Menu

- Last week: Bellman-Ford
  - $O(VE)$ time
  - general weights

- Today: Dijkstra
  - $O((V+E)\log V)$ time
  - non-negative weights
Single source shortest path problem

• Problem: Given a digraph $G = (V, E)$ with non-negative edge-weight function $w$, and a node $s$, find $\delta(s, v)^*$ for all $v$ in $V$
• Want a fast algorithm…
• Question: what if all edge weights are equal to 1 ?

*Paths can be found as well
BFS

Running time: $O(V+E)$
Weighted graphs

Question II: what if all edge weights are integers in the range $1 \ldots W$?
Algorithm:
• Create an unweighted graph by splitting each edge with weight $w$ into $w$ pieces
• Run BFS

Running time: $O(V+WE)$
Greedy approach

**Idea:** Greedy.

1. Maintain a set $S$ of vertices whose shortest-path distances from $s$ are known.
2. At each step add to $S$ the vertex $v \in V - S$ whose distance estimate from $s$ is minimal.
3. Update the distance estimates of vertices adjacent to $v$. 
Dijkstra’s algorithm

\[d[s] \leftarrow 0\]

for each \(v \in V - \{s\}\) do  
\[d[v] \leftarrow \infty\]

\(S \leftarrow \emptyset\)

\(Q \leftarrow V\)  \(\triangleright Q\) is a priority queue maintaining \(V - S\)

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  
\[d[v] \leftarrow d[u] + w(u, v)\]

relaxation step

Implicit \textsc{Decrease-Key}
Dijkstra: Example

initialization

(0,*)
Dijkstra: Example

1st iteration
Dijkstra: Example

2nd iteration

(3, a)⁺

(0, *)⁺

(5, a)⁺

(8, a)⁺

(10, b)

(∝, -)

(∝, -)

(∝, -)

(∝, -)

(∝, -)

(∝, -)

(∝, -)

(∝, -)

Introduction to Algorithms

4/5/11  11
Dijkstra: Example

3rd iteration
Dijkstra: Example

4th iteration
Dijkstra: Example

6th iteration

Introduction to Algorithms
Dijkstra: Example

7th iteration

- (0,*)
- (3,a)
- (7,d)
- (9,d)
- (13,g)
- (14,i)
- (16,f)
- (19,i)
- (5,a)
- (10,b)
- (11,b)
- (12,b)
Dijkstra: Example

8th iteration
Dijkstra: Example

9th iteration

Shortest-path tree
Correctness — Part I

**Lemma.** Initializing $d[s] \leftarrow 0$ and $d[v] \leftarrow \infty$ for all $v \in V - \{s\}$ establishes $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps.

**Proof.** Recall relaxation step:

$$\text{if } d[v] > d[u] + w(u, v) \text{ set } d[v] \leftarrow d[u] + w(u, v)$$
Correctness — Part II

**Theorem.** Dijkstra’s algorithm terminates with $d[v] = \delta(s, v)$ for all $v \in V$.

**Proof.**
- It suffices to show that $d[v] = \delta(s, v)$ for every $v \in V$ when $v$ is added to $S$.
- Suppose $u$ is the first vertex added to $S$ for which $d[u] \neq \delta(s, u)$. Let $y$ be the first vertex in $V - S$ along a shortest path from $s$ to $u$, and let $x$ be its predecessor:

\[ S, \text{ just before adding } u. \]
Correctness — Part II (continued)

- Since $u$ is the first vertex violating the claimed invariant, we have $d[x] = \delta(s, x)$
- Since subpaths of shortest paths are shortest paths, it follows that $d[y]$ was set to $\delta(s, x) + w(x, y) = \delta(s, y)$ just after $x$ was added to $S$
- Consequently, we have $d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]$
- But, $d[y] \geq d[u]$ since the algorithm chose $u$ first
- Hence $d[y] = \delta(s, y) = \delta(s, u) = d[u]$ - contradiction
Analysis of Dijkstra

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

\text{DECREASE-KEY}

Time = \Theta(V) \cdot T_{\text{EXTRACT-MIN}} + \Theta(E) \cdot T_{\text{DECREASE-KEY}}
### Analysis of Dijkstra (continued)

Time = $\Theta(V) \cdot T_{\text{Extract-Min}} + \Theta(E) \cdot T_{\text{Decrease-Key}}$

<table>
<thead>
<tr>
<th>Q</th>
<th>$T_{\text{Extract-Min}}$</th>
<th>$T_{\text{Decrease-Key}}$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>$O(V)$</td>
<td>$O(1)$</td>
<td>$O(V^2)$</td>
</tr>
<tr>
<td>binary heap</td>
<td>$O(lg V)$</td>
<td>$O(lg V)$</td>
<td>$O(E \ lg V)$</td>
</tr>
<tr>
<td>Fibonacci heap</td>
<td>$O(lg V)$</td>
<td>$O(1)$</td>
<td>$O(E + V \ lg V)$</td>
</tr>
</tbody>
</table>

$Q$ denotes the total number of vertices, $T_{\text{Extract-Min}}$ and $T_{\text{Decrease-Key}}$ are the times for operations on a given data structure, and the Total column gives the overall time complexity.
Tuesday: generic algorithm

\[ d[s] \leftarrow 0 \]
\[
\text{for each } v \in V - \{s\} \quad \left\{ \begin{array}{c}
\text{do } d[v] \leftarrow \infty \\
\text{initialization}
\end{array} \right. \\
\text{while there is an edge } (u, v) \in E \text{ s. t. } \]
\[
d[v] > d[u] + w(u, v) \quad \left\{ \begin{array}{c}
\text{do select one such edge “somehow”} \\
\text{set } d[v] \leftarrow d[u] + w(u, v) \\
\text{relaxation step}
\end{array} \right. \\
\text{endwhile}
\]

How to do it in \( O( (V+E)\log V ) \) time?