April 27 – Graph Problems + End of Semester
Wade Fagen-Ulmschneider
Dijkstra’s Algorithm (SSSP)

DijkstraSSSP(G, s):
6     foreach (Vertex v : G):
7         d[v] = +inf
8         p[v] = NULL
9         d[s] = 0
10    PriorityQueue Q // min distance, defined by d[v]
11    Q.buildHeap(G.vertices())
12    Graph T         // "labeled set"
13
15    repeat n times:
16        Vertex u = Q.removeMin()
17        T.add(u)
18        foreach (Vertex v : neighbors of u not in T):
19            if cost(u, v) + d[u] < d[v]:
20                d[v] = cost(u, v) + d[u]
21                p[v] = m
Dijkstra’s Algorithm (SSSP)

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

Q: How does Dijkstra handle negative weight edges, without a negative weight cycle?
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] = u

    return T
```
Landmark Path Problem

Suppose you want to travel from A to G.  

Q1: What is the shortest path from A to G?
Landmark Path Problem

Suppose you want to travel from $A$ to $G$.

**Q2:** What is the fastest algorithm to use to find the shortest path?
Landmark Path Problem

In your journey between A and G, you also want to visit the landmark L.

Q3: What is the shortest path from A to G that visits L?
Landmark Path Problem

In your journey between A and G, you also want to visit the landmark L.

Q4: What is the fastest algorithm to find this path?

Q5: What are the specific call(s) to this algorithm?
End of Semester Logistics

CS 225 Final Exam

• The final exam begins on Thursday, May 3rd

• The final exam is a 3 hour CBTF exam, is a cumulative exam, and has the format of a combined theory + programming exam

• The last office hours is Wednesday, May 2nd

• We’ll use lecture on Wednesday, May 2nd as a final exam review!
End of Semester Logistics

“Pre-Final” Grade Dump

• I believe there’s only a few remaining issues left with grading; I’ll be starting to wrap these up myself over the weekend:
  • +EC from creative components
  • Working on recovering repos that were force deleted

• As soon as possible after MP7’s deadline, we’ll provide a “Pre-Final” grade in Compass that incorporates everything except the final exam into your CS 225 grade.
End of Semester Logistics

End of Semester Grade Review
• Excel sheet will be provided once final grades are posted.
• Must submit an Excel sheet for this review.
End of Semester Logistics

You’re Awesome -- +1 To Your Skills!
My Passion: Data Discovery

GPAs at Illinois:

Diversity at Illinois:

And others:

http://waf.cs.illinois.edu/discovery/
Floyd-Warshall Algorithm

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

```
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

<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>

A | B | C | D |
---|---|---|---|
A |   |   |   |
B |   |   |   |
C |   |   |   |
D |   |   |   |
Floyd-Warshall Algorithm

Initially:

```
A  B  C  D
A  0  -1
B  0  4  3
C  0  -2
D  2  0
```

```
foreach (Vertex u : G):
  foreach (Vertex v : G):
    foreach (Vertex w : G):
      if d[u, v] > d[u, w] + d[w, v]:
        d[u, v] = d[u, w] + d[w, v]
```
Floyd-Warshall Algorithm

Initially:

Let \( u = A; \) \( v \) and \( w \) explores for better paths:

```
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

Initially:

Let $u = A$; $v$ and $w$ explores for better paths:

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

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 w : G):
15               if d[u, v] > d[u, w] + d[w, v]:
16                   d[u, v] = d[u, w] + d[w, v]
```

Initially:

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

Let u = A; v and w explores for better paths:

- A → B: -1
- A → C: 2
- A → D: 1
- B → C: 4
- B → D: 3
- C → D: -2
- D → A: 2
Floyd-Warshall Algorithm

Initially:

Let $u = B$; $v$ and $w$ explores for better paths:

<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>2</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

```python
12. foreach (Vertex u : G):
13.     foreach (Vertex v : G):
14.         foreach (Vertex w : G):
15.             if $d[u, v] > d[u, w] + d[w, v]$
```
Floyd-Warshall Algorithm

Initially:

Let $u = B$; $v$ and $w$ explores for better paths:

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

12. foreach (Vertex $u : G$):
13.   foreach (Vertex $v : G$):
14.     foreach (Vertex $w : G$):
15.         if $d[u, v] > d[u, w] + d[w, v]$:

- $A$: source vertex
- $B$: destination vertex
- $C$: intermediate vertex
- $D$: another intermediate vertex