Class 2: Randomized Algorithms
Online Class

• Attend synchronous class (if possible), participate, ask questions

On Zoom:

• Use the largest display you have. Your phone is not a good choice
• Join Zoom with video ON and audio Muted
• View options: Fit to Window
• To participate:
  • Use Raise Hand in Reactions
  • Unmute and talk
  • Chat to Everyone or Host.
Min-Cut Algorithm

**Input:** An $n$-node graph $G$.

**Output:** A minimal set of edges that disconnects the graph.

1. **Repeat** $n - 2$ times:
   1. Pick an edge uniformly at random.
   2. Contract the two vertices connected by that edge, eliminate all edges connecting the two vertices.

2. Output the set of edges connecting the two remaining vertices.

**Theorem**

The algorithm outputs a min-cut edge-set with probability

$$\geq \frac{2}{n(n-1)}.$$

What’s the probability space? The space changes each step.
Conditional Probabilities

**Definition**

The **conditional probability** that event $E_1$ occurs given that event $E_2$ occurs is

$$
\Pr(E_1 \mid E_2) = \frac{\Pr(E_1 \cap E_2)}{\Pr(E_2)}.
$$

The conditional probability is only well-defined if $\Pr(E_2) > 0$.

By conditioning on $E_2$ we restrict the sample space to the set $E_2$. Thus we are interested in $\Pr(E_1 \cap E_2)$ “normalized” by $\Pr(E_2)$. A condition $E_2$ defines a new sample space, with a new probability function $P(\cdot \mid E_2)$. 
Analysis of the Algorithm

Assume that the graph has a min-cut set of $k$ edges. We compute the probability of finding one such set $C$.

$$\Pr(\text{Alg. returns any minimal cut set}) \geq \Pr(\text{Alg. returns } C)$$

Two parts proof:

- **Deterministic analysis part:**
  
  **Lemma**
  
  *If no edge of $C$ was contracted, the algorithms outputs $C$.*

- **Probabilistic analysis part:**
  
  **Lemma**
  
  $$\Pr(\text{no edge of } C \text{ is contracted}) \geq \frac{2}{n(n-1)}.$$
Deterministic part:

**Lemma**

If no edge of $C$ was contracted, no edge of $C$ was eliminated.

**Proof.**

Let $X$ and $Y$ be the two set of vertices cut by $C$. If the contracting edge connects two vertices in $X$ (res. $Y$), then all its parallel edges also connect vertices in $X$ (res. $Y$).

**Corollary**

If the algorithm terminates before contracting any edge of $C$, the algorithm gives a correct answer.

**Corollary**

Vertex contraction does not reduce the size of the min-cut set. Every cut set in the new graph is a cut set in the original graph.
Probabilistic Analysis:

- Let $E_i = \text{"the edge contracted in iteration } i \text{ is not in } C\text{."}$
- Let $F_i = \cap_{j=1}^{i} E_j = \text{"no edge of } C \text{ was contracted in the first } i \text{ iterations"}.$

Since the minimum cut-set has $k$ edges, all vertices have degree $\geq k$, and the graph has $\geq nk/2$ edges.

- There are at least $nk/2$ edges in the graph, $k$ edges are in $C$. Thus, $Pr(E_1) = Pr(F_1) \geq 1 - \frac{2k}{nk} = 1 - \frac{2}{n}$.

- Conditioning on $E_1$, after the first vertex contraction we are left with an $n - 1$ node graph, with minimum cut set, and minimum degree $\geq k$. The new graph has at least $k(n - 1)/2$ edges, thus $Pr(E_2 \mid F_1) \geq 1 - \frac{k}{k(n-1)/2} \geq 1 - \frac{2}{n-1}$.

- Similarly, $Pr(E_i \mid F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}.$
Useful identities:

$$Pr(A \mid B) = \frac{Pr(A \cap B)}{Pr(B)}$$

$$Pr(A \cap B) = Pr(A \mid B)Pr(B)$$

$$Pr(A \cap B \cap C) = Pr(A \mid B \cap C)Pr(B \cap C)$$

$$= Pr(A \mid B \cap C)Pr(B \mid C)Pr(C)$$

Let $E_1, \ldots, E_n$ be a sequence of events. Let $F_i = \bigcap_{j=1}^{i} E_i$

$$Pr(F_n) = Pr(E_n \mid F_{n-1})Pr(F_{n-1}) =$$

$$Pr(E_n \mid F_{n-1})Pr(E_{n-1} \mid F_{n-2}) \ldots P(E_2 \mid F_1)Pr(F_1)$$
We need to compute

\[ Pr(F_{n-2}) = Pr(\cap_{j=1}^{n-2} E_j) \]

We have

\[ Pr(E_1) = Pr(F_1) \geq 1 - \frac{2k}{nk} = 1 - \frac{2}{n} \]

and

\[ Pr(E_i \mid F_{i-1}) \geq 1 - \frac{k}{k(n-i+1)/2} = 1 - \frac{2}{n-i+1}. \]

\[ Pr(F_{n-2}) = Pr(E_{n-2} \cap F_{n-3}) = Pr(E_{n-2} \mid F_{n-3}) Pr(F_{n-3}) = \]

\[ Pr(E_{n-2} \mid F_{n-3}) Pr(E_{n-3} \mid F_{n-4}) \ldots Pr(E_2 \mid F_1) Pr(F_1) = \]

\[ Pr(F_1) \prod_{j=2}^{n-2} Pr(E_j \mid F_{j-1}) \]
The probability that the algorithm computes the minimum cut-set is

\[
Pr(F_{n-2}) = Pr(\cap_{j=1}^{n-2} E_j) = Pr(F_1) \prod_{j=2}^{n-2} Pr(E_j | F_{j-1})
\]

\[
\geq \prod_{i=1}^{n-2} \left( 1 - \frac{2}{n - i + 1} \right) = \prod_{i=1}^{n-2} \left( \frac{n - i - 1}{n - i + 1} \right)
\]

\[
= \left( \frac{n - 2}{n} \right) \left( \frac{n - 3}{n - 1} \right) \left( \frac{n - 4}{n - 2} \right) \cdots \frac{2}{n(n - 1)}.
\]
Theorem

Assume that we run the randomized min-cut algorithm \(n(n-1)\log n\) times and output the minimum size cut-set found in all the iterations. The probability that the output is not a min-cut set is bounded by \(\frac{1}{n^2}\).

Proof.

The algorithm has a one side error: the output is never smaller than the min-cut value. The probability that \(C\) is not the output of any of the \(n(n-1)\log n\) runs is

\[
\leq \left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)\log n} \leq e^{-2\log n} = \frac{1}{n^2}.
\]
\[
\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1) \log n} \leq e^{\log n} = \frac{1}{n^2}.
\]

The Taylor series expansion of \(e^{-x}\) gives
\[
e^{-x} = 1 - x + \frac{x^2}{2!} - \ldots.
\]

Thus, for \(x < 1\),
\[
1 - x \leq e^{-x}.
\]

**Theorem**

1. The algorithm outputs a min-cut edge set with probability 
   \[\geq \frac{2}{n(n-1)}.\]

2. The smallest output in \(O(n^2 \log n)\) iterations of the algorithm 
   gives a correct answer with probability \(1 - 1/n^2\).
Example: Finding the \( k \)-Smallest Element in an ordered set.

Procedure \( \text{Order}(S, k) \);

**Input:** A set \( S \), an integer \( k \leq |S| = n \).

**Output:** The \( k \) smallest element in the set \( S \).
Example: Finding the $k$-Smallest Element

Procedure Order($S, k$);

**Input:** A set $S$, an integer $k \leq |S| = n$.

**Output:** The $k$ smallest element in the set $S$.

1. If $|S| = k = 1$ return $S$.
2. Choose a random element $y$ uniformly from $S$.
3. Compare all elements of $S$ to $y$. Let $S_1 = \{x \in S \mid x \leq y\}$ and $S_2 = \{x \in S \mid x > y\}$.
4. If $k \leq |S_1|$ return Order($S_1, k$) else return Order($S_2, k - |S_1|$).

**Theorem**

1. The algorithm always returns the $k$-smallest element in $S$.
2. The algorithm performs $O(n)$ comparisons in expectation.
A random variable $X$ on a sample space $\Omega$ is a real-valued function on $\Omega$; that is, $X : \Omega \rightarrow \mathbb{R}$. A discrete random variable is a random variable that takes on only a finite or countably infinite number of values.

Discrete random variable $X$ and real value $a$: the event “$X = a$” represents the set $\{s \in \Omega : X(s) = a\}$.

$$
\Pr(X = a) = \sum_{s \in \Omega : X(s) = a} \Pr(s)
$$
The expectation of a discrete random variable $X$, denoted by $E[X]$, is given by

$$E[X] = \sum_i i \Pr(X = i),$$

where the summation is over all values in the range of $X$. The expectation is finite if $\sum_i |i| \Pr(X = i)$ converges; otherwise, the expectation is unbounded.

The expectation (or mean or average) is a weighted sum over all possible values of the random variable.
The **median** of a random variable $X$ is a value $m$ such

$$\Pr(X < m) \leq 1/2 \quad \text{and} \quad \Pr(X > m) < 1/2.$$
Linearity of Expectation

**Theorem**

For any two random variables $X$ and $Y$

$$E[X + Y] = E[X] + E[Y].$$

**Lemma**

For any constant $c$ and discrete random variable $X$,

$$E[cX] = cE[X].$$
Example: Finding the $k$-Smallest Element

Procedure Order($S, k$);

**Input:** A set $S$, an integer $k \leq |S| = n$.

**Output:** The $k$ smallest element in the set $S$.

1. If $|S| = k = 1$ return $S$.
2. Choose a random element $y$ uniformly from $S$.
3. Compare all elements of $S$ to $y$. Let $S_1 = \{x \in S \mid x \leq y\}$ and $S_2 = \{x \in S \mid x > y\}$.
4. If $k \leq |S_1|$ return Order($S_1, k$) else return Order($S_2, k - |S_1|$).

**Theorem**

1. *The algorithm always returns the $k$-smallest element in $S*
2. *The algorithm performs $O(n)$ comparisons in expectation.*
Proof

• We say that a call to Order($S, k$) was successful if the random element was in the middle $1/3$ of the set $S$. A call is successful with probability $1/3$.

• After the $i$-th successful call the size of the set $S$ is bounded by $n(2/3)^i$. Thus, need at most $\log_{3/2} n$ successful calls.

• Let $X$ be the total number of comparisons. Let $T_i$ be the number of iterations between the $i$-th successful call (included) and the $i + 1$-th (excluded):

$$E[X] \leq \sum_{i=0}^{\log_{3/2} n} n(2/3)^i E[T_i].$$

• $T_i$ has a geometric distribution $G(1/3)$. 
The Geometric Distribution

Definition

A geometric random variable $X$ with parameter $p$ is given by the following probability distribution on $n = 1, 2, \ldots$.

$$\Pr(X = n) = (1 - p)^{n-1} p.$$ 

Example: repeatedly draw independent Bernoulli random variables with parameter $p > 0$ until we get a 1. Let $X$ be number of trials up to and including the first 1. Then $X$ is a geometric random variable with parameter $p$. 
Lemma

Let $X$ be a discrete random variable that takes on only non-negative integer values. Then

$$E[X] = \sum_{i=1}^{\infty} \Pr(X \geq i).$$

Proof.

$$\sum_{i=1}^{\infty} \Pr(X \geq i) = \sum_{i=1}^{\infty} \sum_{j=i}^{\infty} \Pr(X = j)$$

$$= \sum_{j=1}^{\infty} \sum_{i=1}^{j} \Pr(X = j)$$

$$= \sum_{j=1}^{\infty} j \Pr(X = j) = E[X].$$
For a geometric random variable $X$ with parameter $p$,

$$\Pr(X \geq i) = \sum_{n=i}^{\infty} (1 - p)^{n-1} p = (1 - p)^{i-1}.$$ 

$$E[X] = \sum_{i=1}^{\infty} \Pr(X \geq i)$$

$$= \sum_{i=1}^{\infty} (1 - p)^{i-1}$$

$$= \frac{1}{1 - (1 - p)}$$

$$= \frac{1}{p}$$
Proof

- Let $X$ be the total number of comparisons.
- Let $T_i$ be the number of iterations between the $i$-th successful call (included) and the $i + 1$-th (excluded):
  - $\mathbb{E}[X] \leq \sum_{i=0}^{\log_{3/2} n} n(2/3)^i \mathbb{E}[T_i]$.
  - $T_i \sim G(1/3)$, therefore $\mathbb{E}[T_i] = 3$.
  - Expected number of comparisons:
    \[
    \mathbb{E}[X] \leq \sum_{j=0}^{\log_{3/2} n} 3n \left(\frac{2}{3}\right)^j \leq 9n.
    \]

Theorem

1. The algorithm always returns the $k$-smallest element in $S$
2. The algorithm performs $O(n)$ comparisons in expectation.

What is the probability space?
Finding the $k$-Smallest Element with no Randomization

Procedure Det-Order($S$, $k$);

Input: An array $S$, an integer $k \leq |S| = n$.

Output: The $k$ smallest element in the set $S$.

1. If $|S| = k = 1$ return $S$.
2. Let $y$ be the first element in $S$.
3. Compare all elements of $S$ to $y$. Let $S_1 = \{x \in S \mid x \leq y\}$ and $S_2 = \{x \in S \mid x > y\}$.
4. If $k \leq |S_1|$ return Det-Order($S_1$, $k$) else return Det-Order($S_2$, $k - |S_1|$).

Theorem

The algorithm returns the $k$-smallest element in $S$ and performs $O(n)$ comparisons in expectation over all possible input permutations.
Randomized Algorithms:

- Analysis is true for any input.
- The sample space is the space of random choices made by the algorithm.
- Repeated runs are independent.

Probabilistic Analysis:

- The sample space is the space of all possible inputs.
- If the algorithm is deterministic repeated runs give the same output.
A **Monte Carlo Algorithm** is a randomized algorithm that may produce an incorrect solution. For decision problems: A **one-side error** Monte Carlo algorithm errs only one one possible output, otherwise it is a **two-side error** algorithm. A **Las Vegas** algorithm is a randomized algorithm that always produces the correct output. In both types of algorithms the run-time is a random variable.