Lecture

Graph

Shortest path

- In general, in graph theory the shortest path between two vertices in a graph is a path between those two vertices in such way that the sum of weights of edges in that path is the smallest than the sum of weights of edges in any path between those two vertices provided the graph is a weighted graph.

- So the shortest path problem is to find such a path between the given vertices.

Single-Source Shortest Path Problem

- **Single-Source Shortest Path Problem** - The problem of finding shortest paths from a source vertex *v* to all other vertices in the graph.
  - Dijkstra's Algorithm

Dijkstra's algorithm

Dijkstra's algorithm - is a solution to the single-source shortest path problem in graph theory.

Works on both directed and undirected graphs.

Approach: Greedy

Input: Weighted graph G={E,V} and source vertex $s \in V$,

Output: the shortest path from a given source vertex $s \in V$ to all other vertices

Dijkstra's Algorithm

1. d[s] -> 0
2. for each vertex $v \in \{V[G] - S\}$
3. do $d[v] = \infty$
4. $\pi[v] = NIL$
5. $S = \emptyset$
6. $Q = G.V$
7. While $Q \neq \emptyset$
8. do $u = EXTRACT \min(Q)$
9. $S = SU\{u\}$
10. for each vertex $v \in Adj[u]$
11. do $if\ (v \in Q\ and\ d[v] > d[u] + w(u, v))$
12. $d[v] = d[u] + w(u, v))$
13 $\pi[v] = u$

Analysis: O(E log V)

- 2-6 -> O(V)
- 8 -> O(V log V)
- 10-13 -> O(E log (V))

1. d[s] -> 0
2. for each vertex $v \in \{V[G] - S\}$
3. do $d[v] = \infty$
4. $\pi[v] = NIL$
5. $S = \emptyset$
6. $Q = G.V$
7. While $Q \neq \emptyset$
8. do $u = EXTRACT \min(Q)$
9. $S = SU\{u\}$
10. for each vertex $v \in Adj[u]$
11. do $if\ (v \in Q\ and\ d[v] > d[u] + w(u,v))$
12. $d[v] = d[u] + w(u,v))$
13 $\pi[v] = u$

All-Pairs Shortest Paths Problem

- Given a weighted digraph G=(V,E),determine the length of the shortest path (i.e., distance) between all pairs of vertices in G.

- It aims to compute shortest path from each vertex v to every other vertex u.

Floyd-Warshall algorithm

- Given a weighted graph, we want to know the shortest path from one vertex in the graph to another.
  - The Floyd-Warshall algorithm determines the shortest path between all pairs of vertices in a graph, if there is no negative cycle.

Main idea

- a path exists between two vertices i, j, if
  - there is an edge from i to j; or
  - there is a path from i to j going through intermediate vertices $\{1, \ldots . k\}$;

- These are two situations:
  1) k is an intermediate vertex on the shortest path.
  2) k is not an intermediate vertex on the shortest path.

Matrix Representation

- The graph is represented by an n x n matrix with the weights of the edges

- **Output Format:** an n x n distance $D=[\ d_{i,j}\ ]$ where $d_{i,j}$ is the distance from vertex i to j.

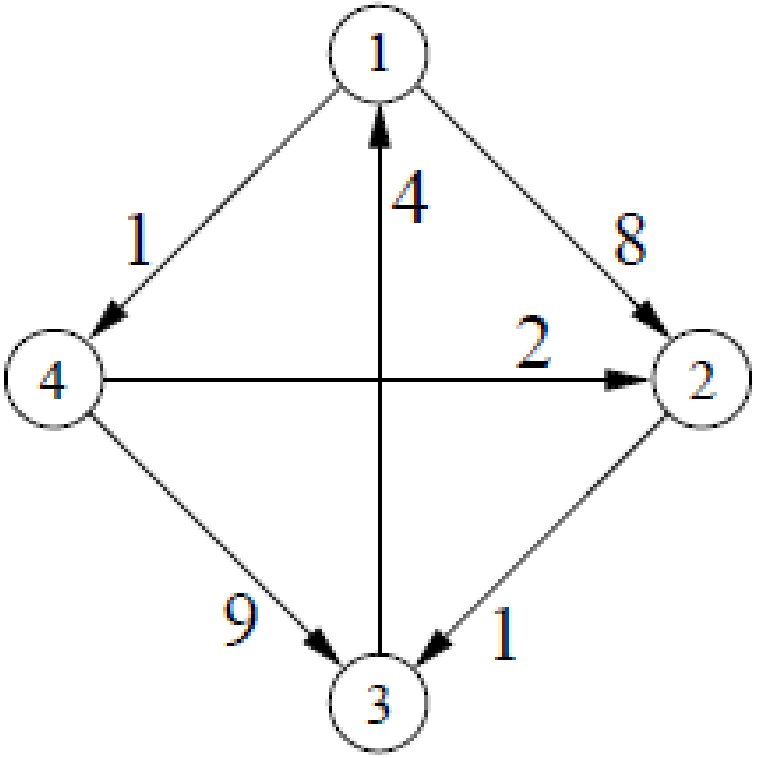
Floyd Warshall Algorithm

```
1.  Floyd_Warshall (W) {
2.  for i = 1 to n do { // initialize
3.      for j = 1 to n do {
4.          d[i,j] = W[i,j]
5.          pred[i,j] = null
6.                  }
7.                      }
8.  for k = 1 to n do                          // use intermediates {1..k}
9.      for i = 1 to n do              // …from i
10.         for j = 1 to n do                  // …to j
11.             if (d[i,k] + d[k,j]) < d[i,j]) {
12.                 d[i,j] = d[i,k] + d[k,j]    // new shorter path length
13.                 pred[i,j] = k   }           // new path is through k
14.                 return d                // matrix of final distances
15.                     }
```

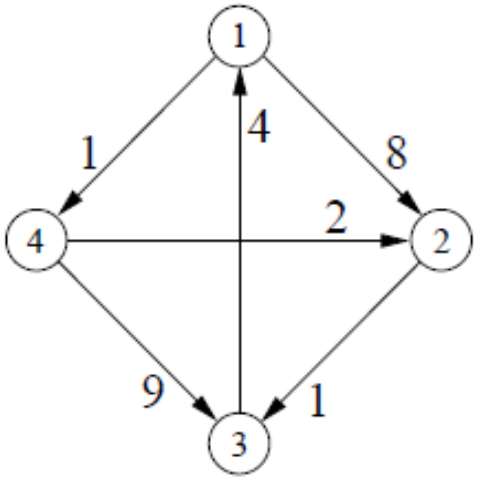
Compute shortest path using predecessor information

- The pred[I, j] can be used to extract the final path.
- Here is the idea, whenever we discover that the shortest path from i to j passes through an intermediate vertex k, we set pred[i,j] = k. If the shortest path does not pass through any intermediate vertex, then pred[I, j] = null.
- To find the shortest path from i to j, we consult pred[I,j]. If it is null, then the shortest path is just the edge (i; j). Otherwise, we recursively compute the shortest path from i to pred[i, j] and the shortest path from pred[I, j] to j.

Example



Initially



$$D^{(0)} = \begin{bmatrix} 0 & 8 & ? & 1 \\ ? & 0 & 1 & ? \\ 4 & ? & 0 & ? \\ ? & 2 & 9 & 0 \end{bmatrix}$$

? = infinity

$$P^{(0)} = \begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

Ist Iteration: k=1

$$D^{(0)} = \begin{bmatrix} 0 & 8 & ? & 1 \\ ? & 0 & 1 & ? \\ 4 & ? & 0 & ? \\ ? & 2 & 9 & 0 \end{bmatrix} \quad P^{(0)} = \begin{pmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{pmatrix}$$

? = infinity

```
if (d[i,k] + d[k,j]) < d[i,j]) {
    d[i,j] = d[i,k] + d[k,j]
    pred[i,j] = k   }
```

$$D^{(1)} = \begin{bmatrix} 0 & 8 & ? & 1 \\ ? & 0 & 1 & ? \\ 4 & 12 & 0 & 5 \\ ? & 2 & 9 & 0 \end{bmatrix} \quad P^{(1)} = \begin{pmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 \end{pmatrix}$$

2nd Iteration: k=2

```
if (d[i,k] + d[k,j]) < d[i,j]) {
    d[i,j] = d[i,k] + d[k,j]
    pred[i,j] = k   }
```

$$D^{(1)} = \begin{bmatrix} 0 & 8 & ? & 1 \\ ? & 0 & 1 & ? \\ 4 & 12 & 0 & 5 \\ ? & 2 & 9 & 0 \end{bmatrix} \quad P^{(1)} = \begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ ? & 0 & 1 & ? \\ 4 & 12 & 0 & 5 \\ ? & 2 & 3 & 0 \end{bmatrix} \quad P^{(2)} = \begin{bmatrix} 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 2 & 0 \end{bmatrix}$$

3rd Iteration: k=3

$$D^{(2)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ ? & 0 & 1 & ? \\ 4 & 12 & 0 & 5 \\ ? & 2 & 3 & 0 \end{bmatrix} \quad P^{(2)} = \begin{pmatrix} 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 2 & 0 \end{pmatrix}$$

```
if (d[i,k] + d[k,j]) < d[i,j]) {
    d[i,j] = d[i,k] + d[k,j]
    pred[i,j] = k   }
```

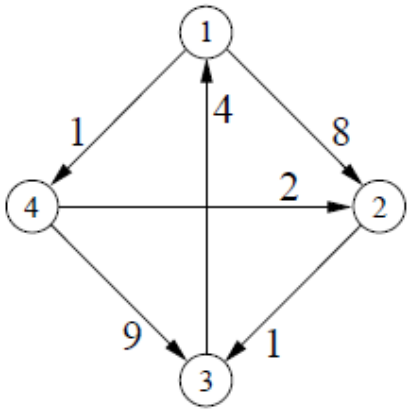
$$D^{(3)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \quad P^{(3)} = \begin{pmatrix} 0 & -1 & 2 & -1 \\ 3 & 0 & -1 & 3 \\ -1 & 1 & 0 & 1 \\ 3 & -1 & 2 & 0 \end{pmatrix}$$

4^{th} Iteration: k=4

$$D^{(3)} = \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \quad \mathbf{P^{(3)}} = \begin{pmatrix} 0 & -1 & 2 & -1 \\ 3 & 0 & -1 & 3 \\ -1 & 1 & 0 & 1 \\ 3 & -1 & 2 & 0 \end{pmatrix}$$

```
if (d[i,k] + d[k,j]) < d[i,j]) {
    d[i,j] = d[i,k] + d[k,j]
    pred[i,j] = k   }
```

$$D^{(4)} = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \quad \mathbf{P^{(4)}} = \begin{pmatrix} 0 & 4 & 4 & -1 \\ 3 & 0 & -1 & 3 \\ -1 & 4 & 0 & 1 \\ 3 & -1 & 2 & 0 \end{pmatrix}$$



$$P^{(4)} = \begin{bmatrix} 0 & -1 & 2 & -1 \\ 3 & 0 & -1 & 3 \\ -1 & 4 & 0 & 1 \\ 3 & -1 & 2 & 0 \end{bmatrix}$$

- Shortest Path from 1 to 4

  1->4
- Shortest Path from 4 to 3

  4->3 i-e 4->2->3
- Shortest Path from 3 to 2

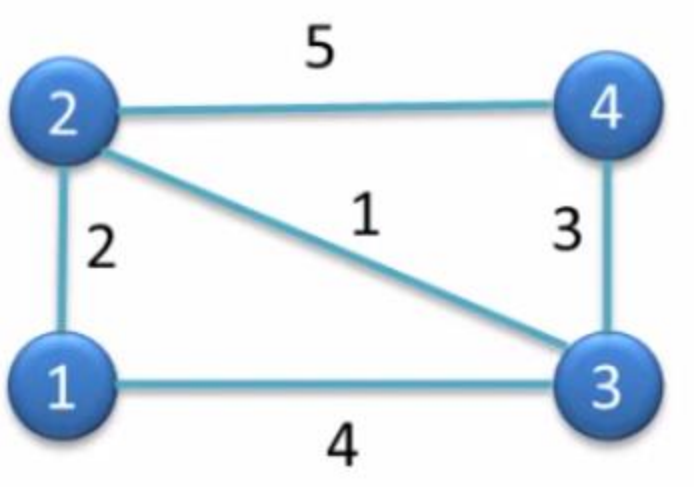
  3->1->4->2

$$D^{(4)} = \begin{bmatrix} 0 & 3 & 4 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 7 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix}$$

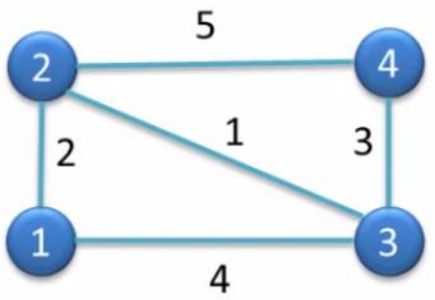
Analysis of Floyd Warshall Algorithm

- $O(n^3)$

Task



Initially



D_0	1	2	3	4
1	0	2	4	∞
2	2	0	1	5
3	4	1	0	3
4	∞	5	3	0

Distance Table

S_0	1	2	3	4
1	0	-1	-1	-1
2	-1	0	-1	-1
3	-1	-1	0	-1
4	-1	-1	-1	0

Sequence Table

Compute shortest path using predecessor information

```
Path(i,j) {
    if pred[i,j] = null
        output(i,j)
    else {
        Path(i, pred[i,j]);
        Path(pred[i,j], j);
    }
}
```