Randomized and Approximation Algorithms

Today

Randomized and Approximation Algorithms

- Minimum Cuts
- Median Finding
- Vertex Cover

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?

Minimum Cuts

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

$$\text{cut}(A) = |\{(u, v) \in E, u \in A, v \notin A\}|$$

- 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 $\Rightarrow O(mn^2)$ 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**

\[ S(v) = \{v\} \text{ for all } v \in V. \]

\[ \textbf{while } |V| > 2 \textbf{ do} \]

\[ \text{Pick edge } (u,v) \in E \text{ uniformly at random.} \]

\[ \text{Contract edge } (u,v) \text{ to get } G' \text{ with new node } w. \]

\[ \text{Set } S(w) \leftarrow S(u) \cup S(v). \]

\[ \text{Update } G \leftarrow G'. \]

\[ \textbf{end while} \]

\[ \text{Return } S(v) \text{ for } v \in V. \]

---

**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 \(u \in A, v \in B\).
  
  \[ \Pr[\text{mistake in round 1}] = \Pr[\text{select } u \in A, v \in B] = \frac{k}{\# \text{ of edges}} \]

- \# of edges \(\geq \frac{1}{2}kn\) 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 \(j\)

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

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

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

  \[ = \frac{2}{n(3-1)} \]

---

**Global Min Cuts Takeaways**

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

---

**Contraction Algorithm**

<table>
<thead>
<tr>
<th>(S(v) = {v} \text{ for all } v \in V.)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>while</strong> (</td>
</tr>
<tr>
<td>[ \text{Pick edge } (u,v) \in E \text{ uniformly at random.} ]</td>
</tr>
<tr>
<td>[ \text{Contract edge } (u,v) \text{ to get } G' \text{ with new node } w. ]</td>
</tr>
<tr>
<td>[ \text{Set } S(w) \leftarrow S(u) \cup S(v). ]</td>
</tr>
<tr>
<td>[ \text{Update } G \leftarrow G'. ]</td>
</tr>
<tr>
<td><strong>end while</strong></td>
</tr>
<tr>
<td><strong>Return</strong> (S(v) \text{ for } v \in V.)</td>
</tr>
</tbody>
</table>

---

**Contraction Algorithm Analysis**

<table>
<thead>
<tr>
<th>[ \Pr[\text{mistake in round 1}] \leq \frac{k}{2kn} = \frac{1}{n} ]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Consider round</strong> (j + 1):</td>
</tr>
<tr>
<td>▸ <strong>Every cut in contracted graph is a cut in</strong> (G), so every supernode has degree at least (k).</td>
</tr>
<tr>
<td>[ \Pr[\text{mistake in } j + 1</td>
</tr>
</tbody>
</table>

---

**Final steps**

<table>
<thead>
<tr>
<th>▸ <strong>Let</strong> (E_j) <strong>be the event that</strong> ((A,B)) <strong>is not contracted in round</strong> (j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \Pr[E_j</td>
</tr>
</tbody>
</table>

---

**Global Min Cuts Takeaways**

<table>
<thead>
<tr>
<th>▸ <strong>Simple randomized algorithm works pretty well.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>▸ <strong>Technical Tools</strong></td>
</tr>
<tr>
<td>▸ Chain Rule</td>
</tr>
<tr>
<td>▸ Some calculus</td>
</tr>
</tbody>
</table>
Median Find

**Problem.** Given a set of numbers $S = \{a_1, \ldots, a_n\}$ the median is the number in the middle if the numbers were sorted.

- If $n$ is odd then $k$th smallest element where $k = (n + 1)/2$.
- If $n$ is even then $k$th smallest element where $k = n/2$.

**Deterministic algorithm?**

- Sort numbers, take $k$th smallest.
- $\mathcal{O}(n \log n)$.

**How to choose splitter?**

- We want recursive calls to work on much smaller sets.
- Best case, splitter is the median:
  $$T(n) \leq T(n/2) + cn = O(n)$$ runtime
- Worst case, splitter is largest element:
  $$T(n) \leq T(n - 1) + cn = O(n^2)$$ runtime
- Middle case, splitter separates $\epsilon n$ elements
  $$T(n) \leq T((1 - \epsilon)n) + cn$$
  $$T(n) \leq cn \left[ 1 + (1 - \epsilon) + (1 - \epsilon)^2 + \ldots \right] \leq \frac{cn}{\epsilon}$$

How can we stay close to the best case?

Divide and Conquer Algorithm

**Choose splitter (or pivot) $a_i \in S$**

**Form sets** $S^- = \{a_j : a_j < a_i\}, S^+ = \{a_j : a_j > a_i\}$.

- If:
  - $|S^-| = k - 1$: $a_i$ is the target.
  - $|S^-| \geq k$: recurse on $(S^-, k)$.
  - $|S^-| < k - 1$, recurse on $(S^+, k - (|S^-| + 1))$.

Looks kind of like quicksort...

Fact. Algorithm is correct.

Pseudocode

**SELECT(S,k):**

Choose splitter $a_i \in S$.

for each $a_j \in S$ do
  Put $a_j \in S^-$ if $a_j < a_i$.
  Put $a_j \in S^+$ if $a_j > a_i$.
end for

If $|S^-| = k - 1$, return $a_i$.

If $|S^-| \geq k$, return $\text{SELECT}(S^-, k)$.

Else, return $\text{SELECT}(S^+, k - (|S^-| + 1))$.

Looks kind of like quicksort...

Fact. Algorithm is correct.

**More generally**

**Problem.** Given a set of numbers $S = \{a_1, \ldots, a_n\}$ and number $k$, return $k$th smallest number. (Assume no duplicates)

**Special cases:**

- $k = 1$: minimum element $O(n)$
- $k = n$: maximum element $O(n)$.

Why is it $O(n \log n)$ for $k = n/2$?

Randomized Splitters

**Idea.** Choose splitter uniformly at random.

**Analysis.** Phase $j$ when $n(3/4)^j + 1 \leq |S| \leq n(3/4)^j$.

- **Claim.** Expect to stay in phase $j$ for two rounds.
  - Call splitter *central* if separates $1/4$ fraction of elements.
  - $\Pr[\text{central splitter}] = 1/2$.
  - If $X$ is number of attempts until central splitter,
    $$\mathbb{E}[X] = \sum_{j=1}^{\infty} j \Pr[X = j] = \sum_{j=1}^{\infty} j p(1 - p)^{j-1} = \frac{p}{1 - p} \sum_{j=1}^{\infty} j(1 - p)^j = \frac{p}{1 - p} \frac{(1 - p)^2}{p^2} = \frac{1}{p}$$
Analysis

- Let \( Y \) be a r.v. equal to number of steps of the algorithm
  - \( Y = Y_0 + Y_1 + Y_2 + \ldots \) where \( Y_j \) is steps in phase \( j \)
  - One iteration in phase \( j \) takes \( cn(3/4)^j \) steps.
  - \( E[Y_j] \leq 2cn(3/4)^j \) since expect two iterations.

\[
E[Y] = \sum_j E[Y_j] \leq \sum_j 2cn(3/4)^j = 2cn \sum_j (3/4)^j \leq 8cn
\]

Theorem

Expected running time of \( \text{Select}(n,k) \) is \( O(n) \).

Applications

- Randomized median find in expected linear time

Quicksort (Sketch)

- Choose pivot at random. Form \( S^-, S^+ \).
- Recursively sort both.
- Concatenate together.

Theorem. Quicksort has expected \( O(n \log n) \) time.

Approximation Algorithms

- We’ve seen important problems that are NP-complete. For these problems, should we just give up? No.
- Perhaps we can approximate them. For example, for a minimization problem can we design an algorithm such that whenever we run the algorithm we can guarantee that

\[
\frac{\text{value of our solution}}{\text{value of optimum solution}} \leq \alpha
\]

for some value of \( \alpha \geq 1 \). Such an algorithm is called an \( \alpha \)-approximation algorithm.

Vertex Cover

- Input. An undirected graph \( G = (V,E) \).
- Goal. Find the smallest subset of nodes \( S \subseteq V \) such that for every edge \( e \in E \), at least one of the end points of \( e \) is in \( S \).

Algorithm

- \( S \leftarrow \emptyset \)
- While the graph \( G \) has any edges:
  - Pick an edge \( (u,v) \)
  - Add \( u \) and \( v \) to \( S \)
  - Remove nodes \( u \) and \( v \) from \( G \) along with all incident edges
- Return \( S \)

Analysis

- Let \( M = \{e_1, \ldots, e_k\} \) be the edges picked by the algorithm and note that \( |S| = 2k \).
- Lemma: The minimum vertex cover has size at least \( k \).
- Proof: Since the endpoints of \( e_1, \ldots, e_k \) are all distinct, it takes at least \( k \) nodes to cover the edges in \( M \).
- Lemma: The nodes in \( S \) are a vertex cover.
- Proof: Consider any edge \( e = (u,v) \in E \). At the end of the algorithm, \( e \) isn’t in the graph. The only way \( e \) could have been removed is if \( u \) or \( v \) was added to \( S \). Hence \( S \) is a vertex cover.
- Therefore the algorithm achieves an approximation ratio of:

\[
\frac{\text{value of our solution}}{\text{value of optimum solution}} \leq \frac{2k}{k} = 2
\]
A randomized approximation algorithm!

- $S \leftarrow \emptyset$
- For each $(u, v) \in E$:
  - If neither $u$ nor $v$ are in $S$
  - Randomly select one, add to $S$
- Return $S$

Analysis

- Let $OPT$ denote the optimal vertex cover.
- At each round, we maintain
  \[ E|S \cap OPT| \geq E|S \setminus OPT| \]
- Since when we add an element, $OPT$ must as well, and we agree with probability $1/2$.
- Implies $E|S| \leq 2|OPT|$