Lecture 9
• Greedy Algorithms for Graph Problems
Hw 3 and Stirling’s formula

- Richard Burhans pointed out 2 independent typos
  - Problem: they were mutually consistent!
Hw 3 and Stirling’s formula

- Richard Burhans pointed out 2 independent typos
  - Problem: they were mutually consistent!
- HW 3: find positive $a$ such that $\Pr(\ldots) = \Theta(n^a)$
Hw 3 and Stirling’s formula

• Richard Burhans pointed out 2 independent typos
  – Problem: they were mutually consistent!
• HW 3: find positive $a$ such that $\Pr(\ldots) = \Theta(n^{-a})$
  – (Probabilities are at most 1)
Hw 3 and Stirling’s formula

• Richard Burhans pointed out 2 independent typos
  – Problem: they were mutually consistent!
• HW 3: find positive $a$ such that $Pr(\ldots) = \Theta(n^{-a})$
  – (Probabilities are at most 1)

• Lecture 3: Stirling’s formula

\[ n! = \left(\sqrt{2\pi n}\right) \left(\frac{n}{e}\right)^n \left(1 + o(1)\right) \]
Shortest Path Problem

• **Input:**
  – Directed graph $G = (V, E)$.
  – Source node $s$, destination node $t$.
  – for each edge $e$, length $\ell(e) = \text{length of } e$.

• **Find:** shortest directed path from $s$ to $t$. 

Length of path $(s, 2, 3, 5, t)$ is $9 + 23 + 2 + 16 = 50$. 

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Dijkstra's Algorithm (Greedy)

- Maintain a set of **explored nodes** \( S \) whose shortest path distance \( d(u) \) from \( s \) to \( u \) is known.
- Initialize \( S = \{ s \} \), \( d(s) = 0 \).
- Repeatedly choose unexplored node \( v \) which minimizes
  \[
  \pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell(e),
  \]
- add \( v \) to \( S \), and set \( d(v) = \pi(v) \).
Dijkstra's Algorithm (Greedy)

- Maintain a set of **explored nodes** $S$ whose shortest path distance $d(u)$ from $s$ to $u$ is known.
- Initialize $S = \{ s \}$, $d(s) = 0$.
- Repeatedly choose unexplored node $v$ which minimizes $\pi(v) = \min_{e = (u,v): u \in S} d(u) + \ell(e)$, 
  - add $v$ to $S$, and set $d(v) = \pi(v)$.

BFS with weighted edges

$A. \text{Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne}$
Proof of Correctness
(Greedy Stays Ahead)

Invariant. For each node $u \in S$, $d(u)$ is the length of the shortest path from $s$ to $u$.

Proof: (by induction on $|S|$)

- **Base case:** $|S| = 1$ is trivial.
- **Inductive hypothesis:** Assume for $|S| = k \geq 1$.
  
  - Let $v$ be next node added to $S$, and let $(u,v)$ be the chosen edge.
  - The shortest $s$-$u$ path plus $(u,v)$ is an $s$-$v$ path of length $\pi(v)$.
  - Consider any $s$-$v$ path $P$. We'll see that it's no shorter than $\pi(v)$.
  - Let $(x,y)$ be the first edge in $P$ that leaves $S$, and let $P'$ be the subpath to $x$.
  
  - $P + (x,y)$ has length $\leq d(x) + \ell(x,y) \leq \pi(y) \leq \pi(v)$
Implementation

• For unexplored nodes, maintain $\pi(v) = \min_{e=(u,v): u \in S} d(u) + \ell(e)$.
  – Next node to explore = node with minimum $\pi(v)$.
  – When exploring $v$, for each edge $e = (v,w)$, update $\pi(w) = \min \{ \pi(w), \pi(v) + \ell(e) \}$.

Efficient implementation: Maintain a priority queue of unexplored nodes, prioritized by $\pi(v)$. 
Implementation of Dijkstra\((G, \ell)\)

\[d[s] \leftarrow 0\]

\[
\text{for each } v \in V - \{s\} \text{ do } d[v] \leftarrow \infty; \pi[v] \leftarrow \infty
\]

\[S \leftarrow \emptyset\]

\[Q \leftarrow V\]

\(Q\) is a priority queue maintaining \(V - S\), keyed on \(\pi[v]\)

\[
\text{while } Q \neq \emptyset \text{ do } u \leftarrow \text{Extract-Min}(Q)
\]

\[
S \leftarrow S \cup \{u\}; \quad d[u] \leftarrow \pi[u]
\]

\[
\text{for each } v \in \text{Adjacency-list}[u] \text{ do if } \pi[v] > \pi[u] + \ell(u, v) \text{ then } \pi[v] \leftarrow d[u] + \ell(u, v)
\]

Implicit \text{DECREASE-KEY}
Demo of Dijkstra’s Algorithm

Graph with nonnegative edge lengths:
Demo of Dijkstra’s Algorithm

Initialize:

\[ Q: \begin{array}{ccccc}
A & B & C & D & E \\
0 & \infty & \infty & \infty & \infty \\
\end{array} \]

\[ S: \{\} \]
Demo of Dijkstra’s Algorithm

“$A$” $\leftarrow$ **Extract-Min**($Q$):

$Q$: $\begin{array}{ccccc}
A & B & C & D & E \\
0 & \infty & \infty & \infty & \infty
\end{array}$

$S$: $\{ A \}$

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Demo of Dijkstra’s Algorithm

Explore edges leaving $A$:

$Q$: $\begin{align*}
A & \quad B & \quad C & \quad D & \quad E \\
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & \infty & \infty
\end{align*}$

$S$: \{ $A$ \}
Demo of Dijkstra’s Algorithm

“C” ← \textbf{\textsc{Extract-Min}}(Q):

\[ Q: \begin{array}{cccccc}
A & B & C & D & E \\
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & \infty & \infty \\
\end{array} \]

\[ S: \{ A, C \} \]
Demo of Dijkstra’s Algorithm

Explore edges leaving $C$:

$Q$:  

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>5</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
</tbody>
</table>

$S$: $\{A, C\}$
Demo of Dijkstra’s Algorithm

“E” ← \textbf{Extract-Min}(Q):

\[ Q: \begin{array}{ccccc} 
A & B & C & D & E \\
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & \infty & \infty \\
7 & & & & 5 \\
\end{array} \]

\[ S: \{A, C, E\} \]
Demo of Dijkstra’s Algorithm

Explore edges leaving $E$:

$Q:$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>5</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

$S$: \{ A, C, E \}
Demo of Dijkstra’s Algorithm

“B” ← \text{\textsc{Extract-Min}}(Q):

\begin{tabular}{cccc}
Q: & A & B & C & D & E \\
\hline
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & \infty & \infty \\
7 & 7 & 11 & 11 & 5 \\
\end{tabular}

S: \{ A, C, E, B \}
Demo of Dijkstra’s Algorithm

Explore edges leaving B:

\[ A \quad B \quad C \quad D \quad E \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>S</td>
<td></td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

\[ S: \{ A, C, E, B \} \]
Demo of Dijkstra’s Algorithm

"D" $\leftarrow$ **Extract-Min**(Q):

$Q$: \begin{align*}
A & \quad B & \quad C & \quad D & \quad E \\
0 & \infty & \infty & \infty & \infty \\
10 & 3 & \infty & 11 & \infty \\
7 & 7 & 11 & 9 & \\
\end{align*}

$S$: \{ A, C, E, B, D \}
Analysis of Dijkstra

while $Q \neq \emptyset$
    do $u \leftarrow \text{Extract-Min}(Q)$
        $S \leftarrow S \cup \{u\}$
        for each $v \in \text{Adj}[u]$
            do if $d[v] > d[u] + w(u, v)$
                then $d[v] \leftarrow d[u] + w(u, v)$

Handshaking Lemma $\Rightarrow \leq m$ implicit $\text{DECREASE-KEY}$’s.

<table>
<thead>
<tr>
<th>PQ Operation</th>
<th>Dijkstra</th>
<th>Array</th>
<th>Binary heap</th>
<th>d-way Heap</th>
<th>Fib heap $^\dagger$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtractMin</td>
<td>$n$</td>
<td>$n$</td>
<td>$\log n$</td>
<td>HW3</td>
<td>$\log n$</td>
</tr>
<tr>
<td>DecreaseKey</td>
<td>$m$</td>
<td>1</td>
<td>$\log n$</td>
<td>HW3</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>$n^2$</td>
<td>$\text{n log } n$</td>
<td>$m \log \frac{m}{n}$</td>
<td>$n$</td>
<td></td>
</tr>
</tbody>
</table>

$^\dagger$ Individual ops are amortized bounds

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Review Question

• Is Dijsktra’s algorithm correct with negative edge weights?
Minimum spanning tree (MST)

Input: A connected undirected graph $G = (V, E)$ with weight function $w: E \rightarrow \mathbb{R}$.
• For now, assume all edge weights are distinct.

Output: A spanning tree $T$ — a tree that connects all vertices — of minimum weight:

$$w(T) = \sum_{(u,v) \in T} w(u,v).$$
Example of MST
Example of MST

A. Smith; based on slides by E. Demaine, C. Leiserson, S. Raskhodnikova, K. Wayne
Greedy Algorithms for MST

• **Kruskal's**: Start with $T = \emptyset$. Consider edges in ascending order of weights. Insert edge $e$ in $T$ unless doing so would create a cycle.

• **Reverse-Delete**: Start with $T = E$. Consider edges in descending order of weights. Delete edge $e$ from $T$ unless doing so would disconnect $T$.

• **Prim's**: Start with some root node $s$. Grow a tree $T$ from $s$ outward. At each step, add to $T$ the cheapest edge $e$ with exactly one endpoint in $T$. 
Cycles and Cuts

• **Cycle:** Set of edges the form \((a,b),(b,c),(c,d),\ldots,(y,z),(z,a)\).

• **Cut:** a subset of nodes \(S\). The corresponding **cutset** \(D\) is the subset of edges with exactly one endpoint in \(S\).

\[\text{Cycle } C = (1,2),(2,3),(3,4),(4,5),(5,6),(6,1)\]

\[\text{Cut } S = \{4,5,8\}\]

\[\text{Cutset } D = (5,6), (5,7), (3,4), (3,5), (7,8)\]
Cycle-Cut Intersection

• **Claim.** A cycle and a cutset intersect in an even number of edges.

• **Proof:** A cycle has to leave and enter the cut the same number of times.

![Diagram showing cycle C intersecting with cut sets S and V - S]
**Cut property.** Let $S$ be a subset of nodes. Let $e$ be the min weight edge with exactly one endpoint in $S$. Then the MST contains $e$.

**Cycle property.** Let $C$ be a cycle, and let $f$ be the max weight edge in $C$. Then the MST does not contain $f$. 
Proof of Cut Property

**Cut property:** Let $S$ be a subset of nodes. Let $e$ be the min weight edge with exactly one endpoint in $S$. Then the MST $T^*$ contains $e$.

**Proof:** (exchange argument)

- Suppose $e$ does not belong to $T^*$.
- Adding $e$ to $T^*$ creates a cycle $C$ in $T^*$.
- Edge $e$ is both in the cycle $C$ and in the cutset $D$ corresponding to $S \Rightarrow$ there exists another edge, say $f$, that is in both $C$ and $D$.
- $T' = T^* \cup \{ e \} - \{ f \}$ is also a spanning tree.
- Since $c_e < c_f$, cost($T'$) < cost($T^*$). Contradiction. \qed
Proof of Cycle Property

Cycle property: Let $C$ be a cycle in $G$. Let $f$ be the max weight edge in $C$. Then the MST $T^*$ does not contain $f$.

• **Proof: (exchange argument)**
  – Suppose $f$ belongs to $T^*$.
  – Deleting $f$ from $T^*$ creates a cut $S$ in $T^*$.
  – Edge $f$ is both in the cycle $C$ and in the cutset $D$ corresponding to $S \Rightarrow$ there exists another edge, say $e$, that is in both $C$ and $D$.
  – $T' = T^* \cup \{ e \} - \{ f \}$ is also a spanning tree.
  – Since $c_e < c_f$, $\text{cost}(T') < \text{cost}(T*)$. Contradiction. ▪
Review Questions

Let $G$ be a connected undirected graph with distinct edge weights. Answer true or false:

• Let $e$ be the cheapest edge in $G$. Some MST of $G$ contains $e$?

• Let $e$ be the most expensive edge in $G$. No MST of $G$ contains $e$?
Review Questions

Let $G$ be a connected undirected graph with distinct edge weights. Answer true or false:

- Let $e$ be the cheapest edge in $G$. Some MST of $G$ contains $e$?
  (Answer: Yes, by the Cut Property)

- Let $e$ be the most expensive edge in $G$. No MST of $G$ contains $e$?
  (Answer: False. Counterexample: if $G$ is a tree, all its edges are in the MST)
Prim's Algorithm: Correctness

• Prim's algorithm. [Jarník 1930, Prim 1959]
  – Apply cut property to S.
  – When edge weights are distinct, every edge that is added **must** be in the MST.
  – Thus, Prim’s alg. outputs the MST.