ECE686 - PROJECT 1

Wireless Communication Networks

Author
Amirhosein Yari
University of Alberta
1634761

Contents

1 Bellman-Ford Algorithm

1.1 Algorithm summery

In Bellman-Ford Algorithm, we update our path each time with the following formula: $D_i^h = min_j[d_{ij} + D_j^{h-1}]$ for all $i \neq 1$ and save the path which makes these minimums feasible.

1.2 Code

I used Python programming language to implement the Bellman-Ford Algorithm.
Firstly I defined the graph by each weight and in a matrix representation, the G[i,j] element is the weight from node i+1 to j+1. The weight (distance) of each node with itself is obviously zero: G[i,i] = 0 and those nodes which are not connected have a distance of infinity (float("Inf")). And other weights are as follows:

```
import numpy as np
N = 7 #number of nodes of graph
G = np.ones([N,N])*float("Inf")
for i in range(N):
    G[i,i] = 0 #each node with it self

#node 1
G[0,1] = 4
G[1,0] = 4
G[0,2] = 5
G[2,0] = 5
#node 2
G[1,2] = 6
G[2,1] = 6
G[1,3] = 3
G[3,1] = 3
G[1,4] = 10
G[4,1] = 10
#node 3
G[2,3] = 3
G[3,2] = 4
G[2,5] = 9
G[5,2] = 9
#node 4
G[3,4] = 4
G[4,3] = 6
G[3,5] = 3
G[5,3] = 3
#node 5
G[4,5] = 2
G[5,4] = 3
G[4,6] = 2
G[6,4] = 2
#node 6
G[5,6] = 2
G[6,5] = 2
#node 7 is linked before (last node)
#print(G)
```

For Algorithm implementation there is matrix D which has distances in the h iteration in its rows (D[h]), and also the variable path is an array which has the path saved in it, each element shows the next node of the path: if path[i] = k, then from node i+1 you should go to node k to follow the shortest path and then from k, path[k-1] is containing the next node of the shortest tree path.
There are N (=7) iterations and for h = 0, distances to node 1 is infinity. After that, each distance gets an update with respect to the last step and in a np.min() function. For not choosing yourself in the process of argmin() selection there is an if statement and then the path[i] value is assigned. In case there is no way to node 1, the path is gonna be -1 (likely in the first step of h.).

```
path = np.zeros(N) #each node is conected to path value node in shortest path
D = np.zeros([N,N])
for h in range(N):
    if h==0:
        for i in range(N):
            D[h,i] = float("Inf")
        D[0,0] = 0
    else:
        D[h,0] = 0
        print("############In h =",h)
        for i in range(1,N):
            D[h,i] = np.min(G[i]+D[h-1])
            if np.argmin(G[i]+D[h-1]) != i: #not going through your self in
                                            min function
                path[i] = np.argmin(G[i]+D[h-1]) + 1
                if D[h,i] == float("Inf"):
                    path[i] = -1
            print("node i =",i+1, "path =",path[i], "D_hi =",D[h,i])
```

1.3 Results

Finally the result of each step is as follow:
————In h = 1
node i = 2 path = 1.0 D-hi = 4.0
node i = 3 path = 1.0 D-hi = 5.0
node i = 4 path = -1.0 D-hi = inf
node i = 5 path = -1.0 D-hi = inf
node i = 6 path = -1.0 D-hi = inf
node i = 7 path = -1.0 D-hi = inf
which means at the second (h=1) step nodes 2 and 3 are going through node 1 with distances of 4 and 5, other nodes has no way to node 1 and path is -1 and distance is infinity. Other steps are as follow:
————In h = 2
node i = 2 path = 1.0 D-hi = 4.0
node i = 3 path = 1.0 D-hi = 5.0
node i = 4 path = 2.0 D-hi = 7.0
node i = 5 path = 2.0 D-hi = 14.0
node i = 6 path = 3.0 D-hi = 14.0

node i = 7 path = -1.0 D-hi = inf
————In h = 3
node i = 2 path = 1.0 D-hi = 4.0
node i = 3 path = 1.0 D-hi = 5.0
node i = 4 path = 2.0 D-hi = 7.0
node i = 5 path = 4.0 D-hi = 13.0
node i = 6 path = 4.0 D-hi = 10.0
node i = 7 path = 5.0 D-hi = 16.0
————In h = 4
node i = 2 path = 1.0 D-hi = 4.0
node i = 3 path = 1.0 D-hi = 5.0
node i = 4 path = 2.0 D-hi = 7.0
node i = 5 path = 6.0 D-hi = 12.0
node i = 6 path = 4.0 D-hi = 10.0
node i = 7 path = 6.0 D-hi = 12.0
————In h = 5
node i = 2 path = 1.0 D-hi = 4.0
node i = 3 path = 1.0 D-hi = 5.0
node i = 4 path = 2.0 D-hi = 7.0
node i = 5 path = 6.0 D-hi = 12.0
node i = 6 path = 4.0 D-hi = 10.0
node i = 7 path = 6.0 D-hi = 12.0
————In h = 6
node i = 2 path = 1.0 D-hi = 4.0
node i = 3 path = 1.0 D-hi = 5.0
node i = 4 path = 2.0 D-hi = 7.0
node i = 5 path = 6.0 D-hi = 12.0
node i = 6 path = 4.0 D-hi = 10.0
node i = 7 path = 6.0 D-hi = 12.0

As it can be seen the algorithm got to the answer at step h=4 and since everything is unchanged. The path is from node 7 to 6, form 6 to 4, from 5 to 6, from 4 to 2, from 3 to 1 and from 2 to 1 directly. Or as it can be seen in the following graph: