Graphs (cont.)

Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)

Lecture 15

sunho.lim@ttu.edu

Adapted partially from Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning; and Algorithms and Data Structures, Douglas Wilhelm Harder, Mmath

Shortest Paths: Dijkstra Algorithm

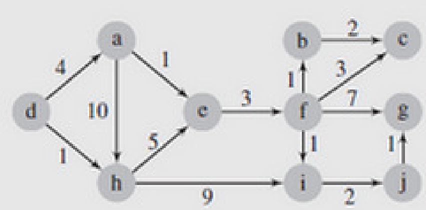
- Finding the shortest path between two nodes,
  - the edges of the graph associated with values, e.g., distance, time, costs, amounts, etc.

- **Dijkstra's algorithm,**
  - find the **shortest path** between **source node** and **every other node**
  - if the path is **longer** than any other path from that point, it is **dropped**, and the other path is expanded
  - each vertex is visited, the new paths are started, and the vertex is then not used anymore
  - once all the vertices are visited, the algorithm is done

Shortest Paths (cont.): Dijkstra Algorithm (cont.)

- Dijkstra's algorithm (cont.)

```
DijkstraAlgorithm (weighted simple digraph, vertex first)
    for all vertices v
      currDist(first) = inf;
    currDist(first) = 0;
    tobechecked = all vertices;
    while tobechecked is not empty
      v = a vertex in tobechecked with minimal currDist(v);
      remove v from tobechecked;
      for all vertices u adjacent to v and in tobechecked
           if currDist(u) > currDist(v) + weight (edge(v, u))
                currDist(u) = currDist(v) + weight (edge(v, u));
                predecessor(u) = v;
```

(a)

Dijkstra's approach may fail when negative weights are used in graphs

if currDist(u) > currDist(v) + weight (edge(v, u))
currDist(u) = currDist(v) + weight (edge(v, u));

iteration:	init	1	2	3	4	5	6	7	8	9	10
active vertex:		d	h	a	e	f	b	i	c	j	g
a	∞	4	4								
b	∞	∞	∞	∞	∞	9					
c	∞	∞	∞	∞	∞	11	11	11			
d	0										
e	∞	∞	6	5							
f	∞	∞	∞	∞	8						
g	∞	∞	∞	∞	∞	15	15	15	15	12	
h	∞	1									
i	∞	∞	10	10	10	9	9				
j	∞	∞	∞	∞	∞	∞	∞	11	11		

(b)

Shortest Paths (cont.): Bellman-Ford Algorithm

- Dijkstra's algorithm (cont.)
  - may fail when **negative weights** are used in graphs

- Bellman – Ford algorithm
  - use the **same technique as Dijkstra's method** to set the current distances
    - find the **shortest path** between **source node** and **every other node**
  - work when some weights are **negative**
  - **all edges** are watched in an attempt to find an improvement for the current distance of the vertices

Shortest Paths (cont.): Bellman-Ford Algorithm (cont.)

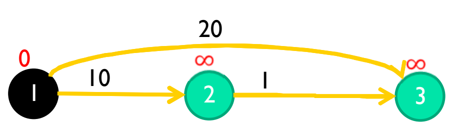
- Bellman – Ford algorithm (cont.)
  - basic idea
    - for all the paths have **at most 0 edge**, find all the shortest paths
    - for all the paths have **at most 1 edge**, find all the shortest paths
    - …
    - for all the paths have **at most |V|-1 edge**, find all the shortest paths

Shortest Paths (cont.): Bellman-Ford Algorithm (cont.)

- Bellman – Ford algorithm (cont.)

```
for each v {
    if(v==s) d_(s,v)=0; else d_(s,v)= ∞;
    π_(s,v)=NIL;  //set the predecessor of v on the shortest path
}
repeat |V|-1 times {
    for each edge (u, v) {
        if(d_(s,v) > d_(s,u)+ w_((u,v))){
            d_(s,v)=d_(s,u) + w_((u,v));
            π_(s,v)=u;
        }
}
for each edge (u, v)
    If (d_(s,v)>d_(s,u)+ w_((u,v))) return false; // there is no solution
}
return true;
```


Shortest Paths (cont.): Bellman-Ford Algorithm (cont.)



What is the **0-edge** shortest path from 1 to 1?

<> with path weight 0

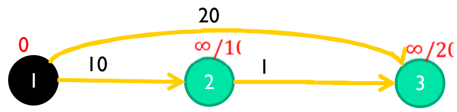
What is the **0-edge** shortest path from 1 to 2?

<> with path weight ∞

What is the **0-edge** shortest path from 1 to 3?

<> with path weight ∞

Shortest Paths (cont.): Bellman-Ford Algorithm (cont.)



What is the at most **1-edge** shortest path from 1 to 1?

<> with path weight 0

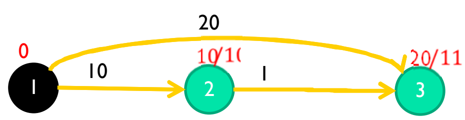
What is the at most **1-edge** shortest path from 1 to 2?

<1, 2> with path weight 10

What is the at most **1-edge** shortest path from 1 to 3?

<1, 3> with path weight 20

Shortest Paths (cont.): Bellman-Ford Algorithm (cont.)



What is the at most **2-edges** shortest path from 1 to 1?

<> with path weight 0

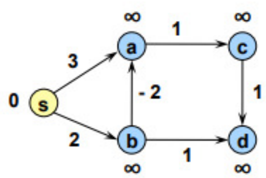
What is the at most **2-edges** shortest path from 1 to 2?

<1, 2> with path weight 10

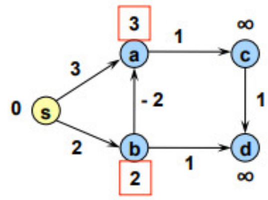
What is the at most **2-edges** shortest path from 1 to 3?

<1, 2, 3> with path weight 11

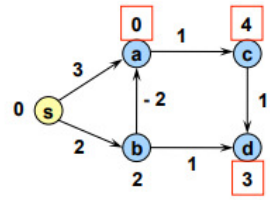
Shortest Paths (cont.): Bellman-Ford Algorithm (cont.)



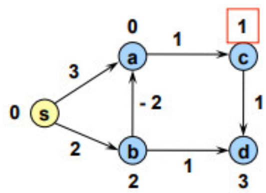
path lengths = 0



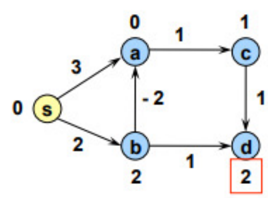
path lengths ≤ 1



path lengths ≤ 2



path lengths ≤ 3



path lengths ≤ 4

All-pairs Shortest Paths: Floyd-Warshall Algorithm

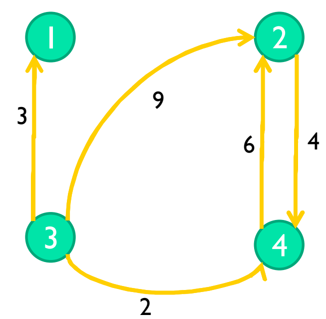
- **All-pairs shortest path** problem:
  - given a weighted, directed graph G=(V, E), for every pair of vertices, find a shortest path.

- Floyd-Warshall algorithm
  - **negative weights** may present, but no negative cycle
  - construct the **shortest path matrix**,

```
for( int i = 0; k < n; k++ )
   for( int j = 0; i < n; i++ )
      for( int k = 0; j < n; j++ )
         if weight[j][k] > weight[j][i] + weight[i][k]
            weight[j][k] = weight[j][i] + weight[i][k]
```

All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)

- For example,



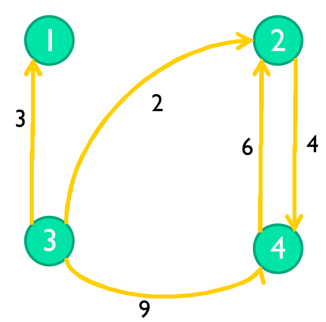
What are the weights of shortest paths with no intermediate vertices, D(0)?

	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	9	0	2
4	∞	6	∞	0

All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)

D(0)

	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	9
4	∞	6	∞	0

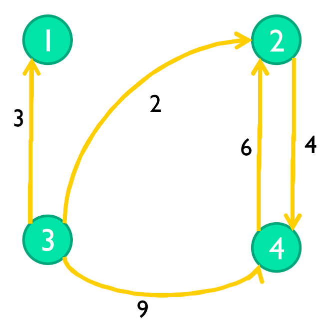


What are the weights of shortest paths with intermediate vertex 1?

D(1)

	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	9
4	∞	6	∞	0

All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)



D(1)

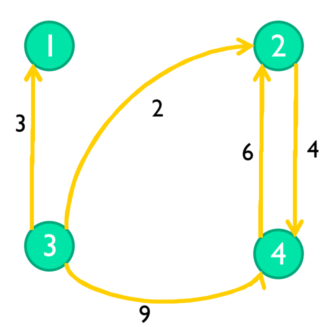
	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	9
4	∞	6	∞	0

What are the weights of shortest paths with intermediate vertices 1 and 2?

D(2)

	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	9
4	∞	6	∞	0

All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)



D(2)

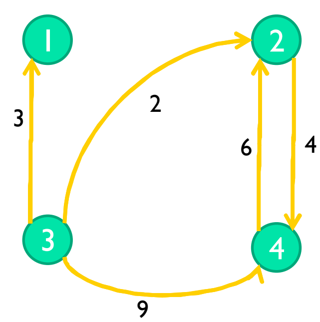
	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	9
4	∞	6	∞	0

What are the weights of shortest paths with intermediate vertices 1, 2 and 3?

D(3)

	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	6
4	∞	6	∞	0

All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)



D(3)

	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	6
4	∞	6	∞	0

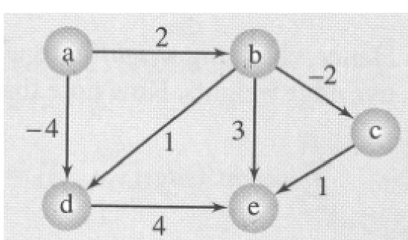
What are the weights of shortest paths with intermediate vertices 1, 2, 3 and 4?

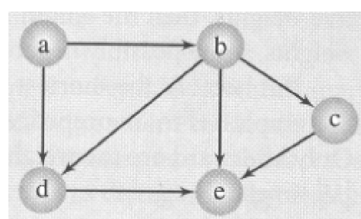
D(4)

	1	2	3	4
1	0	∞	∞	∞
2	∞	0	∞	4
3	3	2	0	6
4	∞	6	∞	0

All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)

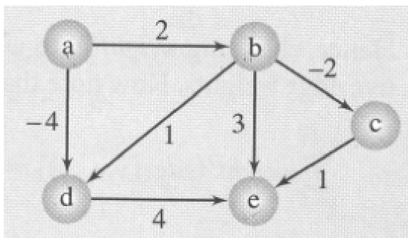
- Another example,

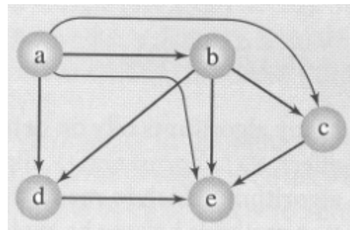




	a	b	c	d	e
$\boxed{a}$	0	2	∞	−4	∞
b	∞	0	−2	1	3
c	∞	∞	0	∞	1
d	∞	∞	∞	0	4
e	∞	∞	∞	∞	0

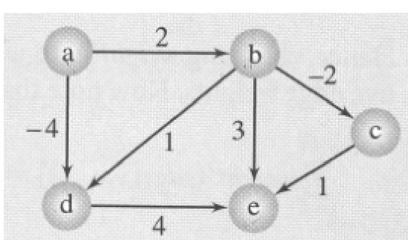
All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)

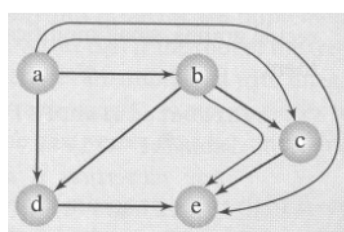




	a	b	c	d	e
a	0	2	0	−4	5
b	∞	0	−2	1	3
c	∞	∞	0	∞	1
d	∞	∞	∞	0	9
e	∞	∞	∞	∞	0

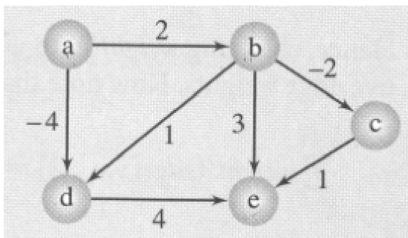
All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)

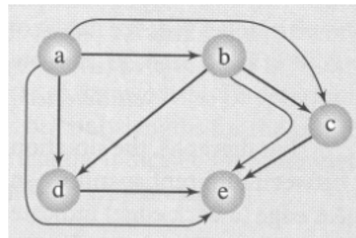




	a	b	c	d	e
a	0	2	0	−4	1
b	∞	0	−2	1	−1
c	∞	∞	0	∞	1
d	∞	∞	∞	0	4
e	∞	∞	∞	∞	0

All-pairs Shortest Paths (cont.): Floyd-Warshall Algorithm (cont.)





	a	b	c	d	e
a	0	2	0	−4	0
b	∞	0	−2	1	−1
c	∞	∞	0	∞	1
d	∞	∞	∞	0	4
e	∞	∞	∞	∞	0