33119

Page:
Date: / /

Assignment - 8

Title: Bellman-Ford Algorithm

Problem Statement:

Write a program to implement Bellaman-Ford Algorithm using Dynamic Programming and verify the time complexity.

Objective:

- Need and significance of Dynamic programming.
- General method of dynamic programming
- Bellman-Ford algorithm to and its analysis.

Theory:

Dynamic Programming Strategy:

Suppose you have problem which can be divided into independent sub problem to find sol^n Divide and conquer is the most strategy to deal with that kind of problem Dynamic Programming is an algorithm design technique for optimization problems: often minimizing or maximing.

Principle of Optimality:

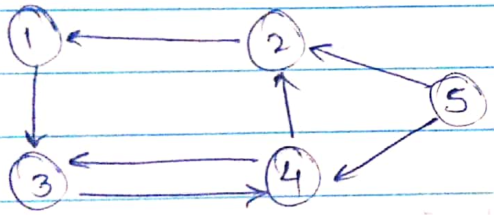
The Principle of optimality states that an optimal sequence of decision has the property that whatever the initial state and decision sequences with regards to the stage resulting from the first decision.

Shortest Path Problem :

Find the shortest paths from a directed acyclic weighted graph G(V, E) starting source to all reachable vertices. Bellman Ford works with negative weighted edges and gives error if negative cycle is found in graph.

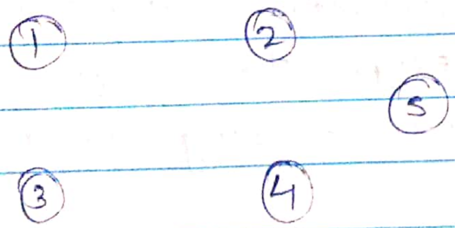
Bellman-Ford Algorithm example.

Given the following directed graph.



(1,3) = 6
(1,4) = 3
(2,1) = 3
(3,4) = 2
(4,2) = 1
(4,3) = 1
(5,2) = 4
(5,4) = 2

using vertex 5 as the source, we initialize all the other distances to ∞

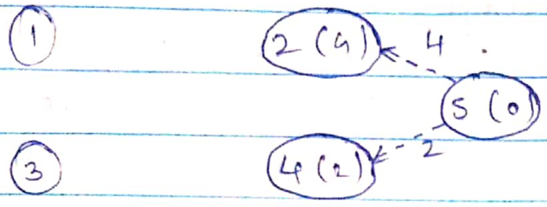


(1,3) = 6
(1,4) = 3
(2,1) = 3
(3,4) = 2
(4,2) = 1
(4,3) = 1
(5,2) = 4
(5,4) = 2

	1	2	3	4	5
d	∞	∞	∞	∞	0
π	1	1	1	1	1

Iteration 1 :-

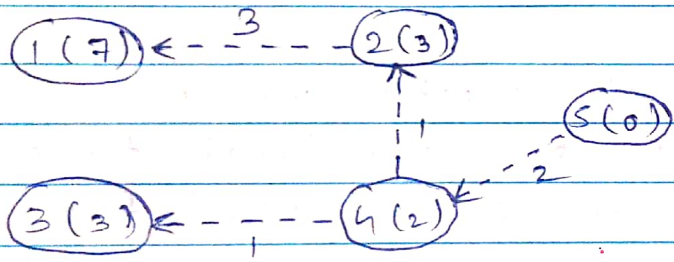
Edges (u_5, u_2) and (u_5, u_4) relax updating the distances to 2 and 4:



	1	2	3	4	5
d	∞	4	∞	2	0
π	1	5	1	5	1

Iteration 2 :

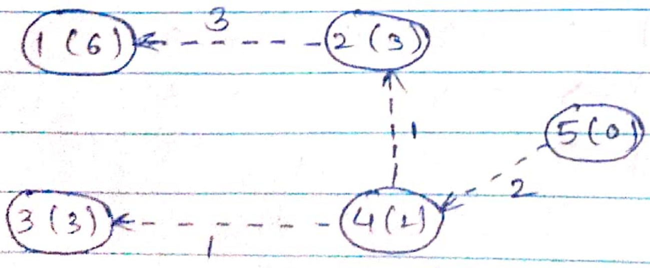
Edges (u_2, u_1), (u_4, u_2) and (u_4, u_3) relax updating the distances to 1, 2 & 4 resp. Note edge (u_4, u_2). Finds a shorter path to vertex 2 by going through vertex 4.



	1	2	3	4	5
d	7	3	3	2	0
π	2	4	4	5	1

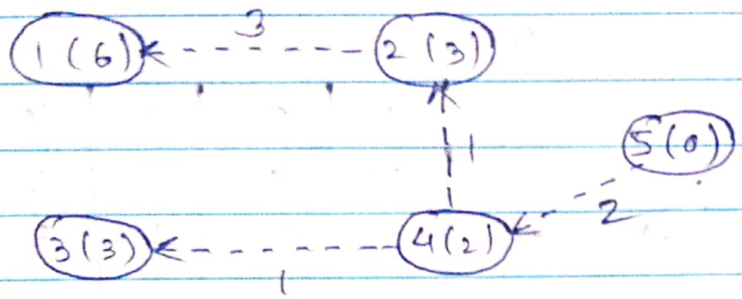
Iteration 3 :

Edge (u_2, u_1) relaxes (since a shorter path to vertex 2 was found in previous iteration) updating the distance to 1



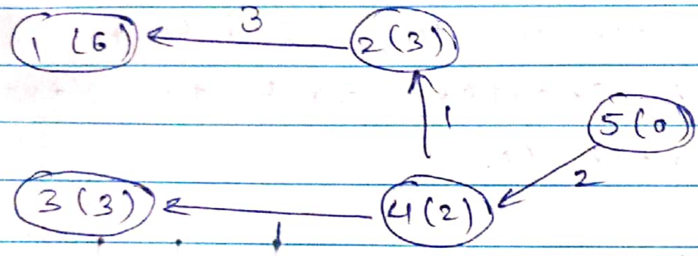
	1	2	3	4	5
d	6	3	3	2	0
π	2	4	4	5	1

Iteration 4: No edge relax



	1	2	3	4	5
d	6	3	3	2	0
π	2	4	4	5	1

The final shortest paths from vertex 5 with corresponding distance is



	1	2	3	4	5
d	6	3	3	2	0
π	2	4	4	5	1

Negative cycle checks we now check the relaxation condition one additional time for each edge. If any of the checks pass then there exists a negative weight cycle in the graph

$V_3.d > u_1.d + w(1,3) \Rightarrow 4 \text{ D } 6 + 6 = 12$

$V_4.d > u_1.d + w(1,4) \Rightarrow 2 \text{ D } 6 + 3 = 9$

$V_1.d > u_2.d + w(2,1) \Rightarrow 6 \text{ D } 3 + 3 = 6$

$V_4.d > u_3.d + w(3,4) \Rightarrow 2 \text{ D } 3 + 2 = 5$

$V_2.d > u_4.d + w(4,2) \Rightarrow 3 \text{ D } 2 + 1 = 3$

$V_3.d > u_4.d + w(4,3) \Rightarrow 3 \text{ D } 2 + 1 = 3$

$V_2.d > u_5.d + w(5,2) \Rightarrow 3 \text{ D } 0 + 4 = 4$

$V_4.d > u_5.d + w(5,4) \Rightarrow 2 \text{ D } 0 + 2 = 2$

The path to any reachable vertex can be found by starting at the vertex and foll. the Π's back to source.

Algorithm :

Bellman - Ford (G, W, S)

1. Initialize - Single - source
2. For i = 1 to |G.V| - 1
3. For each edge (u, v) ∈ G. E
4. Relax (u, v, w)
5. for each edge (u, v) ∈ G. E
6. if v.d > u.d + w (u, v)
7. return False
8. return true

Initialize - Single - Source (G, S)

1. For each vertex V ∈ G.V
2. V.d = ∞
3. V.pi = NIL
4. S.d = 0

Relax (u, v, w)

1. if v.d > u.d + w (u, v)
2. v.d = u.d + w (u, v)
3. v.pi = u

Scanned with CamScanner

Page :
Date : / /

Complexity :-

Basically the algorithm works as follows :

1. Initialize ds, π's, and set s.d = 0 ⇒ O(V)
2. Loop |V|-1 times through all edges cehecking the relaxation condition to compute minimum distances ⇒ (|V|-1) O(E) = O(VE)
3. Loop through all edges checking for negative weight cycles which occurs if any of the relaxa-tion conditions fail → O(E)

The run time of the Bellman-Ford algorithm is O(V+VE+E) = O(VE)

Conclusion : Bellman Ford algo to find single source shortest path is studied and impleme-nted using Dynamic Programming method.