Maze Project Method Bellman Ford's Algorithm to find the shortest path of a maze

Master Of Science

Presented by MANICKAM RAVISEKAR ,
Master of Science in Computer Science,
19599 , Fall 2021

Guidance from Dr., Professor Henry Chang

North Western Polytechnic University
47671 WestingHouse Dr., Fremont, CA 94539

CONTENTS

ACKNOWLEDGEMENT

Our Master's Project on Bellman Ford algorithm for given MAZE was an interesting network analysis and made me to learn new things also it is very useful in designing algorithm for varieties of project by comparing the other algorithms like Dijkstra.

To compare Bellman Ford Algorithm with Dijkstra’s algorithm.

It will allow us which one to select under given condition.

During the learning process I would like to thank Dr. Henry Chang for providing us key input to complete the assignment within give time frame.

Also, for all I would like to always pray to Almighty for giving us wisdom and power to understand things.

ABSTRACT

The main lesson about this assignment is to to study graph search algorithm that finds the shortest path between a given source vertex and all other vertices in the graph or tree .

Bellman Ford Algorithm Explanation

Bellman Ford Algorithm:

This algorithm solves the single source shortest path problem of a directed graph G = (V, E) in which the edge weights may be negative. Examples of negative edge Imagine a logistic network where the weight w(i,j) of an edge ij is the cost to go from vertex i to vertex j.
If you made a business agreement with other companies to transport their products, w(i,j) would be a profit instead of a cost, so you can interpret this weight as a negative cost.
For example, if you have a deck of cards and have a graph to represent possible exchanges, the weights of the vertices may mean the price you need to pay to exchange.
In such case, negative weight of a route between some nodes would mean you are paid more than you pay for such a chain of exchanges.
Moreover, this algorithm can be applied to find the shortest path, if there does not exist any negative weighted cycle.
Total iterations it will undergo is |G.V| - 1 , In our graph tree we have 14 vertex it will iterate 14 – 1 = 13 iterations.

pseudo-code for the Bellman-Ford algorithm :

```
Bellman-Ford-Algorithm (G, w, s)
for each vertex v Є G.V
  v.d := ∞
  v.∏ := NIL
s.d := 0
for i = 1 to |G.V| - 1
  for each edge (u, v) Є G.E
    if v.d > u.d + w(u, v)
      v.d := u.d +w(u, v)
      v.∏ := u
for each edge (u, v) Є G.E
  if v.d > u.d + w(u, v)
    return FALSE
return TRUE
```

Comparison of Bellman Ford Algorithm with Dijkstra's algorithm

Dijkstra's algorithm solves the single-source shortest-paths problem on a directed weighted graph G = (V, E), where all the edges are non-negative (i.e., $w(u, v) \geq 0$ for each edge $(u, v) \in E$).
In the following algorithm, we will use one function Extract-Min(), which extracts the node with the smallest key.

Bellman Ford Algorithm
This algorithm solves the single source shortest path problem of a directed graph G = (V, E)
in which the edge weights may be negative.

BIG-O Comparison of Bellman-Ford and Dijkstra' algorithm

Big-O

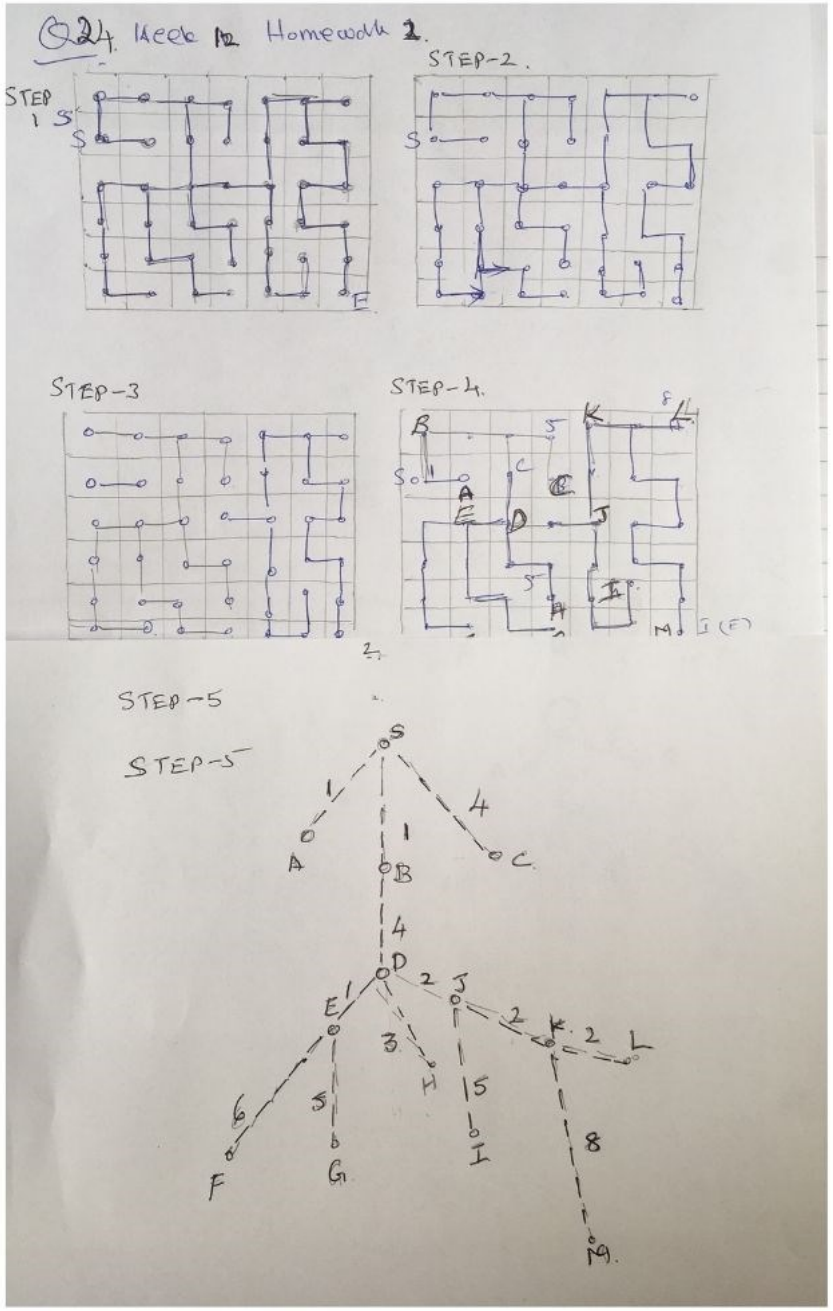
Bellman-Ford algorithm Analysis
The first for loop is used for initialization, which runs in O(V) times.
The next for loop runs |V - 1| passes over the edges, which takes O(E) times.
Hence, Bellman-Ford algorithm runs in O(V, E) time.

Big-O

Dijkstra's algorithm
The complexity of this algorithm is fully dependent on the implementation of Extract-Min function.
If extract min function is implemented using linear search, the complexity of this algorithm is O(V2 + E).
In this algorithm, if we use min-heap on which Extract-Min() function works to return the
node from Q with the smallest key, the complexity of this algorithm can be reduced further.

Fig: 1

Maze steps



Assignment

Bellman Ford's Algorithm to find the shortest path

		S	A	B	C	D	E	F	G	H	I	J	K	L	M
Iteration	1	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	2	0	1	1	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	3	0	1	1	4	5	∞	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	4	0	1	1	4	5	6	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	5	0	1	1	4	5	6	12	∞	∞	∞	∞	∞	∞	∞
Iteration	6	0	1	1	4	5	6	12	10	∞	∞	∞	∞	∞	∞
Iteration	7	0	1	1	4	5	6	12	10	8	∞	∞	∞	∞	∞
Iteration	8	0	1	1	4	5	6	12	10	8	∞	7	∞	∞	∞
Iteration	9	0	1	1	4	5	6	12	10	8	12	7	∞	∞	∞
Iteration	10	0	1	1	4	5	6	12	10	8	12	7	9	∞	∞
Iteration	11	0	1	1	4	5	6	12	10	8	12	7	9	11	∞
Iteration	12	0	1	1	4	5	6	12	10	8	12	7	9	11	17
Iteration	13	0	1	1	4	5	6	12	10	8	12	7	9	11	17

Edges processed for getting the shortest path

Let all edges are processed in the following order

(S,A),(S,B),(S,C),(B,D),(D,E),(E,F),(E,G),(D,H),(D,J),(J,I),(J,k),(K,L),(K,M)

Fig: 2

Distance computation Table for each vertices from source S

Solution Explanation And The Relaxation Formula Used

Fig: 3

For the step-5 tree of Fig 4 ,The execution of the Bellman-Ford Algorithm is source is vertex S and the distance d values are shown as numeric value for each edge in step 5 figure 3.

In this particular example each pass relaxes the edges in the order

Edges processed in the following order

(S,A),(S,B),(S,C),(B,D),(D,E),(E,F),(E,G),(D,H),(D,J),(J,I),(J,K),(K,L),(K,M)

As shown beside calculation worksheet Figure 3.

The Bellman-Ford algorithm returns TRUE in this example as shown in the worksheet diagram.

a) Do following for each edge u-v
   ……If distance[v] > distance[u] + weight of edge uv, then update distance[v]
   ……distance[v] = distance[u] + weight of edge uvg
b) As in diagram iteration after iterations 12 the values remains unchanged so bellman-ford algorithm returns TRUE.

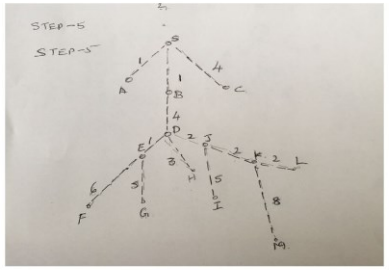
Assignment

Bellman Ford's Algorithm to find the shortest path

		S	A	B	C	D	E	F	G	H	I	J	K	L	M
Iteration	1	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	2	0	1	1	4	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	3	0	1	1	4	5	∞	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	4	0	1	1	4	5	6	∞	∞	∞	∞	∞	∞	∞	∞
Iteration	5	0	1	1	4	5	6	12	∞	∞	∞	∞	∞	∞	∞
Iteration	6	0	1	1	4	5	6	12	10	∞	∞	∞	∞	∞	∞
Iteration	7	0	1	1	4	5	6	12	10	8	∞	∞	∞	∞	∞
Iteration	8	0	1	1	4	5	6	12	10	8	∞	7	∞	∞	∞
Iteration	9	0	1	1	4	5	6	12	10	8	12	7	∞	∞	∞
Iteration	10	0	1	1	4	5	6	12	10	8	12	7	9	∞	∞
Iteration	11	0	1	1	4	5	6	12	10	8	12	7	9	11	∞
Iteration	12	0	1	1	4	5	6	12	10	8	12	7	9	11	17
Iteration	13	0	1	1	4	5	6	12	10	8	12	7	9	11	17

Edges processed for getting the shortest path
Let all edges are processed in the following order
(S,A),(S,B),(S,C),(B,D),(D,E),(E,F),(E,G),(D,H),(D,J),(J,I),(J,k),(K,L),(K,M)

Fig 4 , step 5 of previous slide



Conclusion

Application usage of Bellman-Ford Algorithm

1) Bellman-Ford is used in the routing protocol in network designing , it helps to arrive a solution how to route packets of a data on network , that is it will give weights in the path of network and number of routers we can consider to reach the end point.
2) In gateway routing protocol assist machine exchange path data within a network

References : NPU course material