CS 225

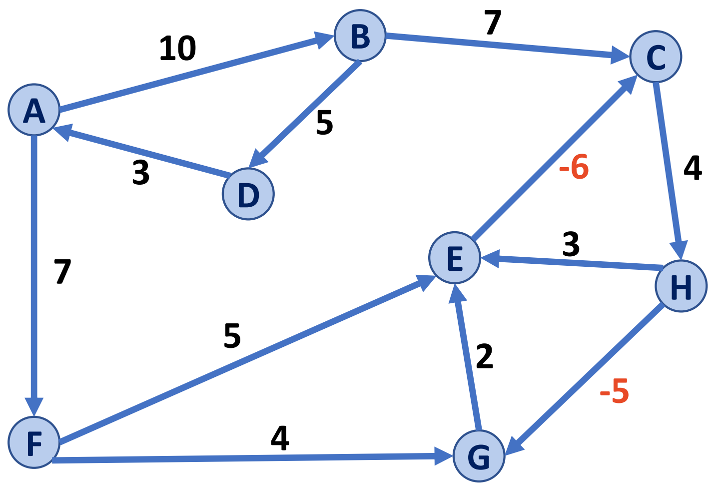
Data Structures

April 29 – Floyd-Warshall's Algorithm

Wade Fagen-Ulmschneider, Craig Zilles

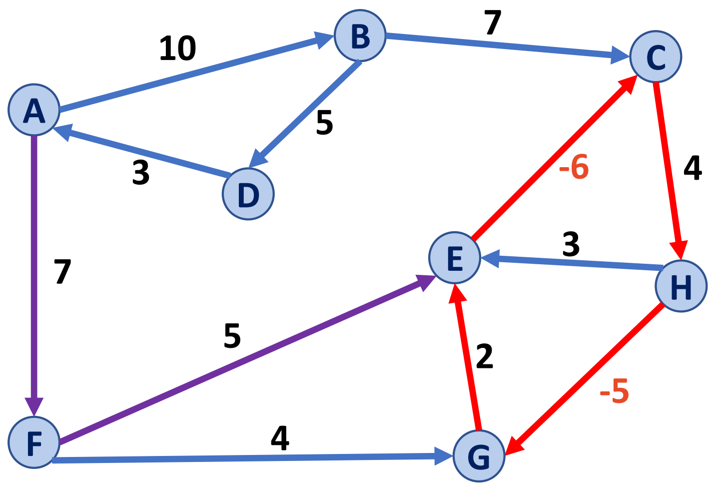
Dijkstra's Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?



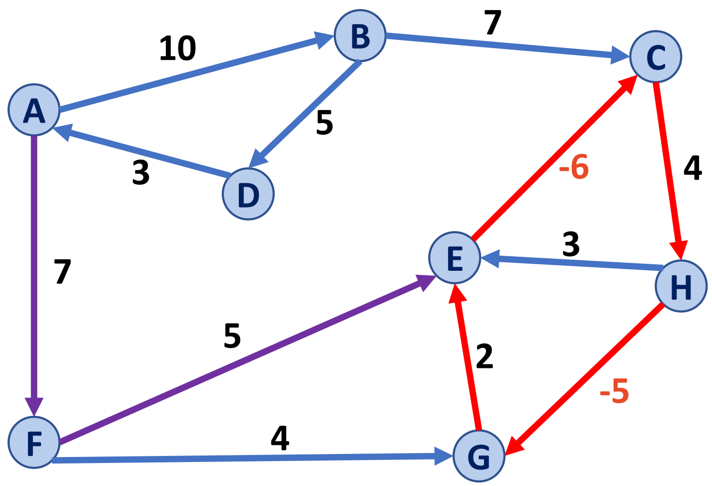
Dijkstra's Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?



Dijkstra's Algorithm (SSSP)

Q: How does Dijkstra handle negative weight cycles?

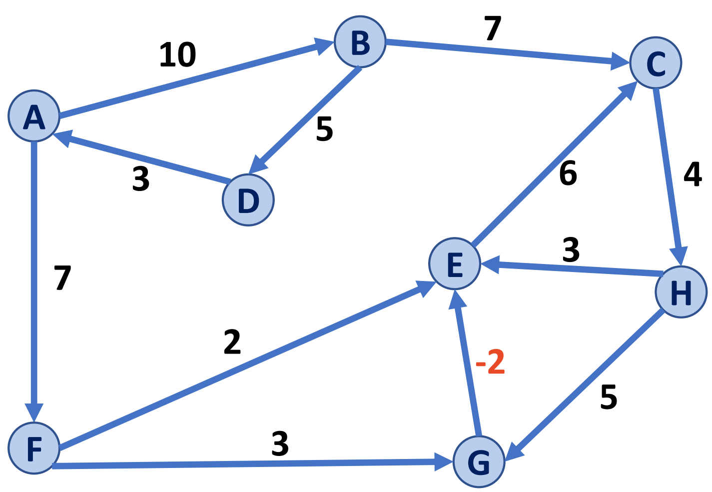


Shortest Path (A ➔ E): A → F → E → (C → H → G → E)*

Length: 12   Length: -5 (repeatable)

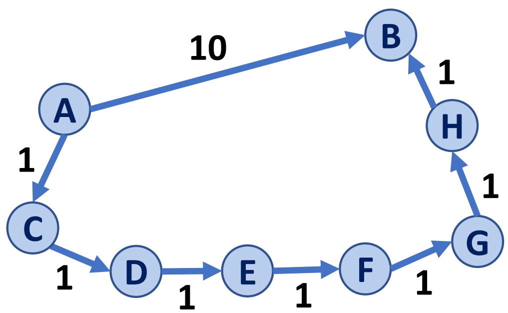
Dijkstra's Algorithm (SSSP)

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

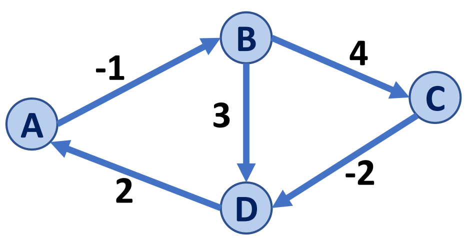
Floyd-Warshall's Algorithm is an alterative to Dijkstra in the presence of negative-weight edges (not negative weight cycles).

```
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 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]
```

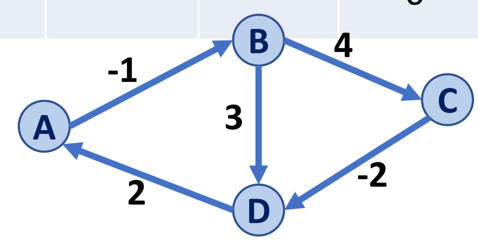
	A	B	C	D
A				
B				
C				
D				



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]
```

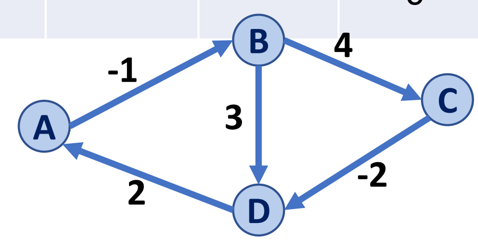
	A	B	C	D
A	0	-1	∞	∞
B	∞	0	4	3
C	∞	∞	0	-2
D	2	∞	∞	0



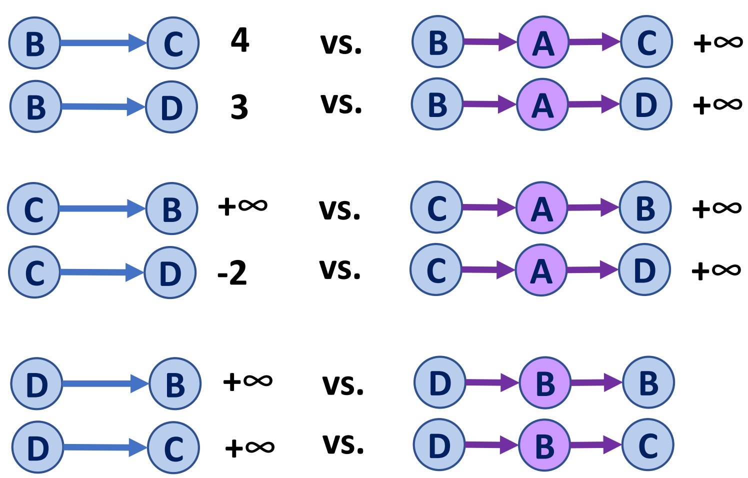
Floyd-Warshall Algorithm

```
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	∞	∞	0



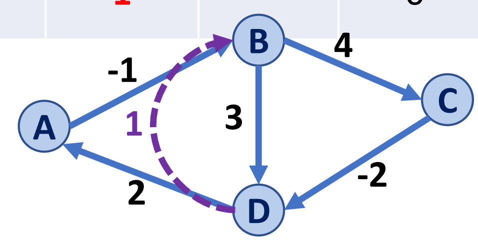
Let us consider k=A:



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]
```

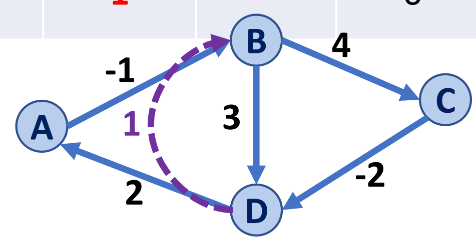
	A	B	C	D
A	0	-1	∞	∞
B	∞	0	4	3
C	∞	∞	0	-2
D	2	1	∞	0



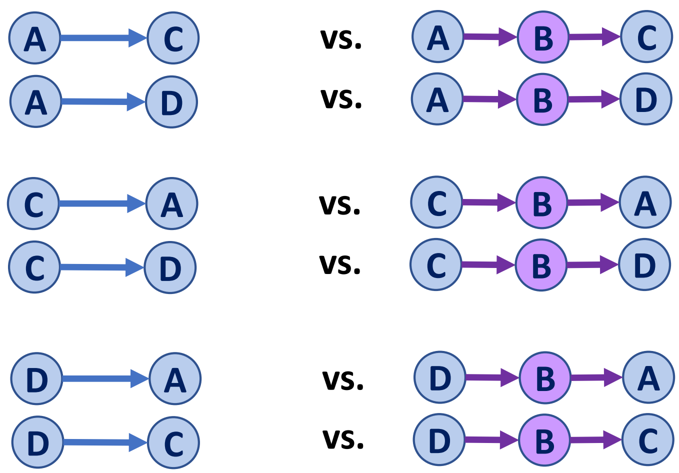
Floyd-Warshall Algorithm

```
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:



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 shortestPath (i,j,k) as:

shortestPath(i, j, 0) = *// base case*

shortestPath(i, j, k) = min(*// recursive*

)

Floyd-Warshall Algorithm Intuition

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

```
shortestPath(i, j, 0) = w(i, j)                                   // base case

shortestPath(i, j, k) = min( shortestPath(i, j, k-1),             // recursive
                        shortestPath(i, k, k-1) + shortestPath(k, j, k-1) )
```

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!