Dec. 11 – Floyd-Warshall’s Algorithm
Wade Fagen-Ulmschneider
## Reinforcement Learning

<table>
<thead>
<tr>
<th>Available Tokens</th>
<th>Learned Move</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Take 1 token → 9</td>
</tr>
<tr>
<td>9</td>
<td>Take 2 tokens → 7</td>
</tr>
<tr>
<td>8</td>
<td>Take 2 tokens → 6</td>
</tr>
<tr>
<td>7</td>
<td>Take 1 token → 6</td>
</tr>
<tr>
<td>6</td>
<td>Take 1 token → 5</td>
</tr>
<tr>
<td>5</td>
<td>Take 2 tokens → 3</td>
</tr>
<tr>
<td>4</td>
<td>Take 1 token → 3</td>
</tr>
<tr>
<td>3</td>
<td>Take 1 token → 2</td>
</tr>
<tr>
<td>2</td>
<td>Take 2 tokens → 0 (win)</td>
</tr>
<tr>
<td>1</td>
<td>Take 1 token → 0 (win)</td>
</tr>
</tbody>
</table>

![Graph showing the Reinforcement Learning process](image-url)
Last week, Google’s DeepMind AI team released a new research paper:

- Using reinforcement learning, an algorithm knowing only the rules of chess trained for 4 hours.
- After training, it destroyed the best chess program (Stockfish):

```
<table>
<thead>
<tr>
<th>Game</th>
<th>White</th>
<th>Black</th>
<th>Win</th>
<th>Draw</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chess</td>
<td>AlphaZero</td>
<td>Stockfish</td>
<td>25</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stockfish</td>
<td>3</td>
<td>47</td>
<td>0</td>
</tr>
<tr>
<td>Shogi</td>
<td>AlphaZero</td>
<td>Elmo</td>
<td>43</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AlphaZero</td>
<td>47</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Go</td>
<td>AlphaZero</td>
<td>AGO 3-day</td>
<td>31</td>
<td>–</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AlphaZero</td>
<td>29</td>
<td>–</td>
<td>21</td>
</tr>
</tbody>
</table>
```

Final Exam Information

Multiple Choice:
• 22 total multiple choice questions
  • 8 questions on graphs
  • 14 questions on pre-graph content
  • No questions specifically about C++, pointers, etc

Programming:
• One “easy” question
  • Heaps, hash tables, disjoint sets, tree encoding, etc. are fair game
• One “hard” question
  • Graph algorithm: be able to implement Prim, Kruskal, Dijkstra, BFS, DFS, etc
• We will likely not tell you which algorithm to use!
• We will post the .h files on Wednesday.
End of Semester Logistics

Regrades on Exams:
• Most of these have been posted.
• Any corrections needed, send Mattox an email, *not* Piazza.
Next Semester (and every Spring!)

CS 421: “Programming Languages”
• Learn what goes into a language!
• Be able to write an interpreter for the language of your choice!
• Learn functional programming in Haskell!
End of Semester Logistics

Regrades on MPs/Labs:
• Regrades are being processed today/tomorrow.
• I will make a Piazza update once grade updates are complete; will follow-up via Piazza.
My Passion: Data Discovery

http://waf.cs.illinois.edu/discovery/

GPAs at Illinois:

Diversity at Illinois:

And others:
CS 305: Data Driven Discovery (Fall 2018)

• Non-majors (no CS, no ECE)
  (Sorry, not my decision! Department feels data visualization in Python is too simple for CS credit.)
  • Benefit: Everyone is nearly on the same playing field – passion of data with core programming tools

• Next offering: Fall 2018!
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

| 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]

A   | B   | C   | D
---|-----|-----|-----
A   |     |     |     
B   |     |     |     
C   |     |     |     
D   |     |     |     

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

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></td>
<td></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>0</td>
<td></td>
</tr>
</tbody>
</table>

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:

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>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**

<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>

Initially:

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

- $A \rightarrow B$: -1
- $A \rightarrow C$: 2
- $B \rightarrow C$: 4
- $B \rightarrow D$: 3
- $C \rightarrow D$: -2
- $D \rightarrow A$: 2

Initially:

$\begin{align*}
&\text{foreach (Vertex } u : G): \\
&\quad \text{foreach (Vertex } v : G): \\
&\quad \quad \text{foreach (Vertex } w : G): \\
&\quad \quad \quad \text{if } d[u, v] > d[u, w] + d[w, v]: \\
&\quad \quad \quad \quad d[u, v] = d[u, w] + d[w, v]
\end{align*}$
Floyd-Warshall Algorithm

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>5</td>
<td>0</td>
<td>4</td>
<td>2</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>0</td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
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]
```

Let u = B; v and w explores for better paths:
Floyd-Warshall Algorithm

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>5</td>
<td>0</td>
<td>4</td>
<td>2</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>0</td>
<td></td>
</tr>
</tbody>
</table>
```

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

- \( d[u, v] = d[u, w] + d[w, v] \)

Initially:

\[
\begin{align*}
A &\rightarrow B & -1 \\
B &\rightarrow C & 4 \\
B &\rightarrow D & 3 \\
C &\rightarrow D & -2 \\
D &\rightarrow A & 2
\end{align*}
\]

\[\begin{array}{c}
A \rightarrow C & 2 \\
A \rightarrow D & 1
\end{array}\]

...explores:

\[\begin{array}{c}
B \rightarrow C & +\infty \\
B \rightarrow D & 3 + 2 = 5 \text{ UPDATE!}
\end{array}\]

...explores:

\[\begin{array}{c}
B \rightarrow A & 5 + 2 = 7 > 4, \text{ no update} \\
B \rightarrow D & +\infty
\end{array}\]

...explores:

\[\begin{array}{c}
B \rightarrow A & 5 + 1 = 6 > 3, \text{ no update} \\
B \rightarrow C & 4 + (-2) = 2 \text{ UPDATE!}
\end{array}\]
Shortest Path Algorithms Runtime:

• Dijkstra’s Algorithm: $O(m + n \lg(n))$

• Floyd-Warshall: $O(n^3)$

All Pairs Shortest Path:

Dense Graphs:

Sparse Graphs:
Graphs

Graph Implementations:
• Edge List
• Adjacency Matrix
• Adjacency List

Graph Traversals:
• Breadth First
• Depth First

Minimum Spanning Trees:
• Kruskal’s Algorithm
• Prim’s Algorithm

Shortest Path:
• Dijkstra’s Algorithm
• Floyd-Warshall’s Algorithm
Exam 13: Makeup Exam starts today
More Info: https://courses.engr.illinois.edu/cs225/fa2017/exams/

MP7: The final MP!
Due: Monday, Dec. 11 at 11:59pm

Final Exam starts Thursday!
Worth 250 points, the largest assessment all semester!