith phase", that is, the number of packs bought during the ith phase (ith phase ends as soon as we see the ith distinct card)
- Thus, $X = \sum_{1=1}^{n} X_i$

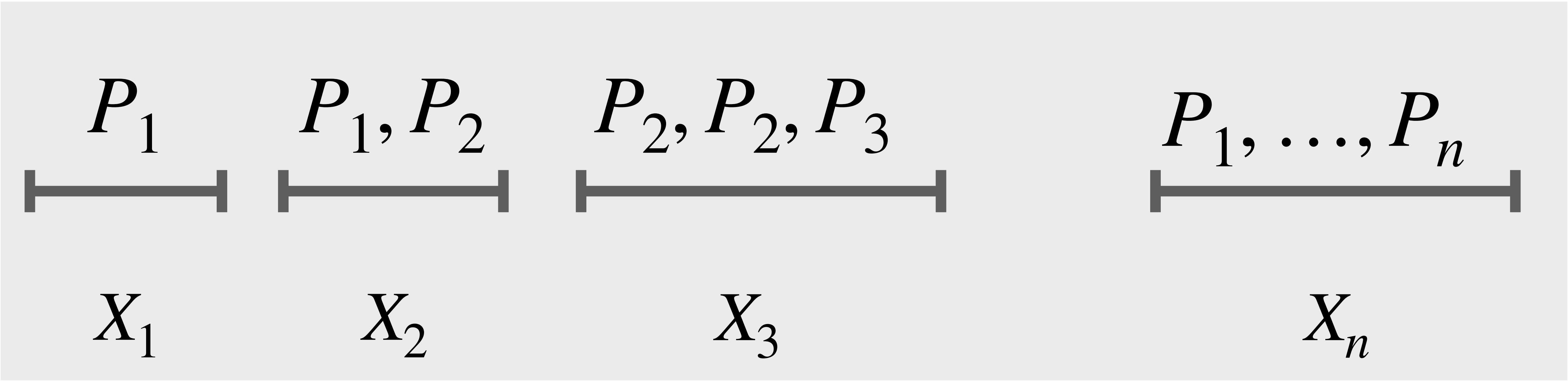


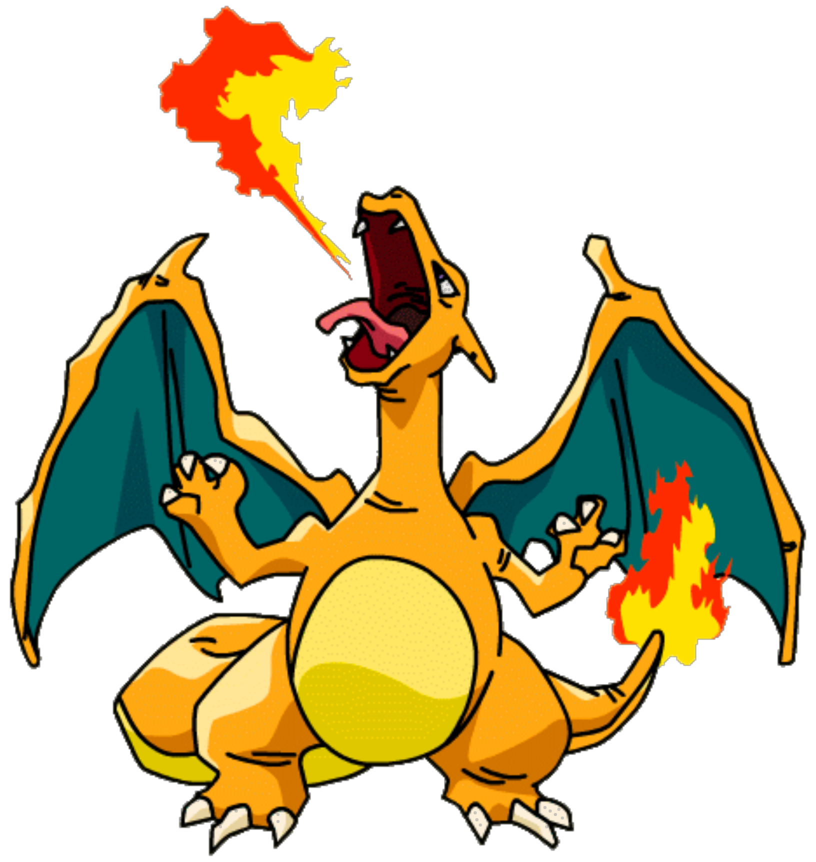
- Each phase can be though of as flipping a biased coin until we see a head, where seeing a head = getting a new card



Pokemon Collector Problem

- $E[X_i]$ is the expected number of coin flips until success (expectation of a geometric r.v.)

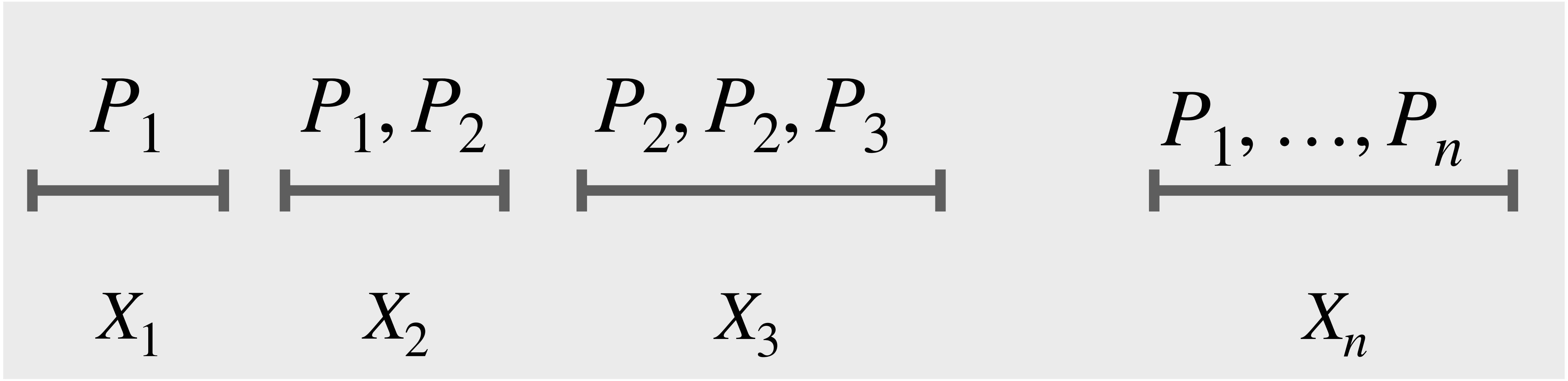


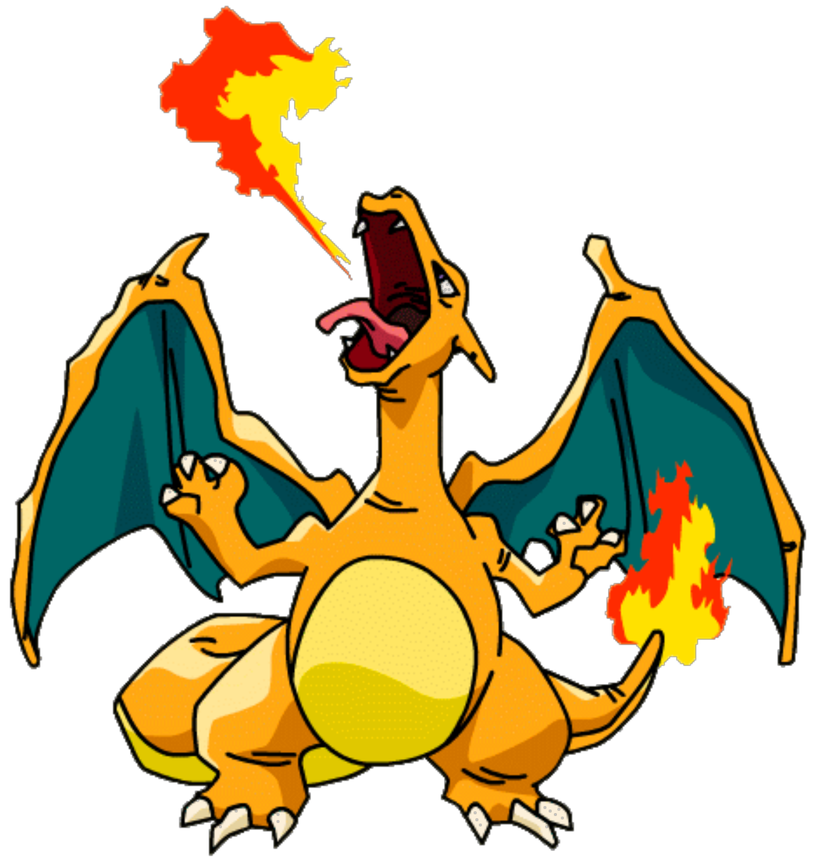


- We know, $E[X_i] = 1/p_i$ where p_i is the probability of success/ probability of seeing a heads during a coin flip in the ith phase
- Before the ith phase starts, we don't have $n - i + 1$ Pokemon
- Each of the n Pokemon are equally likely to be in a pack

Pokemon Collector Problem

- $E[X_i]$ is the expected number of coin flips until success (expectation of a geometric r.v.)



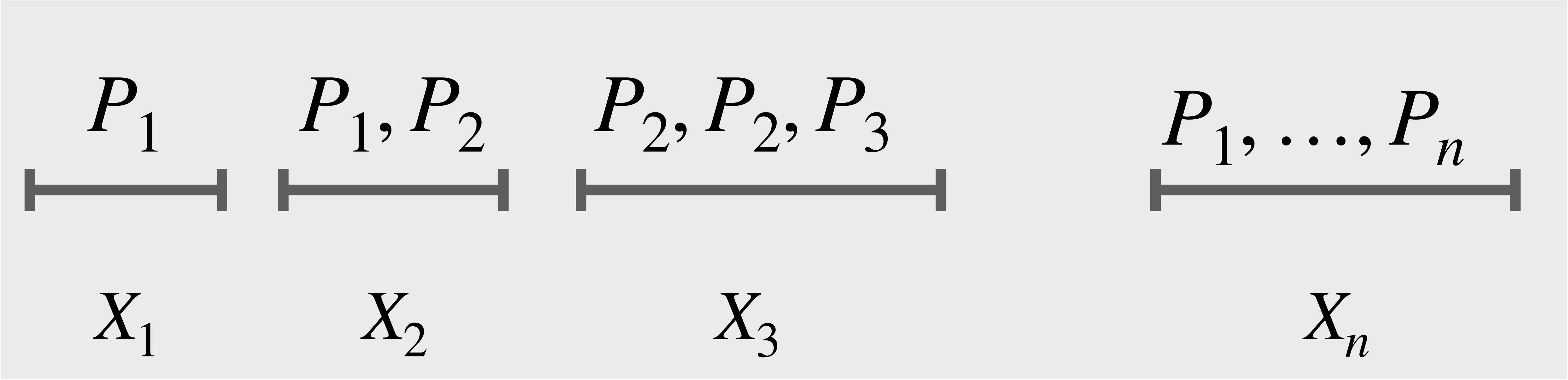


- We know, $E[X_i] = 1/p_i$ where p_i is the probability of success/ probability of seeing a heads during a coin flip in the ith phase

- $p_i = \dfrac{n - i + 1}{n}$

Pokemon Collector Problem

- We know, $E[X_i] = 1/p_i$ where p_i is the probability of success/ probability of seeing a heads during a coin flip in the ith phase

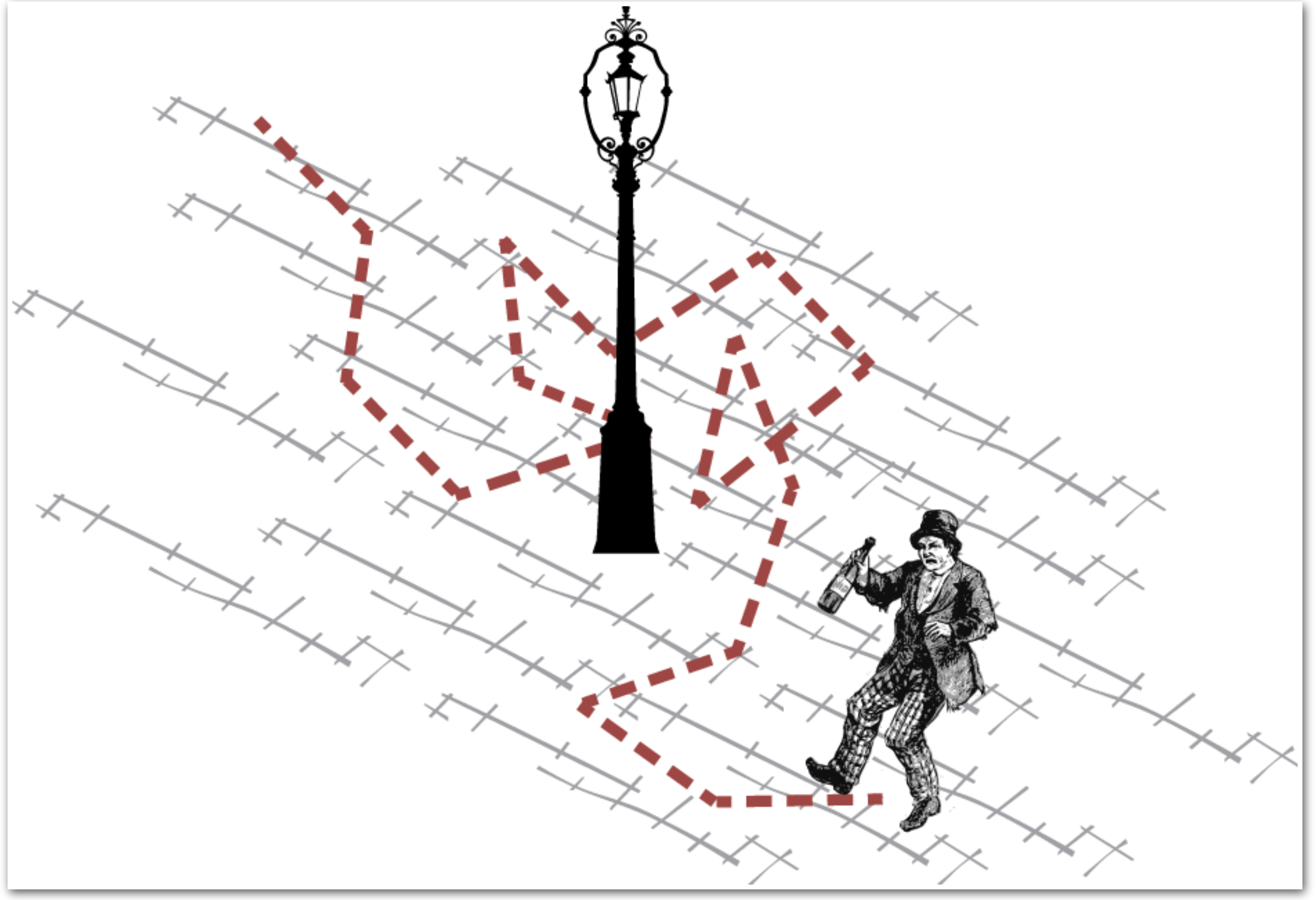


- $E[X_i] =$ Expected[number of flips until first heads] $= 1/p_i = \dfrac{n - i + 1}{n}$
- $$E[X] = E[\sum_{i=1}^{n} X_i] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{n}{n-i+1} = \sum_{i=1}^{n} \frac{n}{i} = nH_n = \Theta(n \log n)$$

Random Walks and Recurrences

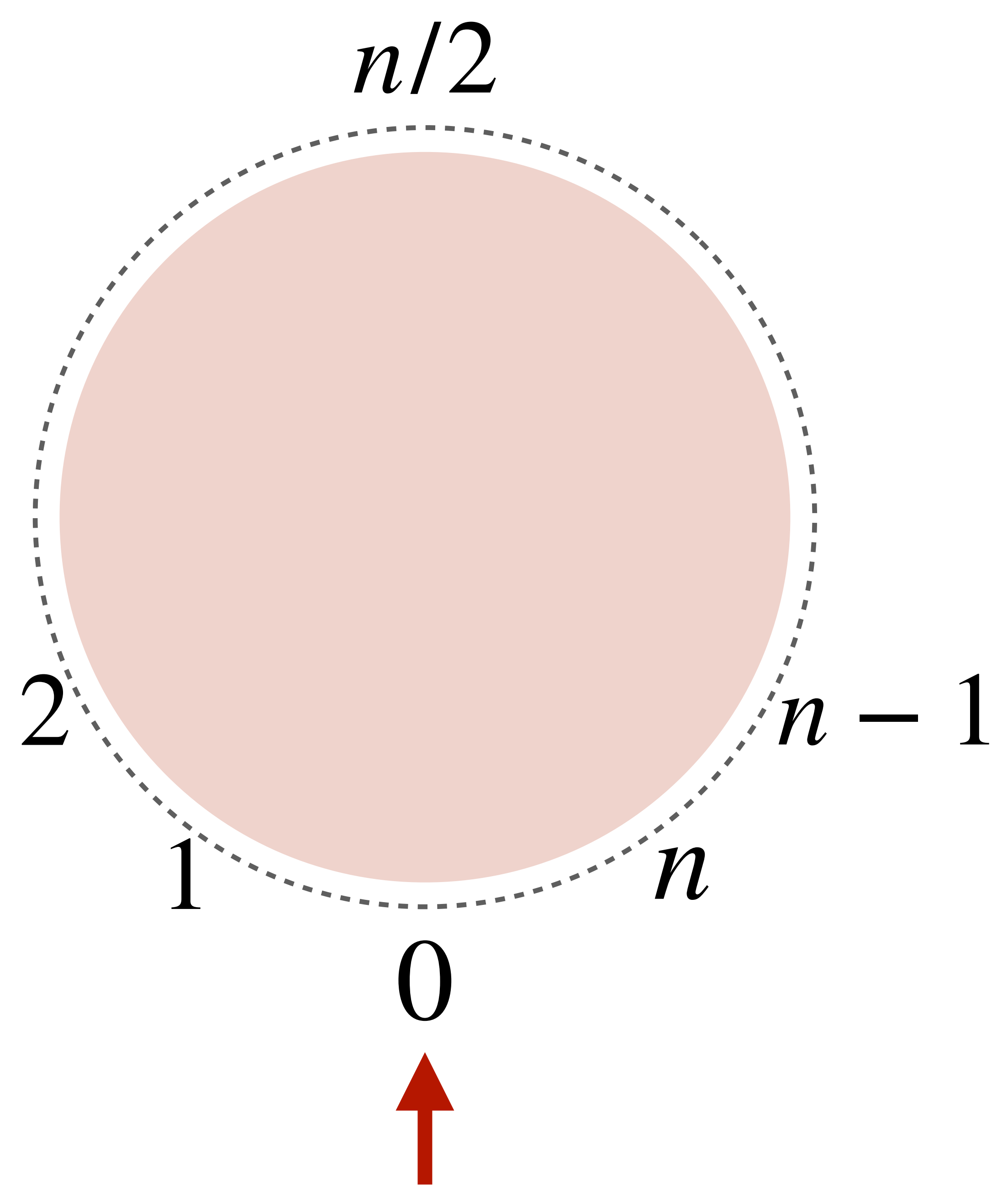
Random Walks

- A drunkard stumbles out of a bar. Each second, he either staggers 1 step to the left or staggers 1 step to the right, with equal probability. His home lies x steps to his left, and a canal lies y steps to his right.
- **Questions.** What is the probability that the drunkard arrives safely at home instead of falling into the canal? What is the expected duration of his journey, however it ends?
- The drunkard's meandering path is called a **random walk**
- Random walks are important as they model various phenomenon:
  - In Physics, random walks model gas diffusion
  - Google search engine uses random walks through the graph of web links to determine the relative importance of website
  - In finance theory, random walks can serve as a model for the fluctuation of market prices



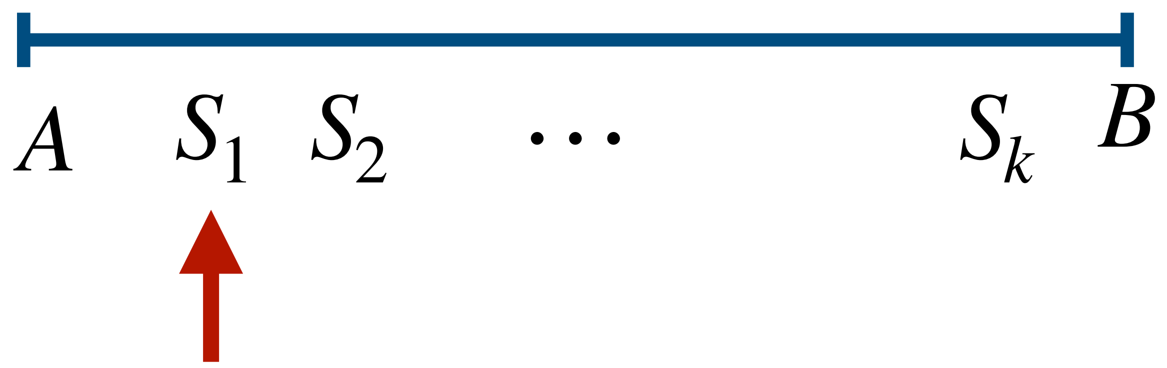
Pass the Candy

- We have n students labelled $1,\ldots,n$ and a professor labelled 0 sitting around in a circle
- Initially the professor has a candy bowl. She withdraws a piece of candy and then passes the bowl either to the left or right, with equal probability
- Each person who receives the bowl takes a piece of candy if they do not already have one; then passes it on randomly
- The ***last*** person to receive the candy wins the game
- Which player is most likely to win? Guess?
- Seems like 1 and n are almost always going to be eliminated right away. Seems like $n/2$ is most likely to win —but by how much?



Simpler Problem

- Suppose the players $A, S_1, \ldots, S_k, B$ are arranged in a line instead and S_1 initially has a the candy
- As before, whenever a player gets the bowl they take a candy and pass it left or right with equal probability
- What is the probability that A gets the candy before B?
- Let P_k be the probability that A gets the candy before B.
- **Base case.** Suppose $k = 1$, then $P_1 = 1/2$
- Suppose $k > 1$. In the first step there are two possibilities: the bowl either moves left to A or right to S_2

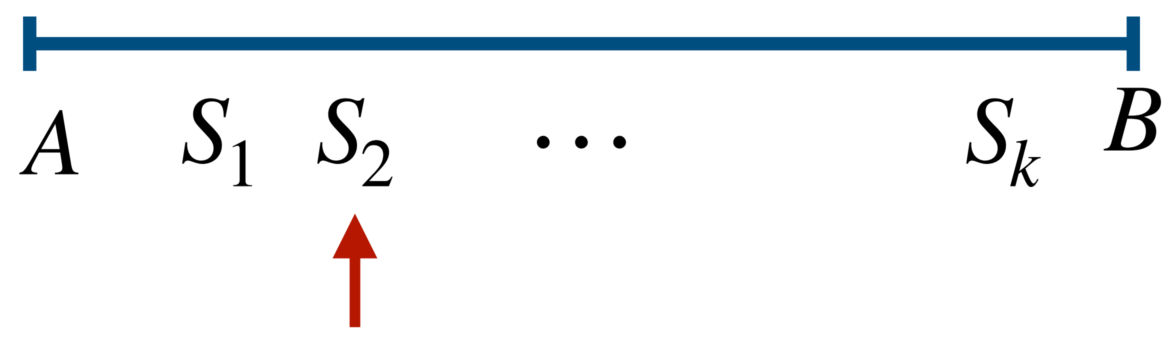


$$
\begin{aligned}
P_k &= \Pr(\text{first step is left}) \\
&\qquad \cdot \Pr(A \text{ gets candy before } B \mid \text{first step is left}) \\
&\quad + \Pr(\text{first step is right}) \\
&\qquad \cdot \Pr(A \text{ gets candy before } B \mid \text{first step is right}) \\
&= \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \Pr(A \text{ gets candy before } B \mid \text{first step is right})
\end{aligned}
$$

Simpler Problem

- $$\begin{aligned} P_k &= \Pr(\text{first step is left}) \\ &\qquad \cdot \Pr(A \text{ gets candy before } B \mid \text{first step is left}) \\ &\quad + \Pr(\text{first step is right}) \\ &\qquad \cdot \Pr(A \text{ gets candy before } B \mid \text{first step is right}) \\ &= \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \boxed{\Pr(A \text{ gets candy before } B \mid \text{first step is right})} \end{aligned}$$

- Recurrence. $P_k = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot P_{k-1} \cdot P_k$ and $P_1 = \frac{1}{2}$ (Base case)
- Solve it using guess and check, and prove by induction
- $P_2 = \dfrac{1}{2 - P_1} = \dfrac{2}{3}$, $P_3 = \dfrac{1}{1 - P_2} = \dfrac{3}{4}$
- $P_k = \dfrac{k}{k+1}$ (Verify this is correct by induction)



New starting configuration

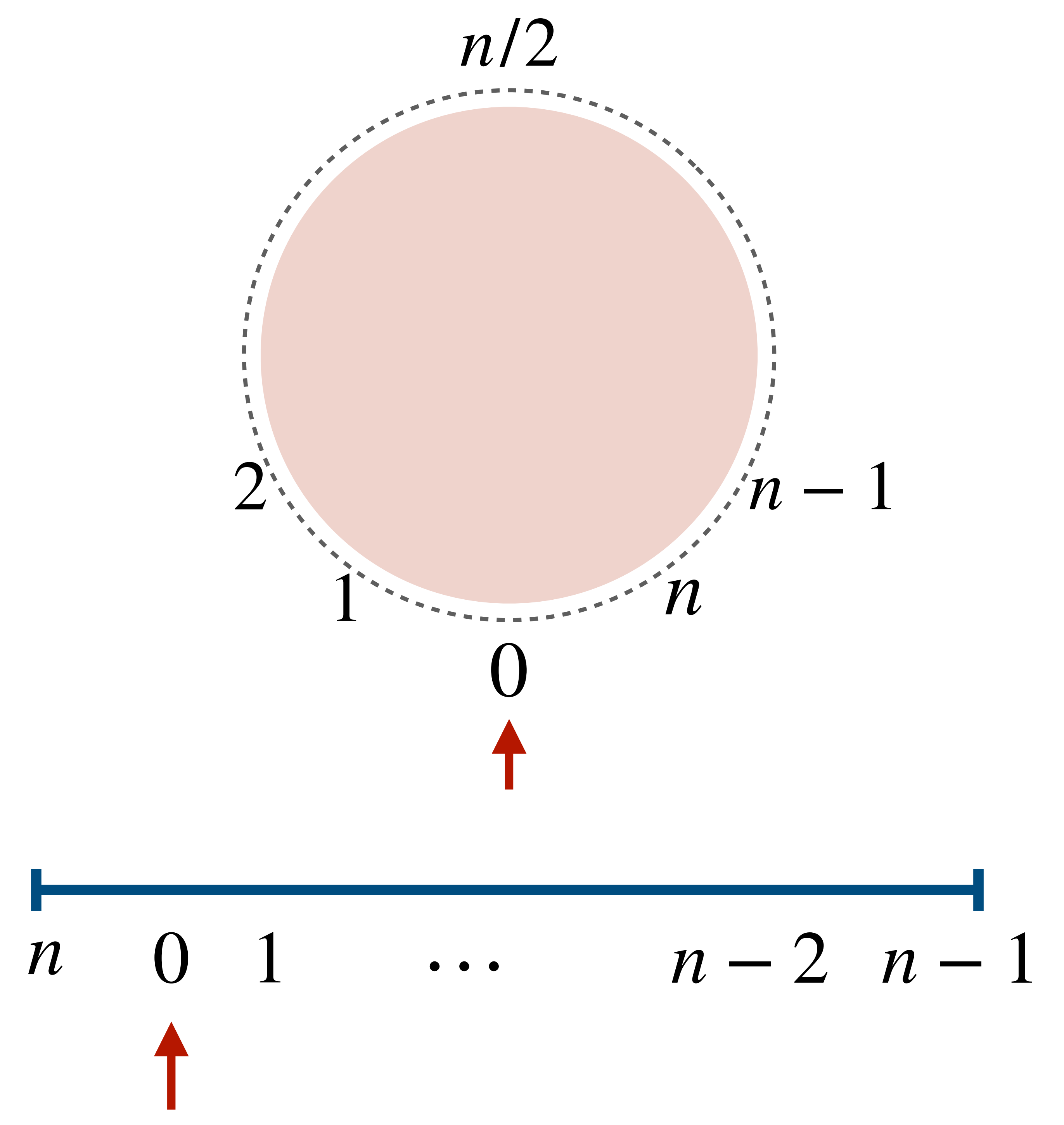
Pr(S_1 gets candy before B) $= P_{k-1}$

If S_1 gets candy, we are back in the initial configuration and A gets the candy before B with probability P_k.

Back to the Candy Game

- Consider player n on the right side of the professor. Only way n can win is the candy travels clockwise all the way around to player $n-1$ before n ever touches it
- Thus, if we cut the circle and arrange on a line as shown, n wins if and only if $n-1$ gets the bowl before n
- This fits our previous simpler problem model where we want to know the probability that B gets candy before A and $k = n-1$
- Pr($n-1$ gets candy before n) $= 1-$Pr(n gets candy before $n-1$)

  $$= 1 - \frac{n-1}{(n-1)+1} = \frac{1}{n}$$
- Student n wins with probability $1/n$!
- We can extend this argument to show each student wins with probability $1/n$ --- **we would never have guessed this**!



n wins only if $n-1$ gets candy before n

Randomized Algorithm I

Min Cut

Randomized Min Cut

- **Global min-cut problem.** Given an undirected, unweighted graph $G = (V, E)$, find a cut (A, B) of minimum cardinality (that is, min # of edges crossing it).
- **Applications.** Network reliability, network design, circuit design, etc.
- **Poly-time network-flow solution** (by reduction to min s-t cut).
    - Replace every undirected edge (u, v) with $u \to v$ and $v \to u$, each of capacity 1
    - Fix any $s \in V$ and compute min s-t cut for every other node $t \in V - \{s\}$
    - $(n - 1)$ executions of min s-t cut
- Gives impression that finding global min cut is harder than finding a min s-t cut, which is not true
- Deceptively simple and efficient randomized algorithm [Karger 1992]

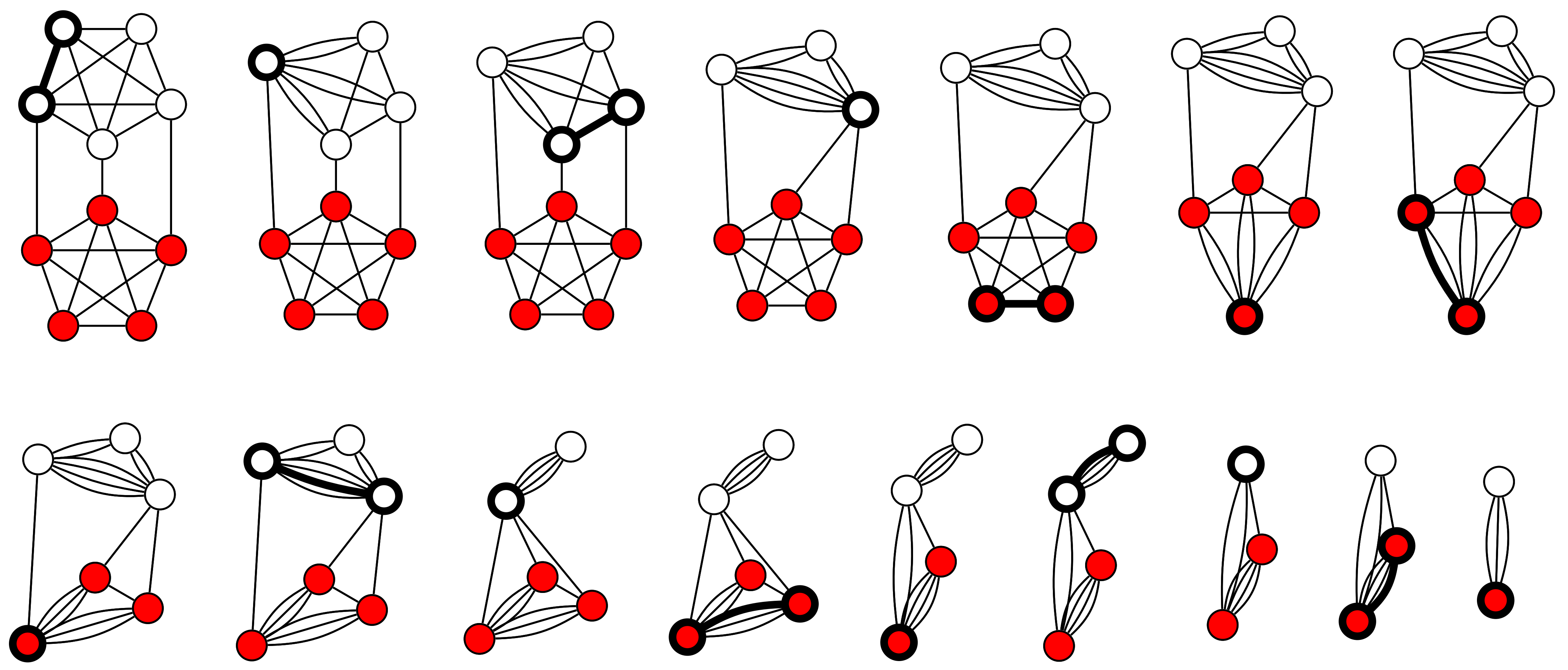
Karger's Min Cut

- **Edge contraction:** Contract edge e in G, denoted $G \leftarrow G/e$
  - Replace u and v by single new super-node w
  - Preserve edges, updating endpoints of u and v to w
  - Keep parallel edges, but delete self-loops
- An edge can be contracted in $O(n)$ time, assuming the graph is represented as an adjacency list

Karger's Min Cut

- Algorithm tries to *guess* the min cut by randomly contracting edges
- Running time:
  - $O(n)$ edge contraction, $O(n)$ iterations: $O(n^2)$
- **Correctness**: How often, if ever, does it return the min cut?

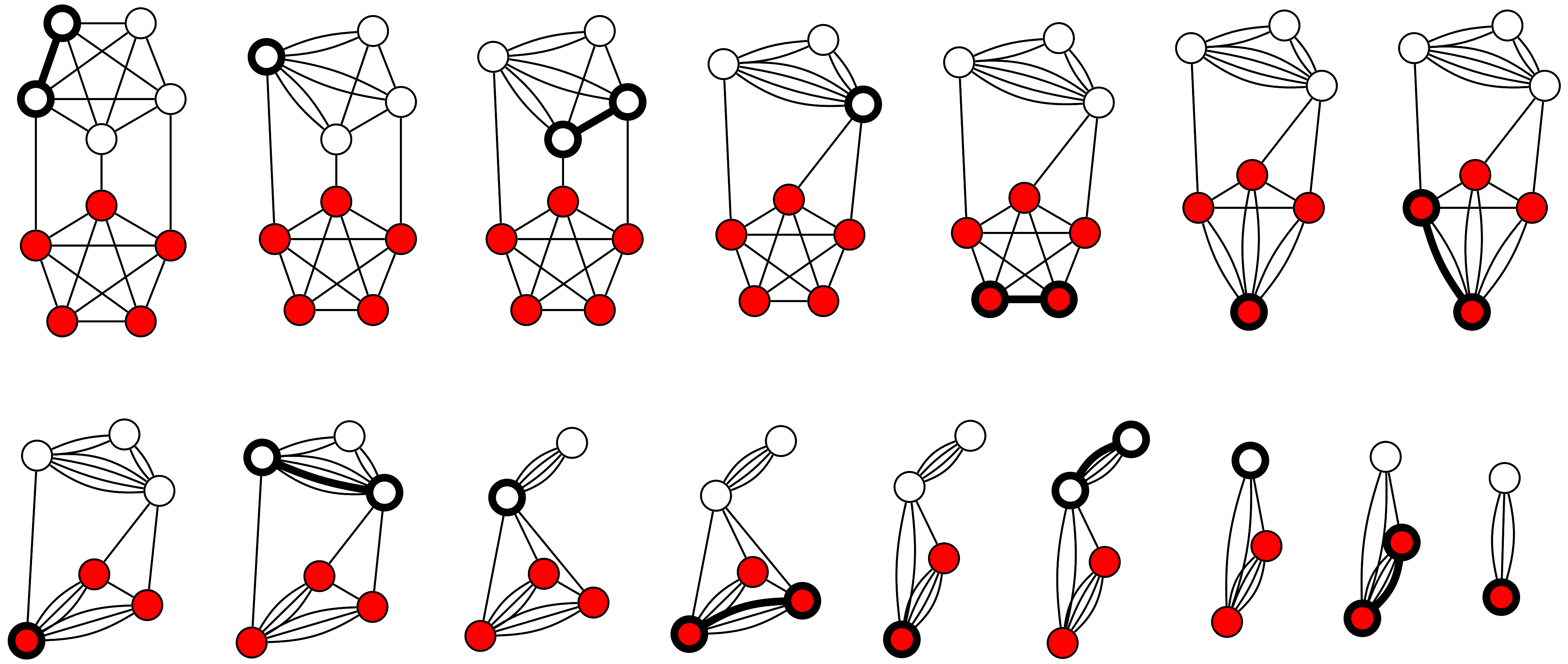
```
GuessMinCut(G):
    for i ← n downto 2
        pick a random edge e in G
        G ← G/e
    return the only cut in G
```



Reference: Thore Husfeldt

Preserving Cuts

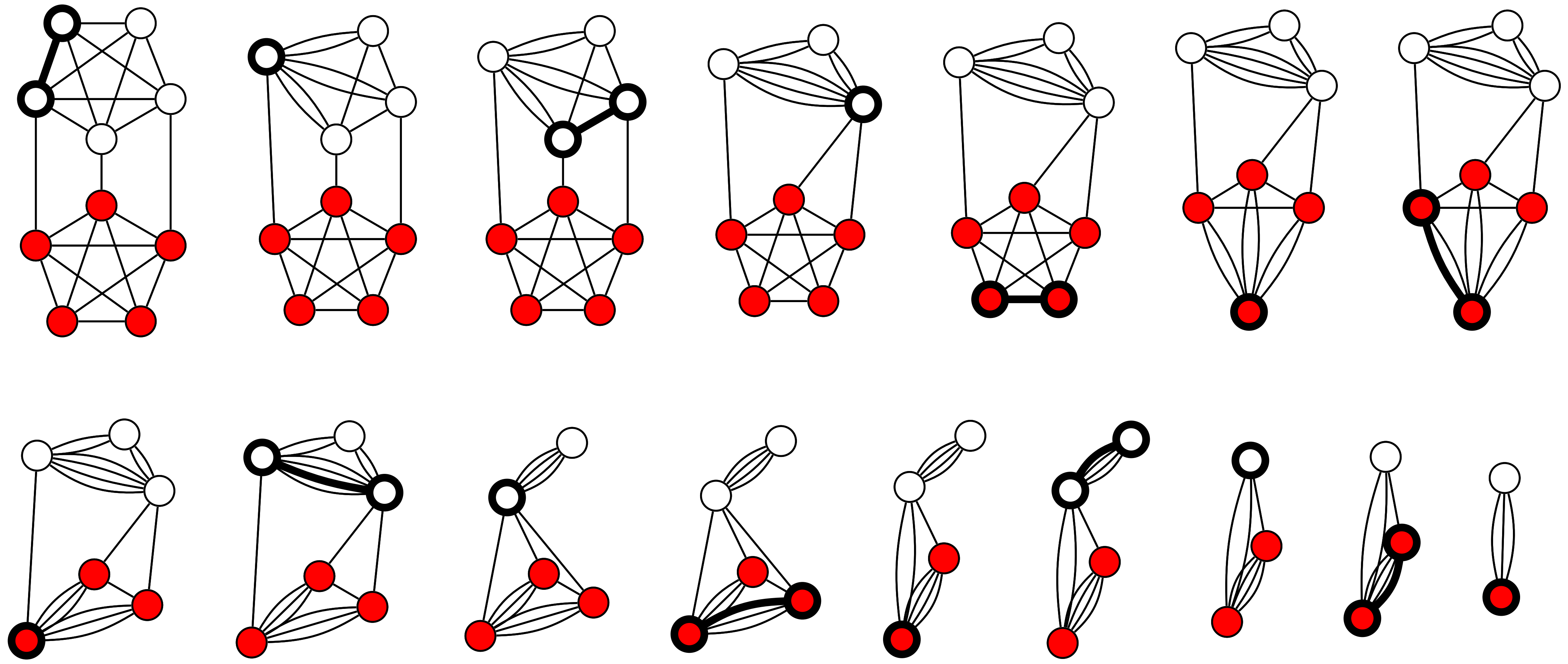
- **Observation**. Any cut in the contracted graph is a cut in the original graph
- Let $C = (S, V - S)$ be any cut, if algorithm never contracts an edge crossing this cut, then it will produce the cut C
- Let C be any arbitrary min cut of cardinality k
- If we pick an edge in G uniformly at random, what is the probability of picking an edge in C?
    - $\dfrac{k}{m}$
- We want to upper bound this probability
- Can we lower bound m in terms of k?



Reference: Thore Husfeldt

Preserving Cuts

- The minimum cut C in G has cardinality k
- What can we say about degree of each vertex in G ?
    - Must be at least k
- G must have at least $nk/2$ edges
- If we pick an edge in G uniformly at random, what is the probability of picking an edge in C?
    - $\dfrac{k}{m} \leq \dfrac{2k}{nk} = \dfrac{2}{n}$



Reference: Thore Husfeldt

Karger's Analysis

- Let C be any arbitrary min cut of cardinality k
  - Pr(picking an edge in C) = $\dfrac{k}{m} \leq \dfrac{k}{nk/2} = \dfrac{2}{n}$
- Probability we don't contract a cut edge in the 1st step $\geq 1 - \dfrac{2}{n}$
- After the first edge is contracted, the algorithm proceeds recursively (with independent random choices) on the $(n-1)$-vertex graph
- Let $P(n)$ denote the probability that the algorithm returns the correct min cut on an n-vertex graph, then

Karger's Analysis

- Let $P(n)$ denote the probability that the algorithm returns the correct min cut on an n-vertex graph, then
- $P(n) \geq \left(1 - \dfrac{2}{n}\right) \cdot P(n-1)$, with base case $P(2) = 1$

- Expanding the recurrence:

$$P(n) \geq \frac{n-2}{n} \cdot \frac{n-3}{n-1} \cdot \frac{n-4}{n-2} \cdot \frac{n-5}{n-3} \ldots\ldots \frac{4}{6} \cdot \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3}$$

- Terms cancel out to get: $P(n) \geq \dfrac{2}{n(n-1)} = 1/\dbinom{n}{2}$

Amplifying Success Probability

- Thus, a single execution of Karger's min cut algorithm finds the min cut with probability at least $1/\binom{n}{2}$, which is low
  - But, we can amplify our success probability!
- Run the algorithm R times (using independent random choices) and pick the best min-cut among them
- What is probability we don't find the min cut after R repetitions?
  - $\left(1 - 1/\binom{n}{2}\right)^R$

Amplifying Success Probability

- If we execute $R = \binom{n}{2}$ times, the probability of failure is
  - $\left(1 - 1/\binom{n}{2}\right)^{\binom{n}{2}}$: how can we simplify this?
  - $\leq \dfrac{1}{e}$

Important Inequality:

$$(1 - x) \leq \left(\frac{1}{e}\right)^{x} \text{ for } x \geq 1$$

- If we set $R = \binom{n}{2} c \ln n$, the failure probability becomes polynomially small in n : $\left(\dfrac{1}{e}\right)^{c \ln n} = \dfrac{1}{n^c}$

With High Probability

- If we run the algorithm $R = \binom{n}{2} c \ln n$ times, we can make the failure probability polynomially small in n: $\left(\frac{1}{e}\right)^{c \ln n} = \frac{1}{n^c}$
- Karger's algorithm finds the min-cut **with high probability (w.h.p.)**

An algorithm is correct **with high probability (w.h.p.)** with respect to input size n if it fails with probability at most $\frac{1}{n^c}$ for any constant $c > 1$.

Example Execution

trial 1

trial 2

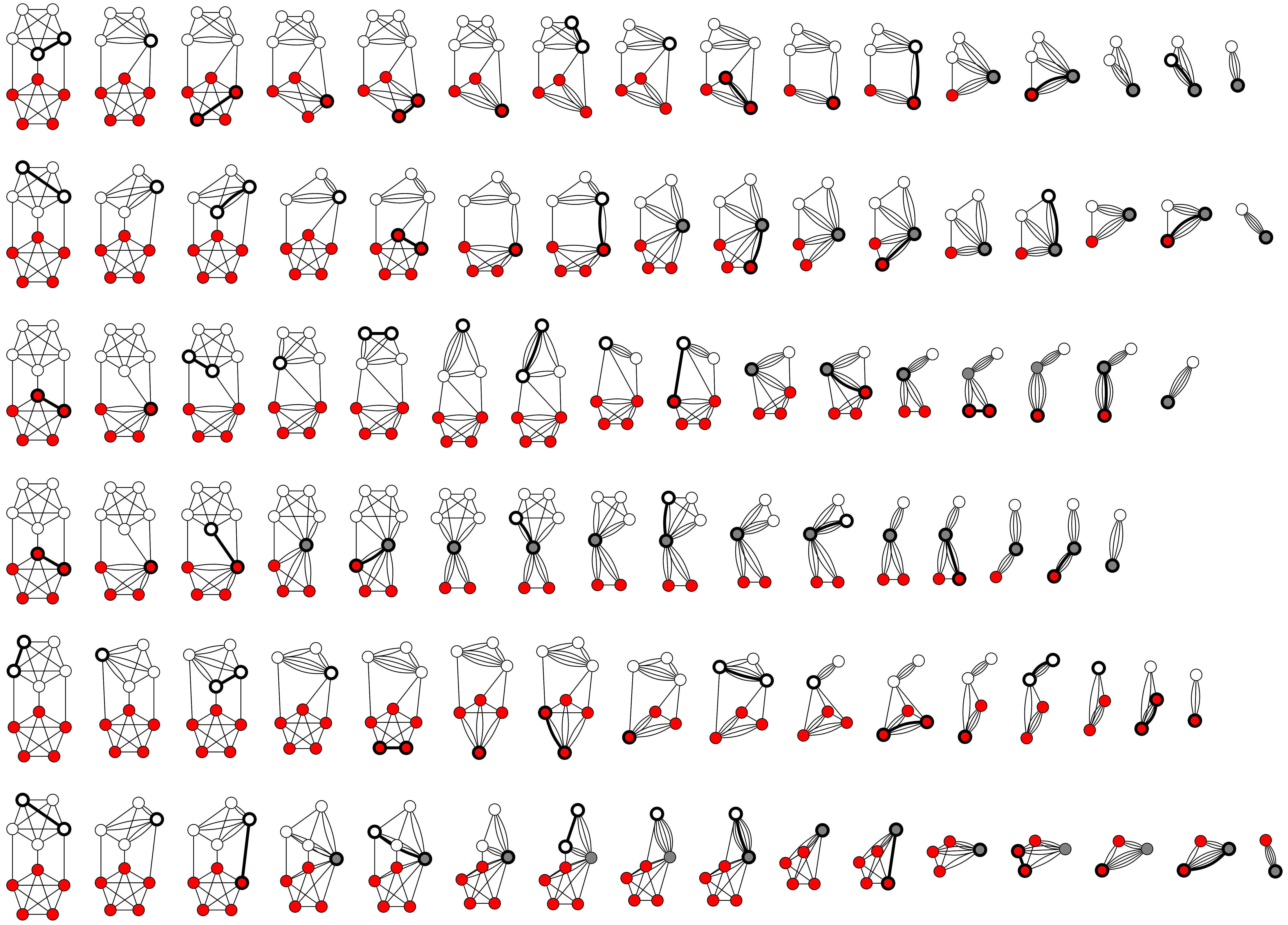
trial 3

trial 4

trial 5
(finds min cut)

trial 6

...



Reference: Thore Husfeldt

Karger's Running Time

- Thus, Karger's algorithm finds the min-cut with high probability (w.h.p.)
- Running time: we perform $\Theta(n^2 \log n)$ iterations, each $O(n^2)$ time
  - $O(n^4 \log n)$ time
  - Faster than naive-flow-techniques, nothing to get excited about
- Improves to $O(n^2 \log^3 n)$ by guessing cleverly! [Karger-Stein 1996]
- **Idea**: Improve the guessing algorithm using the observation:
  - As the graph shrinks, the probability of contracting an edge in the minimum cut increases
  - At first the probability is very small: $2/n$ but by the time there are three nodes, we have a $2/3$ chance of screwing up!

Takeaways

- Karger's algorithm is an example of a "**Monte Carlo**" randomized algorithm
  - Find the correct answer most of the time
- You can increase the success rate of algorithms with one-sided errors by iterating it multiple times and taking the best solution
  - If the probability of success is $1/f(n)$, then running it $O(f(n)\log n)$ times gives a high probability of success
- If you're more intelligent about how you iterate the algorithm, you can often do much better than this
- Next, we'll see an example of a "**Las Vegas**" algorithm
  - Randomized selection and quick sort

Acknowledgments

- Some of the material in these slides are taken from
  - Kleinberg Tardos Slides by Kevin Wayne (https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf)
  - Jeff Erickson's Algorithms Book (http://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf)