CSEP 521: Applied Algorithms
Lecture 3 Randomized Algorithms

Richard Anderson

January 12, 2021

Announcements

- Homework deadline shifted to Thursday (including HW 1)
  - But distribution will remain Tuesdays (HW 2 has been posted)
- Sign up for Gradescope (?)

Randomized Algorithms

- Randomization comes up in many different ways in computation
- Illustrate different ways randomization is used by example algorithms
  - Algorithmic Techniques + Analysis Techniques
  - Demonstrate some incredibly cool idea that are foundational for computing

- K-th largest problem (median problem)
  - Randomization to avoid bad case
  - Random sampling

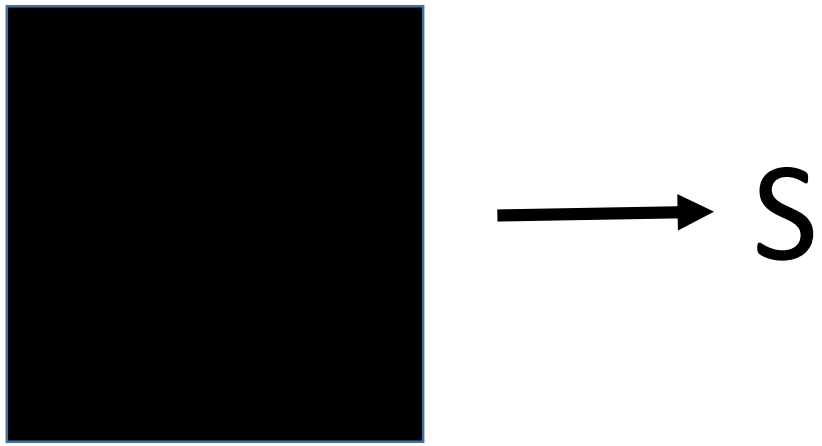
- Random sampling to look for solutions
- Reordering to avoid bad case solutions

Probability theory

- Basic ideas:
  - Sample space and events
  - Random variables
  - Expectation: E(X) – average value of X
  - Linearity of expectation: $E(X + Y) = E(X) + E(Y)$
  - Conditional Probability: $P(A \mid B)$
  - Independence: $P(A \mid B) = P(A)$
- Great review videos
  - Tim Roughgarden (ex-Stanford), Videos 7.1 and 7.2
- Many other references

Randomized Algorithms

- How can random choices be used to make algorithms efficient?
- Suppose you have a minimization problem, where given a solution you can easily compute the value of the solution
- Now suppose you have an algorithm, which returns a specific optimal solutions with probability p.
- Can you find an optimal solution? What can go wrong?



Mincut problem: Karger's Algorithm

- Graph problem
- Random result – might be correct

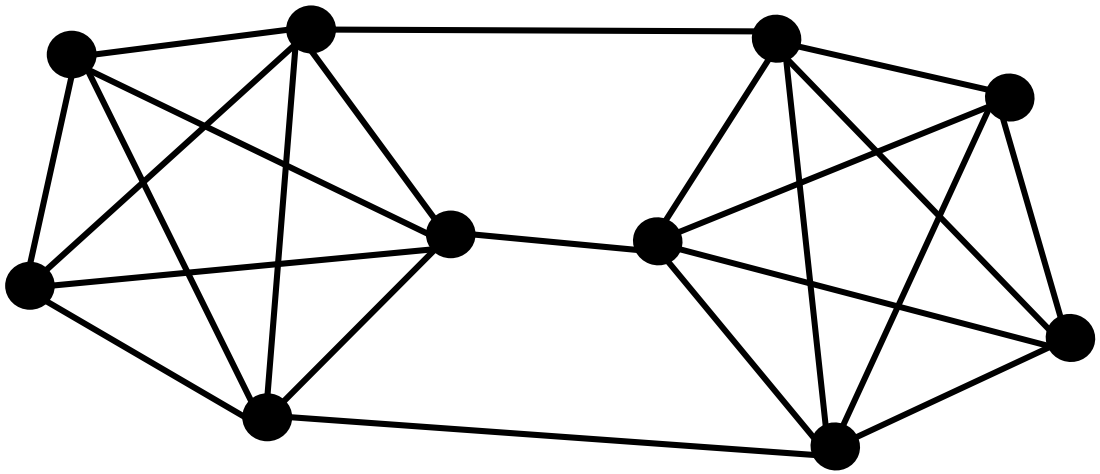
- Given an undirected graph, partition the vertices into two non-empty pieces to minimize the edges between the pieces

- Undirected graph with parallel edges

Notes:
Different from the standard mincut problem presented along with Network Flow
The algorithm presented is not very efficient, but can be improved to be faster than any known deterministic algorithm for the problem

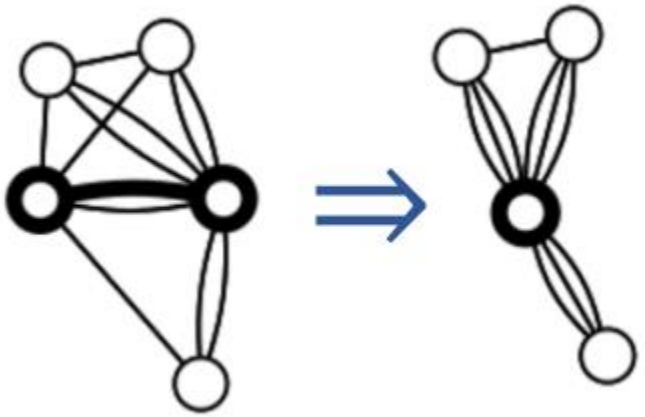
Minimum Cut Problem

- Given an undirected graph with parallel edges, find non-empty sets S and S’ that minimized the number of edges between S and S’



Edge contraction operation

Contraction operation:
For an edge $e \in E$, write G/e for the new graph formed by contracting the edge e.



Observation: The number of edges in the mincut of G/E is at least as great as the number of edges in a mincut of G

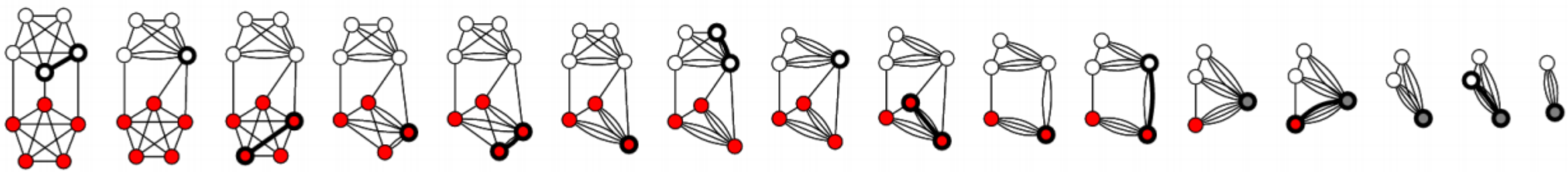
Randomized Algorithm

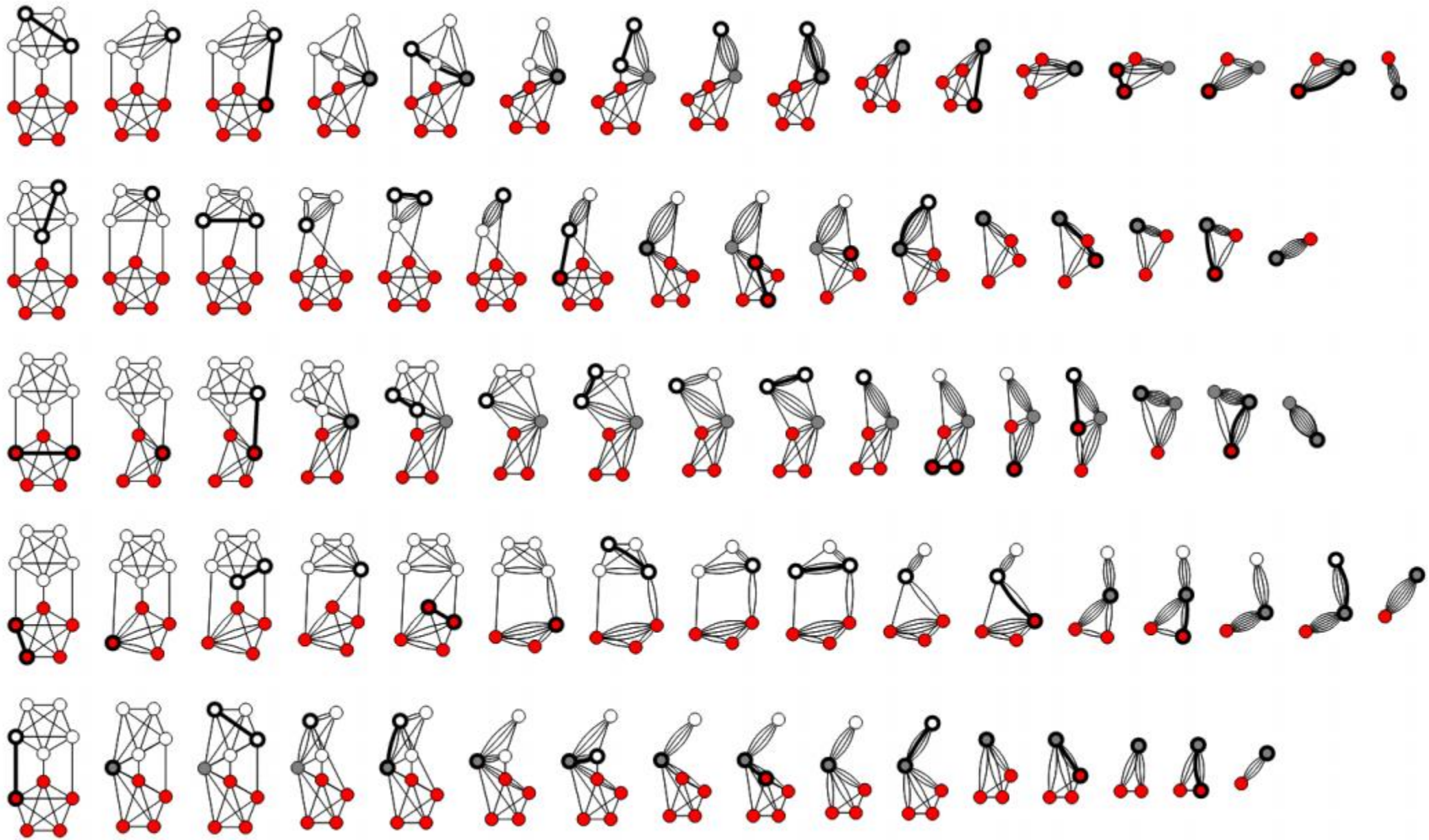
```
procedure contract(G = (V, E)):
while |V| > 2
    choose e ∈ E uniformly at random
    G ← G/e
return the only cut in G
```

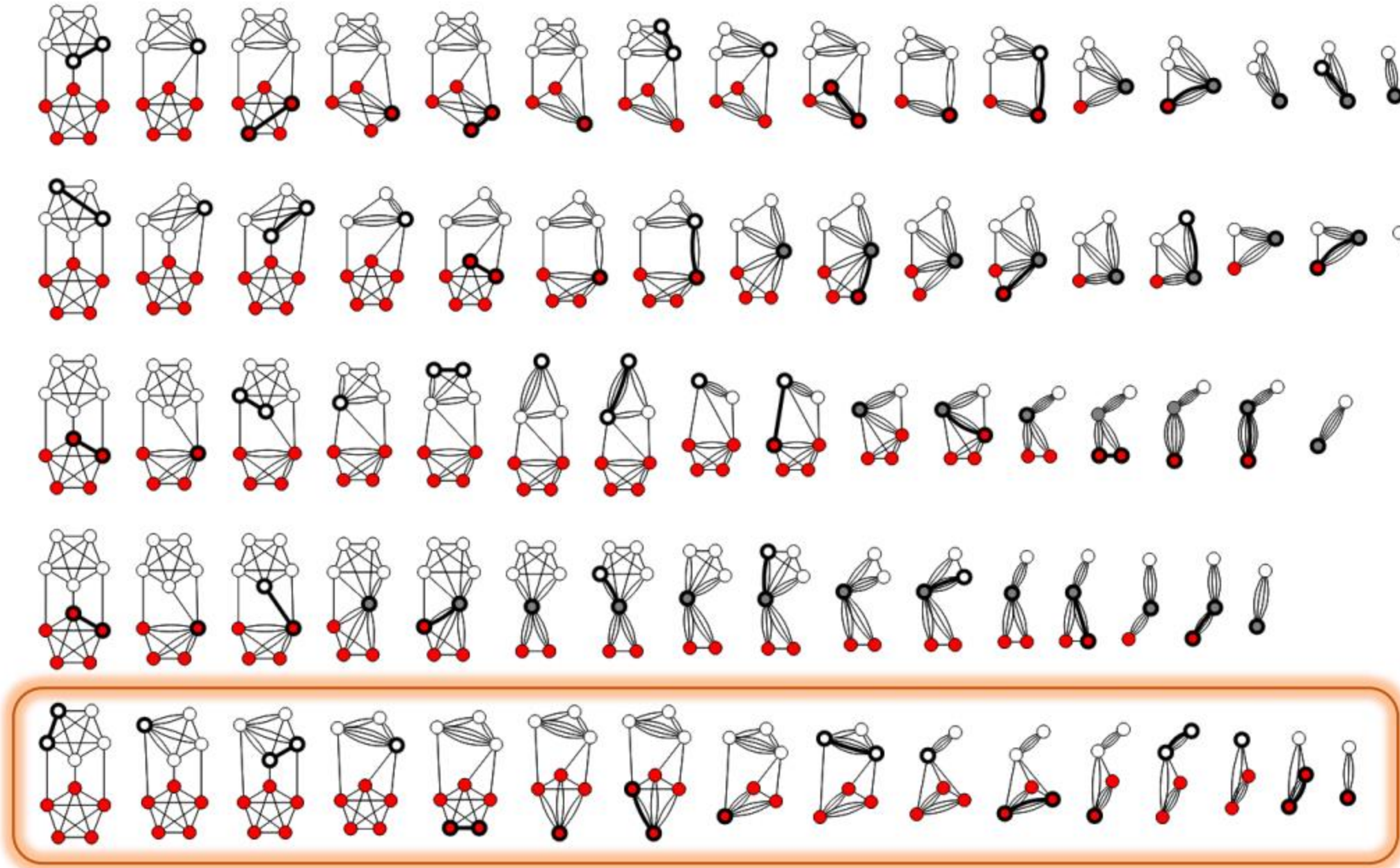
How many times does the while loop execute?

$|V| - 2$ times

```
procedure contract(G = (V, E)):
while |V| > 2
    choose e ∈ E uniformly at random
    G ← G/e
return the only cut in G
```







Analysis

- Let C be a min cut (where C is the set of edges between sets of vertices S and S' and |C| = k). What is the probability that C survives the contraction process?

- G has at least kn/2 edges, otherwise G would have a vertex of degree less than k (and consequently a cut of size less than k).

- Probability that the first edge picked is in C is at most k / (nk/2) = 2/n

Conditional Probabilities

E_i: Event of not picking an edge from C on the i-th step

Probability that no edges from C are picked:

$$p = \text{Prob}[E_1]\ \text{Prob}[E_2 \mid E_1]\ \text{Prob}[E_3 \mid E_1\ E_2] \ldots \text{Prob}[E_{n-2} \mid E_1\ E_2 \ldots E_{n-3}]$$

$$\text{Prob}[E_i \mid E_1\ E_2 \ldots E_{i-1}] >= 1 - 2 / (n- i + 1) = (n - i - 1) / (n - i + 1)$$

$$p >= (n-2)/ n * (n-3) / (n-1) * (n-4) / (n-2)* \ldots * 2/4 * 1/3 = 2 / n(n-1)$$

Min Cut Result

- Probability that C survives is at least 2 / n(n-1) on one iteration
- If we repeat the algorithm n(n-1) / 2 times we have probability at least 1 - 1/e that C has survived at least once

- Mathematical fact: $(1 – 1/k)^k < 1/e$

- If we run $O(n^2 \log n)$ iterations, we have the probability of at least $1 – 1/n$ of finding a mincut

Binary planar partition

- S = {s_1, s_2, . . . s_n}: non-intersecting line segments in the plane

- Binary tree, each node v of the tree is a region r(v) of the plane and has a line l(v) which separates r(v) into regions r_1(v) and r_2(v) which are associated with it children
- The line segments are then cut up and assigned to the associated leaf regions

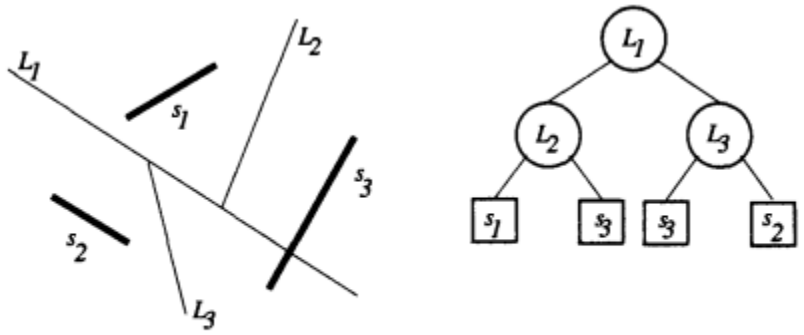


Figure 1.2: An example of a binary planar partition for a set of segments (dark lines). Each leaf is labeled by the line segment it contains. The labels $r(v)$ are omitted for clarity.

Motivation: hidden surface elimination

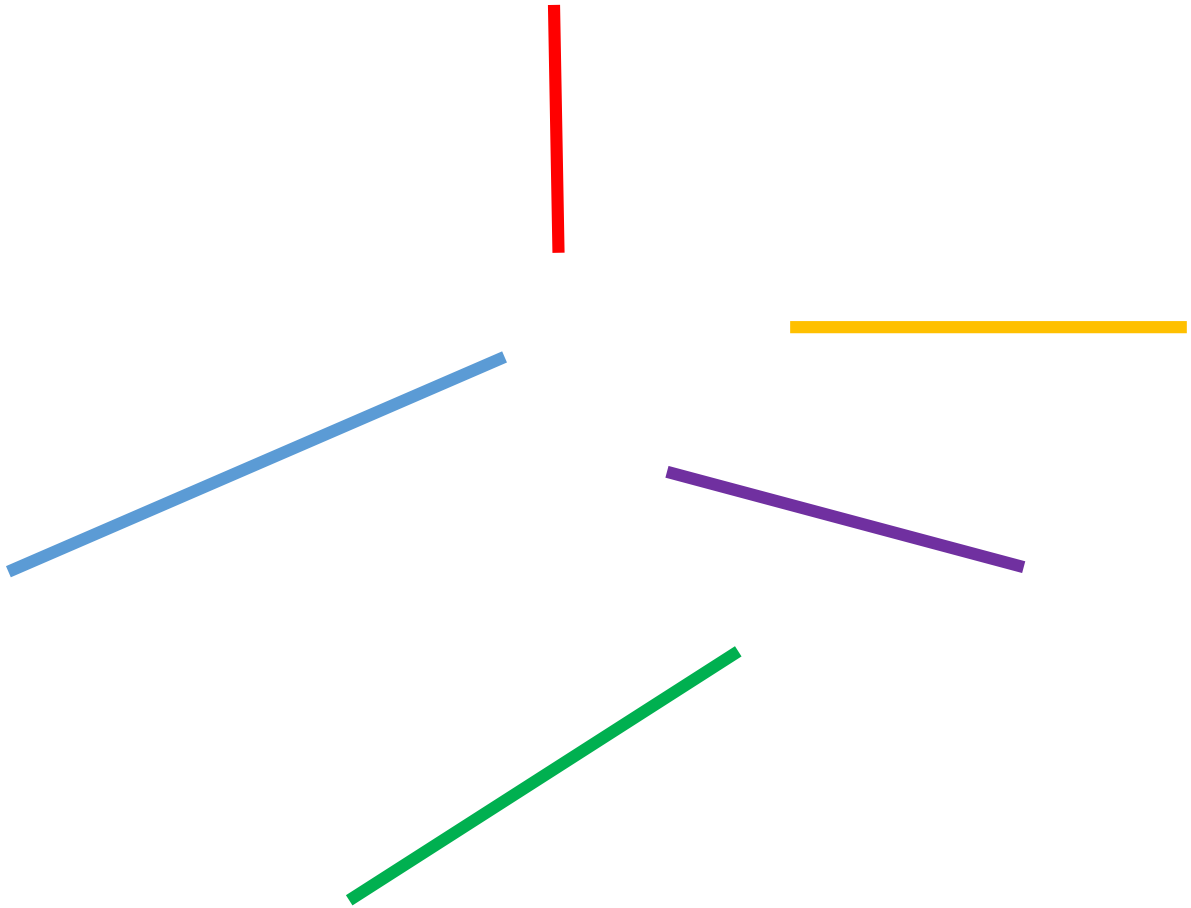
- 2-D Version, draw the line segments visible from any point in space
  - Painters algorithm: draw the lines from back to front in viewing direction

- Tree traversal algorithm. Determine which side of the line you are on for each node. Traverse the subtree on the other side of the line, then the subtree on your side of the line

- Run time is proportional to the size of the tree
  - Tree size O(n)? O(n log n)? $O(n^2)$?

Greedy Algorithm

- Permutation of line segments
- While a region has more than one segment
  - Choose the next segment, and use its line to split segments



Warmup exercises (breakout)

- Show that there exists a set of line segments for which no binary planar partition can avoid breaking up some of the segments into pieces, if each segment I to lie in a different region of the partition
- What is a bad case of partitioning
  - Layout + choice of tree

Random algorithm

- Choose the segments in a random order
- Result: Expected number of partitions is O(n log n)

Argument

- index (u, v) is 1 + number of segments intersected by l(u) before hitting v, and is infinity if l(u) does not intersect v

- index(u,v_1) = index(u,v_2) = 2

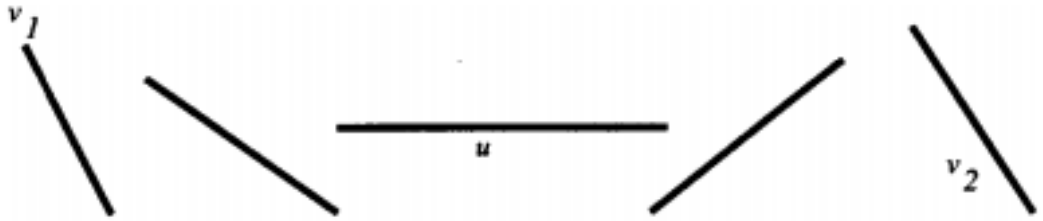


Figure 1.3: An illustration of *index*(u, v).

Mathy stuff

- Cut(u,v) the event that l(u) cuts v
- Suppose index(u,v) = i
- Cut(u,v) happens only if u occurs before any of u_1, u_2, …, u_{i-1}, v in the random permutation
- C_{uv} indicator variable for Cut(u,v)
- $E[C_{uv}] = Pr(Cut(u,v)) = 1/(i + 1)$

Completing the proof

- Size of the partition is n + number of cuts
- Expected size n + E(Sum_{uv} C_{uv})
- Line segment u has at most 2 segments v and w with index(u,v)= i and index(u,w) = i
- Bound n + $2nH_n$

Summary of result

- Binary Space Partition trees are small with random ordering
- This result generalizes to higher dimensions where it is useful for hidden surface elimination
  - Leads to simple algorithms that perform well

- Use of randomization to defeat the bad case