Bellman Ford (-ve weights)

All Pairs Shortest Paths

* directed weighted graph

↳ called as Floyd-Warshall algorithm
↳ used to find all pair of shortest paths problem from a given weighted graph.
↳ generate a matrix which represent the minimum distance from any node to all node in the graph.
↳ Problem is to find the shortest path between all the pairs of vertices in a weighted graph.
↳ follows dynamic programming approach.

↳ Let $G = \langle V, E \rangle$ be a directed graph where V is set of vertices and E is set of edges with nonnegative length.

n - vertices

Let cost be a cost adjacency matrix for G such that $cost(i,j) = 0$, $1 \le i \le n$.

$cost(i,j)$ is the length or cost of edge (i,j)

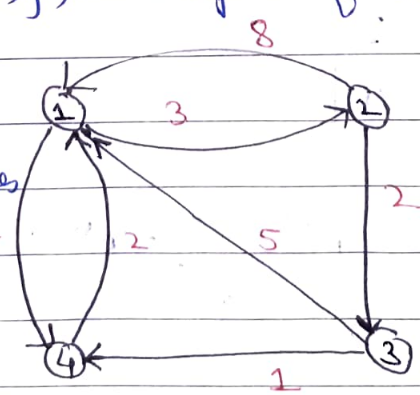
$cost(i,j) = 0$ if $i == j$ // Distance for of node from itself is zero

$cost(i,j) = \infty$ if $i \ne j$ and $(i,j) \notin E$

$cost(i,j) = w(i,j)$, if $i \ne j$ and $(i,j) \in E$
// $w(i,j)$ is weight of the edge (i,j)

Problem :- To find shortest path between every pair of vertices

Starting vertex = 1



$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Looks similar to Dijkstra's algorithm but Dijkstra's finds shortest path from one of source vertex & $O(n^2)$ time

Problem can be solved using greedy method $O(n^2 \times n) = O(n^3)$

Dynamic Programming
└ sequence of decisions to solve the problem.

↳ Start selecting the middle vertex as vertex 1.

Self loop → 0

no edge → ∞

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & & \\ 5 & & 0 & \\ 2 & & & 0 \end{bmatrix} \end{array}$$

→ remain unchanged (1st row & 1st column)

intermediate vertex = 1

Copy these values as it is

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 8 & \infty & 0 \end{bmatrix} \end{array}$$

$A^0[2,3] \quad\quad A^0[2,1] + A^0[1,3]$

$\downarrow \quad\quad 8 + \infty$

2

$2 < \infty$ → choose 2

if single source shortest path is implemented on all pair ~~using~~ Dijkstra's algo then complexity of

$E \log V$

but from all nodes we have to search shortest path

$V \cdot E \log V$

$E \rightarrow V^2$ value

no. of edges $= \frac{n(n-1)}{2}$

$V^3 \log V$

$A^0[2,4]$ $\quad$ $A^0[2,1] + A^0[1,4]$

∞ $\quad > \quad$ $8+7=15$

$A^0[3,2]$

$A^0[3,2]$ $\quad$ $A^0[3,1] + A^0[1,2]$

∞ $\quad > \quad$ $5+3=8$

$A^0[3,4]$

$A^0[3,4]$ $\quad$ $A^0[3,1] + A^0[1,4]$

1 $\quad\quad$ $5+7$

$1 \quad < \quad 12$

Intermediate vertex = 2 generated from intermediate vertex matrix 1

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & & \\ 2 & 8 & 0 & 2 & 15 \\ 3 & & 8 & 0 & \\ 4 & & 8 & & 0 \end{array}$$

2nd row & 2nd column as it is & diagonal = 0

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 8 & 7 & 0 \end{array}$$

$A'[1,3]$

$A'[1,3]$ $\quad$ $A'[1,2] + A'[2,3]$

∞ $\quad > \quad$ $3+2=5$

$A'[3,1]$

$A'[3,1]$ $\quad$ $A'[3,2] + A'[2,1]$

5 $\quad\quad$ $8+8$

$5 \quad < \quad 16$

$A'[1,4] - A'[1,2] + A'[1,4]$

$= 3 + 15$

$7 \quad < \quad 18$

$A'[3,4]$ $\quad$ $A'[3,2] + A'[2,4]$

1 $\quad\quad$ $8+15$

$1 \quad < \quad 23$

$$A^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 8 & 7 & 0 \end{array}$$

intermediate vertex 3

3rd & 3rd column as it is

$A^2[1,2] \quad A^2[1,3] + A^2[3,2]$

$3 \qquad 5 + 8$

$3 < 13$

$A^2[1,4] \quad A^2[1,3] + A^2[3,4]$

$7 \qquad 5 + 1$

$7 > 6$

$A^2[4,2] \quad A^2[4,3] + A^2(3,2]$

$8 \qquad 7 + 8$

$8 < 15$

$$A^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 & 3 \\ 3 & 3 & 6 & 0 & 1 \\ 4 & 2 & 8 & 7 & 0 \end{array}$$

4th row & 4th column as it is

shortest path of all vertices from other vertices.

intermediate vertex

previous

$$A^k[i,j] = \min\left\{ A^{k-1}[i,j],\ A^{k-1}[i,k] + A^{k-1}[k,j] \right\}$$

intermediate vertex

$$A(i,j) = \min\left\{ \min_{1 \le k \le n} \left\{ A^{k-1}(i,k) + A^{k-1}(k,j) \right\}, \text{cost}(i,j) \right\}$$

Consider n elements

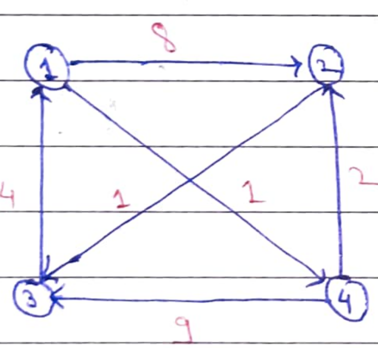
so the square matrix will be $n \times n$

generate the whole matrix

```
for(k=1; k<=n; k++)
{
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
        {
            A[i,j] = min(A[i,j], A[i,k] + A[k,j]);
        }
    }
}
```

generate only one matrix

Time complexity $O(n^3)$ — Three nested loops



Distance matrix

$$D^0 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & \infty & 0 & \infty \\ 4 & \infty & 2 & 9 & 0 \end{array}$$

via vertex 1

$$D^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & 12 & 0 & 5 \\ 4 & \infty & 2 & 9 & 0 \end{array}$$

$$D^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & 9 & 1 \\ 2 & \infty & 0 & 1 & \infty \\ 3 & 4 & 12 & 0 & 5 \\ 4 & \infty & 2 & 3 & 0 \end{array}$$

$$D^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & 9 & 1 \\ 2 & 5 & 0 & 1 & 6 \\ 3 & 4 & 12 & 0 & 5 \\ 4 & 7 & 2 & 3 & 0 \end{array}$$

$$D^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 4 & 1 \\ 2 & 5 & 0 & 1 & 6 \\ 3 & 4 & 7 & 0 & 5 \\ 4 & 7 & 2 & 3 & 0 \end{array}$$

$$A^k[i,j] = \min \begin{cases} A^{k-1}[i,j], \\ A^{k-1}[i,k] + A^{k-1}[k,j] \end{cases}$$

Time complexity

eg:- $A^2[4,3] = \min \begin{Bmatrix} A^1[4,3], \\ A^1[4,2] + A^1[2,3] \end{Bmatrix}$

$= \min\{9, (2+1)\}$

$= \min\{9, 3\}$

$A^2[4,3] = 3$

To create one matrix $= n^2$ $(n \times n)$
(no. of sub problems)

$\therefore$ For D^0 no need to have the eqn. By looking at the graph we can do it.
But from D^1 onwards we require matrix (eqn).
for $D^1 = n^2$ time
$D^2 = n^2$ time
⋮
$D^3 = n^2$ time

How many vertex in the graph = 4

$\therefore$ Total time complexity $= \underbrace{n}_{\text{no. of elements}} \times n^2 \rightarrow$ to create one matrix

$= O(n^3)$

Space complexity →

Size of matrix $= n^2$

To create D^1 only D^0 is used

D^2 — " — D^1

D^3 — " — D^2

D^4 — " — D^3

∴ $\boxed{\text{Space complexity} = n^2 \times 2}$ $O(n^2)$

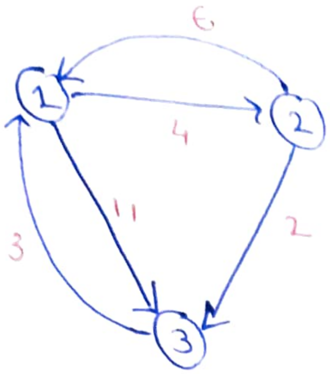
eg:- for D^1 use D^0

D^2 use D^1 and empty D^0 & add D^2 elements (remove elements of D^0)

D^3 use D^2 & remove element of D^1.

Algorithm AllPaths(cost, A, n)

```
// cost [1:n, 1:n] is cost of adjacency matrix of
// a graph with n vertices
// A[i, j] is cost of shortest path from vertex
// i to vertex j
// cost [i, i] = 0.0, for 1 ≤ i ≤ n
{
  for i := 1 to n do
    for j := 1 to n do
      A[i, j] := cost [i, j]; // copy cost into A
  for k := 1 to n do
    for i := 1 to n do
      for j := 1 to n do
        A[i, j] = min(A[i, j], A[i, k] + A[k, j]);
}
```



A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

$k=1$

$i=1$
$j=1$ $A[1,1] = \min(A[1,1], A[1,1] + A[1,1])$
$= \min(0,0) = 0$

$i=1$
$j=2$ $A[1,2] = \min(A[1,2], A[1,1] + A[1,2])$
$= \min(4, 0+4)$
$= \min(4,4) = 4$

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

$i=1$
$j=3$ $A[1,3] = \min(A[1,3], A[1,1] + A[1,3])$
$= \min(11, 0+11)$
$= 11$

$i=2$
$j=1$ $A[2,1] = \min(A[2,1], A[2,1] + A[1,1])$
$= \min(6, 6+0)$
$= 6$

$A[2,3] = \min(A[2,3], A[2,1] + A[1,3])$
$= \min(2, (6+11))$
$= \min(2,17)$
$= 2$

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0