Dijkstra’s algorithm
Announcements

• Topics
  – Dijkstra’s Algorithm (Section 4.4)
  – Next Week: Minimum Spanning Trees

• Reading
  – 4.4, 4.5, 4.7, 4.9

• Midterm: Monday, October 30, in class
Single Source Shortest Path Problem

• Given a graph and a start vertex $s$
  – Determine distance of every vertex from $s$
  – Identify shortest paths to each vertex
    • Express concisely as a “shortest paths tree”
    • Each vertex has a pointer to a predecessor on shortest path
Construct Shortest Path Tree from s
Warmup

• If $P$ is a shortest path from $s$ to $v$, and if $t$ is on the path $P$, the segment from $s$ to $t$ is a shortest path between $s$ and $t$

• WHY?
Dijkstra’s Algorithm

$S = \{ \}; \quad d[s] = 0; \quad d[v] = \text{infinity for } v \neq s$

While $S \neq V$

Choose $v$ in $V-S$ with minimum $d[v]$

Add $v$ to $S$

For each $w$ in the neighborhood of $v$

$$d[w] = \min(d[w], d[v] + c(v, w))$$

Assume all edges have non-negative cost
Simulate Dijkstra’s algorithm (starting from $s$) on the graph

<table>
<thead>
<tr>
<th>Round</th>
<th>Vertex Added</th>
<th>s</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

10/20/2023
Who was Dijkstra?

• What were his major contributions?
Edsger Wybe Dijkstra was one of the most influential members of computing science's founding generation. Among the domains in which his scientific contributions are fundamental are

- algorithm design
- programming languages
- program design
- operating systems
- distributed processing
- formal specification and verification
- design of mathematical arguments
Dijkstra’s Algorithm as a greedy algorithm

- Elements committed to the solution by order of minimum distance
Correctness Proof

• Elements in S have the correct label
• Key to proof: when v is added to S, it has the correct distance label.
Proof

• Let v be a vertex in V-S with minimum \( d[v] \)
• Let \( P_v \) be a path of length \( d[v] \), with an edge \( (u,v) \)
• Let \( P \) be some other path to v. Suppose \( P \) first leaves S on the edge \( (x, y) \)
  
  \[
  P = P_{sx} + c(x,y) + P_{yv}
  \]
  
  \[
  \text{Len}(P_{sx}) + c(x,y) \geq d[y]
  \]
  
  \[
  \text{Len}(P_{yv}) \geq 0
  \]
  
  \[
  \text{Len}(P) \geq d[y] + 0 \geq d[v]
  \]
Negative Cost Edges

• Draw a small example a negative cost edge and show that Dijkstra’s algorithm fails on this example
Dijkstra Implementation

S = { };  \ d[s] = 0; \quad \text{d[v] = infinity for v != s}

While S != V

Choose v in V-S with minimum d[v]

Add v to S

For each w in the neighborhood of v

\[ d[w] = \min(d[w], d[v] + c(v, w)) \]

• Basic implementation requires Heap for tracking the distance values
• Run time O(m log n)
O(n^2) Implementation for Dense Graphs

\[
\begin{align*}
\text{FOR } i & := 1 \text{ TO } n \\
\text{ \quad } d[i] & := \text{Infinity}; \ \text{visited}[i] := \text{FALSE;} \\
\text{d[s]} & := 0;
\end{align*}
\]

\[
\begin{align*}
\text{FOR } i & := 1 \text{ TO } n \\
\text{ \quad } v & := -1; \ \text{dMin} := \text{Infinity;} \\
\text{FOR } j & := 1 \text{ TO } n \\
\text{ \quad } \text{IF visited}[j] = \text{FALSE AND } d[j] < \text{dMin} \\
\text{ \quad \quad } v & := j; \ \text{dMin} := d[j]; \\
\text{IF } v & = -1 \\
\text{ \quad \quad } \text{RETURN;} \\
\text{visited}[v] & := \text{TRUE;}
\end{align*}
\]

\[
\begin{align*}
\text{FOR } j & := 1 \text{ TO } n \\
\text{ \quad } \text{IF } d[v] + \text{len}[v, j] < d[j] \\
\text{ \quad \quad } d[j] & := d[v] + \text{len}[v, j]; \\
\text{prev}[j] & := v;
\end{align*}
\]
Bottleneck Shortest Path

• Define the bottleneck distance for a path to be the maximum cost edge along the path
Compute the bottleneck shortest paths
How do you adapt Dijkstra’s algorithm to handle bottleneck distances

• Does the correctness proof still apply?