CS 225

Data Structures

December 6 – Floyd-Warshall's Algorithm

G Carl Evans

Floyd-Warshall Algorithm

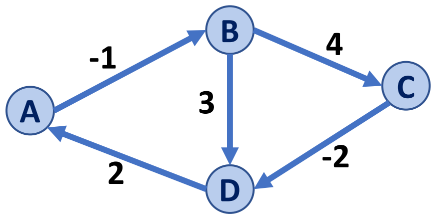
Floyd-Warshall’s Algorithm is an alterative to Dijkstra in the presence of negative-weight edges (not negative weight cycles).

```
FloydWarshall(G):
  Let d be a adj. matrix initialized to +inf
  foreach (Vertex v : G):
    d[v][v] = 0
  foreach (Edge (u, v) : G):
    d[u][v] = cost(u, v)

  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

```
FloydWarshall(G):
  Let d be a adj. matrix initialized to +inf
  foreach (Vertex v : G):
    d[v][v] = 0
  foreach (Edge (u, v) : G):
    d[u][v] = cost(u, v)

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

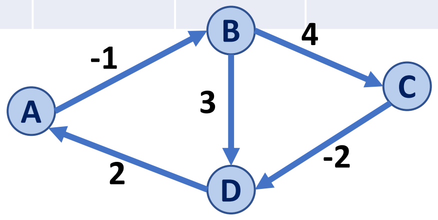
	A	B	C	D
A				
B				
C				
D				



Floyd-Warshall Algorithm

```
foreach (Vertex k: G):
  foreach (Vertex u : G):
    foreach (Vertex v : G):
      if d[u, v] > d[u, k] + d[k, v]:
        d[u, v] = d[u, k] + d[k, v]
```

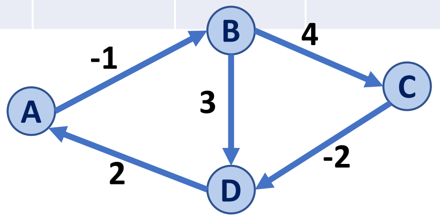
	A	B	C	D
A	0	-1	∞	∞
B	∞	0	4	3
C	∞	∞	0	-2
D	2	∞	∞	0



Floyd-Warshall Algorithm

```
foreach (Vertex k: G):
  foreach (Vertex u : G):
    foreach (Vertex v : G):
      if d[u, v] > d[u, k] + d[k, v]:
        d[u, v] = d[u, k] + d[k, v]
```

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



Let us consider k=A:

$B \rightarrow C$ 4 vs. $B \rightarrow A \rightarrow C$ $+\infty$

$B \rightarrow D$ 3 vs. $B \rightarrow A \rightarrow D$ $+\infty$

$C \rightarrow B$ $+\infty$ vs. $C \rightarrow A \rightarrow B$ $+\infty$

$C \rightarrow D$ -2 vs. $C \rightarrow A \rightarrow D$ $+\infty$

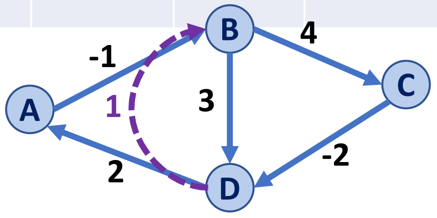
$D \rightarrow B$ $+\infty$ vs. $D \rightarrow A \rightarrow B$

$D \rightarrow C$ $+\infty$ vs. $D \rightarrow A \rightarrow C$

Floyd-Warshall Algorithm

```
foreach (Vertex k : G):
  foreach (Vertex u : G):
    foreach (Vertex v : G):
      if d[u, v] > d[u, k] + d[k, v]:
        d[u, v] = d[u, k] + d[k, v]
```

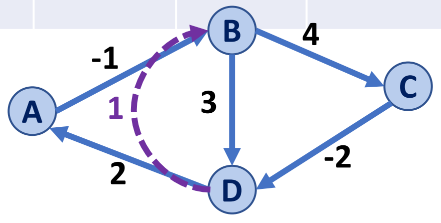
	A	B	C	D
A	0	-1	∞	∞
B	∞	0	4	3
C	∞	∞	0	-2
D	2	1	∞	0



Floyd-Warshall Algorithm

```
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]
```

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



Let us consider k=B:

$A \rightarrow C$ vs. $A \rightarrow B \rightarrow C$

$A \rightarrow D$ vs. $A \rightarrow B \rightarrow D$

$C \rightarrow A$ vs. $C \rightarrow B \rightarrow A$

$C \rightarrow D$ vs. $C \rightarrow B \rightarrow D$

$D \rightarrow A$ vs. $D \rightarrow B \rightarrow A$

$D \rightarrow C$ vs. $D \rightarrow B \rightarrow C$

Floyd-Warshall Algorithm

Running Time?

```
FloydWarshall(G):
  Let d be a adj. matrix initialized to +inf
  foreach (Vertex v : G):
    d[v][v] = 0
  foreach (Edge (u, v) : G):
    d[u][v] = cost(u, v)

  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]
```

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!