April 29 – Floyd-Warshall’s Algorithm
Wade Fagen-Ulmschneider, Craig Zilles
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?

Shortest Path (A \(\rightarrow\) E): \(A \rightarrow F \rightarrow E \rightarrow (C \rightarrow H \rightarrow G \rightarrow E)^*\)

Length: 12 \hspace{1cm} Length: -5 (repeatable)
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle negative weight edges, without a negative weight cycle?
Dijkstra’s Algorithm (SSSP)

Q: How does Dijkstra handle a single heavy-weight path vs. many light-weight paths?
Dijkstra’s Algorithm (SSSP)

What is Dijkstra’s running time?

```
DijkstraSSSP(G, s):
  foreach (Vertex v : G):
    d[v] = +inf
    p[v] = NULL
  d[s] = 0
  PriorityQueue Q // min distance, defined by d[v]
  Q.buildHeap(G.vertices())
  Graph T       // "labeled set"

  repeat n times:
    Vertex u = Q.removeMin()
    T.add(u)
    foreach (Vertex v : neighbors of u not in T):
      if cost(u, v) + d[u] < d[v]:
        d[v] = cost(u, v) + d[u]
        p[v] = m
  return T
```
Floyd-Warshall Algorithm

Floyd-Warshall’s Algorithm is an alternative to Dijkstra in the presence of **negative-weight edges** (**not** negative weight cycles).

<table>
<thead>
<tr>
<th>FloydWarshall(G):</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
</tbody>
</table>
Floyd-Warshall Algorithm

FloydWarshall(G):

6   Let d be a adj. matrix initialized to +inf
7   foreach (Vertex v : G):
8       d[v][v] = 0
9   foreach (Edge (u, v) : G):
10      d[u][v] = cost(u, v)
11
12   foreach (Vertex u : G):
13      foreach (Vertex v : G):
14         foreach (Vertex w : G):
15            if d[u, v] > d[u, w] + d[w, v]:
16               d[u, v] = d[u, w] + d[w, v]
Floyd-Warshall Algorithm

```
12    foreach (Vertex u : G):
13        foreach (Vertex v : G):
14            foreach (Vertex k : G):
15                if d[u, v] > d[u, k] + d[k, v]:
16                    d[u, v] = d[u, k] + d[k, v]
```

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-1</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

```
```
Floyd-Warshall Algorithm

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-1</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

Let us consider k=A:

- B → C: 4 vs. B → A → C: +∞
- B → D: 3 vs. B → A → D: +∞
- C → B: +∞ vs. C → A → B: +∞
- C → D: -2 vs. C → A → D: +∞
- D → B: +∞ vs. D → B → B: +∞
- D → C: +∞ vs. D → B → C: +∞
Floyd-Warshall Algorithm

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>-1</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>0</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>1</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

foreach (Vertex u : G):
foreach (Vertex v : G):
foreach (Vertex k : G):
    if d[u, v] > d[u, k] + d[k, v]:
        d[u, v] = d[u, k] + d[k, v]
Let us consider \( k = B \):

\[
\begin{align*}
12 & \quad \text{foreach (Vertex } u : G) : \\
13 & \quad \text{foreach (Vertex } v : G) : \\
14 & \quad \text{foreach (Vertex } k : G) : \\
15 & \quad \text{if } d[u, v] > d[u, k] + d[k, v] : \\
16 & \quad d[u, v] = d[u, k] + d[k, v]
\end{align*}
\]
Floyd-Warshall Algorithm Intuition

Consider a graph G with vertices V numbered 1 through N.

Consider the function shortestPath(i, j, k) that returns the shortest possible path from i to j using only vertices from the set \{1,2, ... ,k\} as intermediate vertices.

Clearly, shortestPath(i, j, N) returns _____________________________
Floyd-Warshall Algorithm Intuition

For each pair of vertices, the shortestPath(i, j, k) could be either

(1) a path that *doesn't* go through k (only uses vertices in the set {1, ..., k-1}.)

(2) a path that *does* go through k (from i to k and then from k to j, both only using intermediate vertices in {1, ..., k-1})
Floyd-Warshall Algorithm Intuition

If $w(i,j)$ is the weight of the edge between vertices $i$ and $j$, we can recursively define $\text{shortestPath}(i,j,k)$ as:

\[
\text{shortestPath}(i, j, 0) = \begin{cases} 
// base case \\
\end{cases}
\]

\[
\text{shortestPath}(i, j, k) = \min ( \begin{cases} 
// recursive \\
\end{cases} ) 
\]
Floyd-Warshall Algorithm Intuition

If \( w(i,j) \) is the weight of the edge between vertices \( i \) and \( j \), we can recursively define \( \text{shortestPath}(i,j,k) \) as:

\[
\text{shortestPath}(i, j, 0) = w(i, j) \quad // \text{base case}
\]

\[
\text{shortestPath}(i, j, k) = \min( \text{shortestPath}(i, j, k-1), \quad // \text{recursive}
\quad \text{shortestPath}(i, k, k-1) + \text{shortestPath}(k, j, k-1) )
\]
Floyd-Warshall Algorithm

Running Time?

<table>
<thead>
<tr>
<th>FloydWarshall(G):</th>
</tr>
</thead>
<tbody>
<tr>
<td>6  Let d be a adj. matrix initialized to +inf</td>
</tr>
<tr>
<td>7  foreach (Vertex v : G):</td>
</tr>
<tr>
<td>8   d[v][v] = 0</td>
</tr>
<tr>
<td>9  foreach (Edge (u, v) : G):</td>
</tr>
<tr>
<td>10  d[u][v] = cost(u, v)</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12  foreach (Vertex u : G):</td>
</tr>
<tr>
<td>13   foreach (Vertex v : G):</td>
</tr>
<tr>
<td>14     foreach (Vertex w : G):</td>
</tr>
<tr>
<td>15       if d[u, v] &gt; d[u, w] + d[w, v]:</td>
</tr>
<tr>
<td>16       d[u, v] = d[u, w] + d[w, v]</td>
</tr>
</tbody>
</table>
Final Exam Review Session

• Implementations
  • Edge List
  • Adjacency Matrix
  • Adjacency List
• Traversals
  • Breadth First
  • Depth First
• Minimum Spanning Tree
  • Kruskal’s Algorithm
  • Prim’s Algorithm
• Shortest Path
  • Dijkstra’s Algorithm
  • Floyd-Warshall’s Algorithm

...and this is just the beginning. The journey continues to CS 374!