Lecture #22

CS 170

Spring 2021

Randomized Algorithms

- When we can't be both fast and correct, settle for
- Correct, and probably fast (Las Vegas)
    - Ex: Quicksort, choose random "pivot"
    - $t(x,r)$ = runtime on input x, random variable r
    - Expected runtime $T(n)$ on problem of size $|x|=n$:

$$T(n) = \max_{|x|=n} \mathbb{E}\, t(x,r) = \text{average runtime for worst input}$$

- Fast, and probably correct (Monte Carlo)
    - Ex: Freivald: test if $A \cdot B = C$ by testing if $A \cdot (B \cdot x) = C \cdot x$ for k random x, $\mathbb{P}(\text{correct}) \geq 1 - \frac{1}{2^k}$
    - Ex: Karger: global min cut
    - Ex: Polling, to predict election outcome

Quick Review of Probability (CS70)

- Random variable X takes values $x_1, x_2, \dots$ with probabilities $P(X = x_i) = p_i \geq 0$, $\sum_i p_i = 1$
  - Roll fair die, $X \in \{1 \dots 6\}$ each with $p_i = \frac{1}{6}$
- Expectation of X = "average value of X"
  $= \mathbb{E}(X) = \sum_i x_i p_i$
  - Roll fair die, $\mathbb{E}(X) = \frac{1}{6}(1+2+\cdots+6) = 3.5$
- If X and Y are random variables, $\mathbb{E}(aX+bY) = a\,\mathbb{E}(X) + b\,\mathbb{E}(Y)$
- Markov's Inequality: If $X \geq 0$ then

$$\mathbb{E}(X) = \sum_i x_i p_i \geq \sum_{x_i > t} x_i p_i \geq \sum_{x_i > t} t\, p_i = t\, P(X > t)$$

$$\Rightarrow P(X > t) \leq \frac{\mathbb{E}(X)}{t}$$

Randomized Quicksort of $A(1:n)$

- Assume w.l.o.g. that all $A(i)$ distinct
  - Else lexicographic: $(A(i),i) < (A(j),j)$ if $A(i)<A(j)$ elseif $i<j$

Quicksort$(A(1:n))$:
  if $n=1$ return $A(1)$, else
    pick uniformly random pivot $i \in \{1,\dots,n\}$
    $L \leftarrow \{i : A(i) < A(\text{pivot})\}$
    $R \leftarrow \{i : A(i) > A(\text{pivot})\}$
    return (Quicksort$(A(L))$, $A(\text{pivot})$,
            Quicksort$(A(R))$)

Worst case: $|L| = n-1$, $|R| = 0 \Rightarrow T(n) = T(n-1) + O(n) = O(n^2)$

Best case: $|L| = |R| = \frac{n-1}{2} \Rightarrow T(n) \leq 2T(\frac{n}{2}) + n = O(n \log n)$

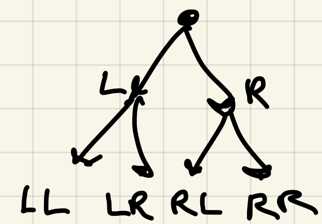
Hope: Expected case same as Best case

Proof that $\mathbb{E}T(n) = O(n \log n)$ (1/2)

- $T(n) = \Theta(\#\text{ comparisons})$
- $X_{ij} = 1$ if i^{th} smallest entry compared to j^{th} smallest, else 0
  - $\#\text{ comparisons} = \sum_{i<j} X_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$
- Each X_{ij} is a random variable so

$$\mathbb{E}(\#\text{ comparisons}) = \sum_{i<j} \mathbb{E}(X_{ij})$$

- How is X_{ij} determined?



  - Let $a_1 < a_2 < \cdots < a_n$ be sorted array
  - $X_{ij} = 1$ iff a_i, a_j in same subarray, and one of them chosen as pivot
  - $X_{ij} = 0$ iff a_i, a_j in same subarray, and one of $a_{i+1}, \ldots, a_{j-1}$ chosen as pivot
  - 2 out of $j-i+1$ ways $X_{ij}=1$, all equally likely

$$\Rightarrow P(X_{ij}=1) = \frac{2}{j-i+1} = \mathbb{E}(X_{ij})$$

Proof that $\mathbb{E}T(n) = O(n \log n)$ (2/2)

- $T(n) = \Theta(\#\text{ comparisons})$
- $X_{ij} = 1$ if i^{th} smallest entry compared to j^{th} smallest, else 0
    - $\#\text{ comparisons} = \sum_{i<j} X_{ij} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$
- Each X_{ij} is a random variable so

$$\mathbb{E}(\#\text{ comparisons}) = \sum_{i<j} \mathbb{E}(X_{ij}) = \sum_{i<j} \frac{2}{j-i+1}$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \leq \sum_{i=1}^{n} \sum_{j=i+1}^{n+i-1} \frac{2}{j-i+1} = \sum_{i=1}^{n} 2\left(\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}\right)$$

$$= 2n\left(\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}\right) \approx 2n \log n$$

- Markov inequality:

$$P(\#\text{comparisons} \geq 200\, n \log n) \lesssim \frac{2n \log n}{200\, n \log n} = 1\%$$

Freivald's Algorithm (1/2)

- Given $n \times n$ matrices A, B, C, test whether $C = A \cdot B$ faster than multiplying $A \cdot B$
- Intuition: if $C \neq A \cdot B$ and x is a random vector, then "probably" $C \cdot x \neq (A \cdot B) \cdot x = A \cdot (B \cdot x)$ which costs just n^2 to test
- Thm: if $x \in \{0,1\}^n$ chosen with each x_i independent, $P(x_i = 0) = \frac{1}{2} = P(x_i = 1)$, and $C \neq A \cdot B$, then $P(Cx \neq ABx) \geq \frac{1}{2}$
- Cor: If we choose N random x, probability that $Cx \neq ABx$ at least once is $\geq 1 - \frac{1}{2^N}$

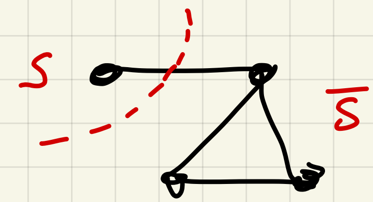
$\Rightarrow$ can make $P(\text{correct})$ as chose to 1 as desired.

Freivald's Algorithm

- Thm: if $x \in \{0,1\}^n$ chosen with each entry x_i independent, $P(x_i=0) = \frac{1}{2} = P(x_i=1)$, and $C \neq A \cdot B$, then $P(Cx \neq ABx) \geq \frac{1}{2}$
- Proof: Let $D = C - A \cdot B \neq 0 \Rightarrow$ some column $D_i \neq 0$
  Let $x \in \{0,1\}^n$, $x' = x$ with $x_i' = 1 - x_i$ so $x' = x \pm e_i$. Then $Dx' = D(x \pm e_i) = Dx \pm D_i$. $D_i \neq 0 \Rightarrow Dx$ and Dx' can't both be zero. All 2^n vectors x come in pairs (x, x'). Since at least one of $Dx, Dx' \neq 0$ from each pair, $P(Dx \neq 0) \geq \frac{1}{2}$.

Karger's Global Mincut Algorithm (1/6)

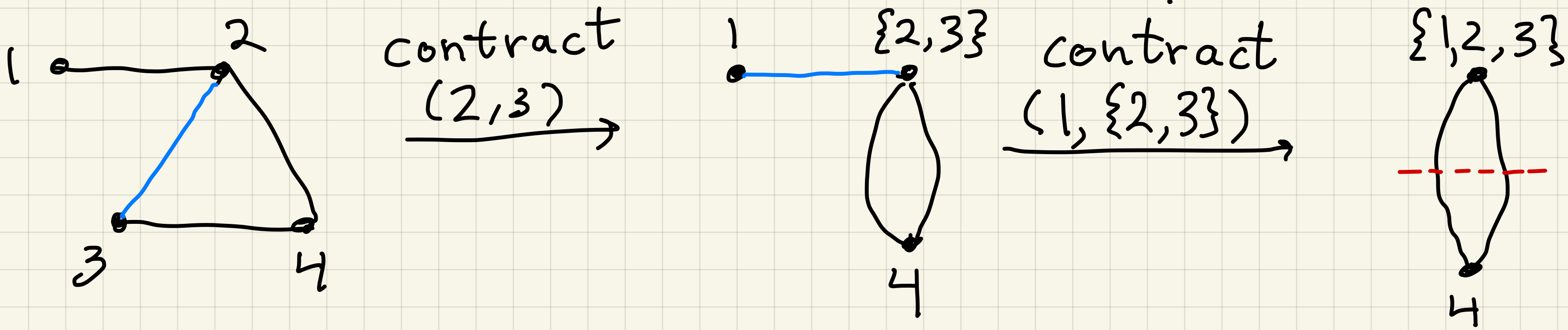
- Def: A cut of an undirected graph $G(V,E)$ is a partition $V = S \cup \bar{S}$, $S \cap \bar{S} = \emptyset$, $S \neq \emptyset$, $\bar{S} \neq \emptyset$
- Def: Size of a cut = # edges connecting $S, \bar{S}$ $= |E \cap (S \times \bar{S})|$
- Def: Global Mincut (GMC) $= (S, \bar{S})$ minimizing size



- Deterministic algorithm:
  - choose $s=1$, for $t=2$ to n, run Ford-Fulkerson to find maxflow from $s \to t$ choose smallest
  - Cost $= O(|V| \cdot (|V| \cdot |E|)) = O(|V|^2 |E|)$
- Can we go faster?

Karger's Global Mincut Algorithm (2/6)

- Def: Given $G(V,E)$, contract(e), $e=(u,v)$ means
  1) replace vertices u,v by one $\{u,v\}$
  2) remove edge (u,v)
  3) replace other (u,w) by $(\{u,v\},w)$ and (v,w) by $(\{u,v\},w)$



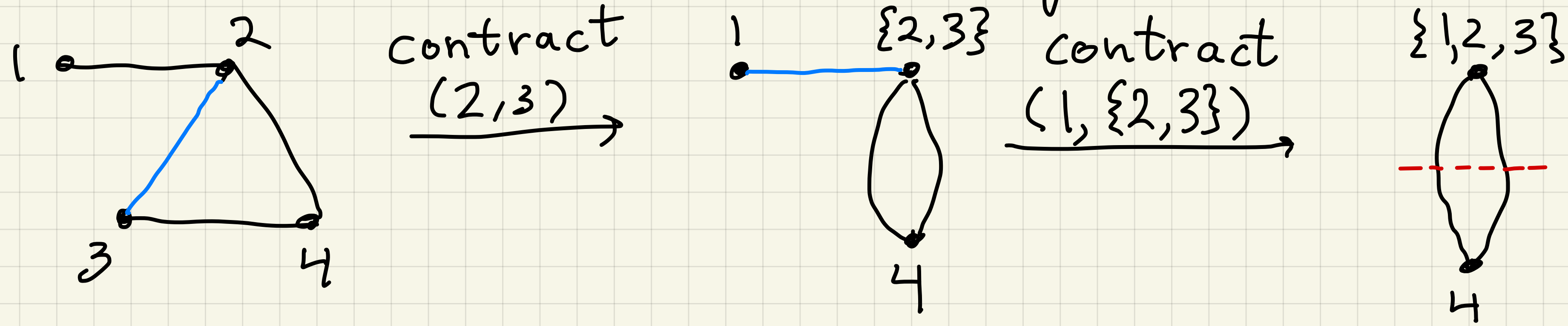
- Karger's Algorithm:

  for $i = 1$ to $|V|-2$

  pick random edge e, contract(e)

  return cut determined by last 2 vertices

Karger's Global Mincut Algorithm (3/6)



- Karger's Algorithm:
  - for $i = 1$ to $|V| - 2$
    - pick random edge e, contract(e)
  - return cut determined by last 2 vertices
- Fact: Karger returns global mincut (GMC) $(S, \bar{S})$
  - $\iff$ last two vertices $= S$ and $\bar{S}$
  - $\iff$ never contracts an edge in GMC
- What is probability that Karger never contracts an edge in GMC?

Karger's Global Mincut Algorithm (4/6)

- Karger's Algorithm:

    for $i = 1$ to $|V|-2$ ... $n = |V|$ below

        pick random edge e, contract(e)

    return cut determined by last 2 vertices

- Fact: Karger returns GMC $(S, \bar{S})$
  $\Leftrightarrow$ never contracts an edge in GMC
- What is probability that Karger never contracts an edge in GMC?
- m_i = # edges, K = size of GMC (# edges from S to $\bar{S}$)
- P(don't pick e in GMC at step i) $= 1 - \frac{K}{m_i}$
- P(don't pick e in GMC at any step) $= \prod_{i=1}^{n-2} \left(1 - \frac{K}{m_i}\right)$

    since each contracted graph has same GMC

Karger's Global Mincut Algorithm (5/6)

- Karger's Algorithm:

  for $i = 1$ to $|V|-2$
      pick random edge e, contract(e)
  return cut determined by last 2 vertices

- $m_i = \#$ edges, $k =$ size of GMC (# edges from S to $\bar{S}$)
- $\deg(v) \geq k$ (else GMC smaller than k)

  $\Rightarrow m_i = \frac{1}{2} \sum_v \deg(v) \geq \frac{1}{2} \sum_v k = \frac{k}{2}(\#\text{vertices}) = \frac{k}{2}(n-i+1)$

- P(Karger gets right answer)

  $= P$(don't pick e in GMC at any step) $= \prod_{i=1}^{n-2} \left(1 - \frac{k}{m_i}\right)$

  $\geq \prod_{i=1}^{n-2} \left(1 - \frac{2}{n-i+1}\right) = \prod_{i=1}^{n-2} \left(\frac{n-i-1}{n-i+1}\right) = \frac{\cancel{n-2}}{n} \frac{\cancel{n-3}}{n-1} \frac{\cancel{n-4}}{\cancel{n-2}} \frac{n-5}{\cancel{n-3}} \cdots \frac{2}{\cancel{4}} \cdot \frac{1}{\cancel{3}}$

  $= \frac{2}{n(n-1)} = 1 / \binom{n}{2}$

Karger's Global Mincut Algorithm (6/6)

- Karger's Algorithm:

```
for i = 1 to |V|-2
    pick random edge e, contract(e)
return cut determined by last 2 vertices
```

- $P(\text{Karger gets right answer}) \geq 1 / \binom{n}{2}$
- Make $P(\text{right answer})$ larger:
    - Repeat Karger N times, choose best (smallest) cut
- To make $P(\text{wrong answer}) \leq p$, let $N = \lceil \binom{n}{2} \ln\left(\frac{1}{p}\right) \rceil$:

$$\left(1 - \frac{1}{\binom{n}{2}}\right)^N \leq \left(e^{\frac{-1}{\binom{n}{2}}}\right)^N \quad \text{since } 1+x \leq e^x$$

$$\leq e^{-\ln\left(\frac{1}{p}\right)} = p$$

- Total Cost $= O(|E| \cdot N) = O(|E| \cdot |V|^2)$
  like Ford-Fulkerson

One more "hot topic"

- Lots of current research on randomized linear algebra algorithms
  - Least squares problems $\min_x \|Ax-b\|_2$
  - PCA (Principle Component Analysis)
  - SVD (Singular Value Decomposition)
- see "References for Randomized Algorithms" at people.eecs.berkeley.edu/~demmel/ma221_Fall20
- High level common approach: replace A by RA, R = random matrix, solve using RA instead