Today

Randomized Algorithms + Probability Tools
- Resource Contention
- Minimum Cuts

Announcements
- Updated grades
- Updated schedule
- Programming Assignment

Randomized Algorithms
- So far: deterministic algorithms on worst case inputs.
- Why worst case?
  - Enables precise statements
  - But maybe not reflective of real-world instances.
  - Average-case analysis? What distribution?

Randomized Algorithms
- So far: deterministic algorithms on worst case inputs.
- Why deterministic algorithms?
  - Easier to understand, pretty powerful.
- Two types of randomized algorithms:
  - Fail with some small probability.
  - Always succeed but running time is random.
- How powerful are randomized algorithms?

Resource Contention
- tl;dr: Randomization helps with symmetry breaking.
- How do we share a resource in distributed settings?
  - Option #1: Coordination between agents (expensive)
  - Option #2: Randomize access
Resource contention

- $n$ agents $P_1, \ldots, P_n$ and a shared resource.
- At each round, agents can request access to the resource.
  - > 1 requests — conflict, no access.
  - 0 requests — wasted round, no access.
  - 1 request — access granted.
- What is a good decentralized protocol where all agents get regular access?

A centralized solution?

- Round-robin access:
  - Cycle through agents 1, \ldots, n.
  - No wasted round.
  - Each agent gets 1 access every $n$ rounds.
  - After $n$ rounds all agents have accessed.
- But a lot of coordination!

Decentralized solution

On each round, each agent accesses with probability $p$.
First question: What choice for $p$?
- Let $A[i, t]$ be the event that $P_i$ attempts to access on round $t$
  $$ \Pr[A[i, t]] = p, \quad \Pr[A[j, t]] = (1 - p) $$
- Let $S[i, t]$ be the event that $P_i$ successfully accesses on round $t$
  $$ S[i, t] = A[i, t] \cap \left( \bigcap_{j \neq i} A[j, t] \right) $$
  $$ \Pr[S[i, t]] = p(1 - p)^{n-1} $$

How to choose $p$?

$$ f(p) = \Pr[S[i, t]] = p(1 - p)^{n-1} $$

$f(p)$ maximized at $p = 1/n$.

Some asymptotics

$$ \Pr[S[i, t]] = \frac{1}{n} \left( 1 - \frac{1}{n} \right)^{n-1} $$

- $(1 - \frac{1}{n})^n \rightarrow \frac{1}{e}$ and $(1 - \frac{1}{n})^{n-1} \rightarrow \frac{1}{e}$

- $\frac{1}{en} \leq \Pr[S[i, t]] \leq \frac{1}{2n}$

Decentralized Solution

- Success probability is $\Theta(1/n)$ in one round
- What about for multiple rounds?
  - Def: $F[i, t]$ = event that $i$ unsuccessful through round $t$
  $$ \Pr[F[i, t]] = \Pr[\bigcap_{r=1}^{t} S[i, r]] $$
  $$ = \prod_{r=1}^{t} \Pr[S[i, r]] $$
  $$ = \left[ 1 - \frac{1}{n} \left( 1 - \frac{1}{n} \right)^{n-1} \right]^t \leq \left( 1 - \frac{1}{en} \right)^t $$
  - With $t = \lfloor en \rfloor$, $\Pr[F[i, t]] \leq \frac{1}{e}$
  - With $t = \lfloor en \rfloor \cdot c \ln n$, $\Pr[F[i, t]] \leq n^{-c}$. 
### Comparison

<table>
<thead>
<tr>
<th></th>
<th>Centralized</th>
<th>Decentralized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rounds until individual success</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n \ln n)$ w.h.p.</td>
</tr>
<tr>
<td>Rounds until everyone succeeds</td>
<td>$\Theta(n)$</td>
<td>$\Theta(n \ln n)$ w.h.p.</td>
</tr>
</tbody>
</table>

### Decentralized Solution

- How long until all agents access at least once?
- Let $F_t$ denote event that some agent failed after $t$ rounds

\[
\Pr[F_t] = \Pr\left[ \bigcup_{i=1}^{n} F[i, t] \right] \leq \sum_{i=1}^{n} \Pr[F[i, t]]
\]

Set $t = 2\lceil en \rceil \ln n$

\[
\Pr[F_t] \leq \sum_{i=1}^{n} n^{-2} = n^{-1}
\]

**Theorem**

With probability at least $1 - 1/n$ all agents succeed in accessing the resource within $2\lceil en \rceil \ln n$ rounds.

### Resource Contention Takeaways

- Simple randomization good for symmetry breaking.
- Technical tools:
  - Intersection of independent events
  - Some calculus
  - Union bound

### Minimum Cuts

**Problem.** Given undirected $G = (V, E)$, partition $V$ into sets $A, V \setminus A$ to minimize,

\[
cut(A) = |\{(u, v) \in E, u \in A, v \notin A\}|
\]

- Previously, we saw how to compute minimum $s - t$ cut in directed graph.
- How do we compute global minimum cut?

### Deterministic Algorithm

**Idea.** Convert into $s - t$ cut in directed graph.
Replace $e = (u, v)$ with directed edges in both directions (with capacity 1).
Pick arbitrary $s$.
for each other vertex $t$ do
  Compute minimum $s - t$ cut.
end for
Return smallest computed $s - t$ cut.

**Running Time.** $n$ max-flow computations $\Rightarrow O(nm^2)$ at best.
Contraction Algorithm Preliminaries

**Def.** Multigraph $G = (V, E)$ is a graph that can have parallel edges.

**Def.** Contracting an edge $(u, v)$ in $G = (V, E)$ produces a new multigraph $G' = (V', E')$

- With new node $w$ instead of $u, v$ ($(u, v)$ edges deleted).
- If $(x, u)$ or $(x, v) \in E$, then $(x, w) \in E'$.
- All other edges preserved.

Contraction Algorithm

$S(v) = \{v\}$ for all $v \in V$.

while $|V| > 2$ do

Pick edge $(u, v) \in E$ uniformly at random.

Contract edge $(u, v)$ to get $G'$ with new node $w$

Set $S(w) \leftarrow S(u) \cup S(v)$.

Update $G \leftarrow G'$.

end while

Return $S(v)$ for $v \in V$.

Contraction Algorithm Analysis

**Theorem.** Alg finds global min cut with probability at least $1/n^2$.

**Proof.** Suppose $(A, B)$ is a global min cut with $\text{cut}(A, B) = k$

- What could go wrong in the first step?
  - Select $(u, v)$ where $u \in A, v \in B$.
  
  $\Pr[\text{mistake in round 1}] = \Pr[select\ u \in A, v \in B] = \frac{k}{\#\ of\ edges}$

- $\#\ of\ edges \geq \frac{1}{4}kn$ since if $\deg(w) < k$ ($\{w\}, V \setminus \{w\}$) is smaller cut!

Final steps

- Let $E_j$ be the event that $(A, B)$ is not contracted in round $j$

  $\Pr[E_j|E_1 \cap \ldots \cap E_{j-1}] \geq 1 - \frac{2}{n-j+1}$

  $\Pr[E_1 \cap \ldots \cap E_{n-2}]$

  $\Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \ldots \cdot \Pr[E_{n-2}|E_1 \cap \ldots \cap E_{n-3}]$

  $\geq \left(1 - \frac{2}{n}\right) \left(1 - \frac{2}{n-1}\right) \ldots \left(1 - \frac{2}{3}\right)$

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

Contraction Algorithm Analysis

$\Pr[\text{mistake in round 1}] \leq \frac{k}{\frac{1}{2}kn} = \frac{2}{n}$

- Consider round $j + 1$:
  - Every cut in contracted graph is a cut in $G$, so every supernode has degree at least $k$.

  $\Pr[\text{mistake in } j + 1|\text{success so far}] \leq \frac{k}{\frac{1}{2}k(n-j)} = \frac{2}{n-j}$

Contraction Algorithm

**Theorem.** Alg finds global min cut with probability at least $1/\binom{n}{2}$.

**Corollary.** If we run $\binom{n}{2}$ in $n$ times, alg succeeds with probability at least $1 - 1/n$.

**Proof.**

$\Pr[\text{Fail all } t\ times] \leq \left(1 - \frac{1}{\binom{n}{2}}\right)^t$

If $t = c\binom{n}{2}$ this is at most $e^{-c}$. 

Global Min Cuts Takeaways

- Simple randomized algorithm works pretty well.
- Technical Tools
  - Chain Rule
  - Some calculus