Differentiation between Bellman-Ford and Dijsktra Algorithm in Routing Protocols

Pooja Ravi, Pragna B Rao

***Abstract*: *The modus operandi of congregating data pre-owned by home area network (HAN) protocols in order to deliver data across confined or extensive distance connection and that which is transmittal over a digital network in the form of packets is called as packet switching. Packet switching necessitates packaging of data in meagre units (packets) that are routed through a network, based on destination address contained within each packet and with the usage of network switches and routers. In packet switching networks, routing is the Avant-graded and multifaceted resolution maker, that controls the direction of network packets from their source in the vicinity of their destination through intermediate network nodes by precise packet advancing techniques. Here, in this disquisition we made an interpretation apropos the dyad pre-eminent shortest path (closest distance) searching algorithms, which are used in routing. They're the BELLMAN-FORD and DIJKSTRA'S algorithm. The anatomization of the differentiation between the two is given concisely*.**

Keywords*: *relaxation, shortest distance, routing protocol, source node, destination node.

I. INTRODUCTION

In the current world scenario, the need of people to use the technology is gradually increasing and almost all of this technology is using the internet as communication media. The internet can be of a very great size that a single routing protocol cannot rule the roost the functionalism of refurbishing the routing inventory of every router. In order to overcome this problem, the internet is fractionated into a self-determining that is, a self-governing system.

In general, protocols that are configured on routers with the purpose of exchanging routing information in which the routing algorithm functions inside only within the dominions is called as intra dominion routing and the divulgation joining the self-determined operational structures is called as inter-dominion routing. The protocols used in intra dominion(commonly routers) routing are known as Interior-gateway protocols.

Revised Manuscript Received on February 28, 2020.
* Correspondence Author

Pooja Ravi*, Department of Computer Science, RN Shetty Institute of Technology, Channasandra, Bangalore, India. E-mail-id:ravibsu@gmail.com

Pragna B Rao, Department of Information Science & Engineering, RN Shetty Institute of Technology, Channasandra, Bangalore, India. E-mail-id:balajiraopragna@gmail.com

Some commonly used protocols for the intra domain routing are RIP (resource information protocol) and OSPF (open shortest path first).The Routing Information Protocol is a effectual routing protocol which utilizes the hop count as a routing criterion to perceive the optimal path allying the source and destination network and the Open Shortest Path First is a linkage routing protocol which is used to detect the ideal path allying the genesis(source) and terminus(destination) router using its own closest path.

The BELLMAN-FORD and the DIKSTRA'S algorithms are the widely known algorithms utilized in the intra dominion routing to amend the routing inventories.

II. BELLMAN-FORD ALGORITHM

The BELLMAN-FORD algorithm is used to perceive the closest path in a weighted digraph (where the visual representation(graph) can have unfavorable/negative edges). The algorithm even detects the extant of an edge with negative weight. The design and composition of Bellman-Ford is analogous to that of Dijkstra's algorithm, but as a replacement of intemperately selecting the least-weighted node not yet prepared to qualify. The algorithm proceeds by qualification, in which the rough calculations to rectify the distance are restored by a recommended distance up until they finally in the fullness of time set foot on(reach) the explication. The algorithm follows the dynamic programming approach, in which the extent from the source node to all the other neighbors if found and chooses the shortest path in that and using that node tries to reach the final node thus, reaching the destination node. The algorithm simply qualifies all the edges, and does this |V|-1 times, where |V| is the quota of vertices in the graph.

A. Procedure

Step 1: Assign the extent from genesis(source) to each and every apex(vertex) as infinite and distance to genesis (source) itself as 0. Generate an array ext [] of size |A| with all the values as infinite except ext [gen] where gen is the genesis(source) apex(vertex).

Step 2: Compute the closest distances. Do following |a|-1 times where |a is the number of apices(vertices) in given visual representation(graph).

Step a: Do the subsequent for each edge u-a

If ext[a]>ext[u] + weight of edge ua, then streamline ext[a]

ext[a] =ext[a] + weight of edge uv

Step 3: Promulgate if there is a negative sequenced cycle in the visual representation(graph). Do the following for each edge u-v

If ext[a]>ext[u] + weight of edge uv, then “Visual representation (graph) contains negative weight sequenced cycle”

III. DIJKSTRA’S ALGORITHM

The DIJKSTRA’S algorithm, was devised by Dutch computer boffin Edgser Dijkstra in 1959, is a visual representation(graph) search algorithm used to detect the lone source closest path for a graph with positive edges. This algorithm is handed-down in routing protocols mostly significantly transitional system to intermediate system and open shortest path first. It is also exerted as a subprogram in supplementary algorithms like Johnson’s algorithm. For a distinct genesis(source) node in the visual representation, the algorithm detects the closest path (lowest cost path) connecting the node and every possible node in the visual representation(graph). The algorithm can also be utilized to find the shortest path from single source to single terminus(destination) by terminating the algorithm once the shortest path to the terminus(destination) node has been determined.

A. Procedure

Step1: Generate set cptSet (closest path tree set) that keeps track of apices (vertices) comprehended in closest path tree, i.e., whose minimal distance from source(genesis) is calculated and finalized. Initially, this set is empty. **Step2:** Allocate a extent value to all apices(vertices) in the input visual representation(graph). Initialize all extent values as INFINITUDE. Allocate extent value as 0 for the genesis(source) apex vertex so that it is picked first. **Step3:** While cptSet doesn’t comprehend all apogee (vertex)**Step a:** Pick a apex u which is not there in *cptSet* and has minimum distance value.

Step b: Include u to cptSet.

Step c: Update distance value of all adjacent apices of u. To update the extent values, iterate through all adjacent vertices. For every adjacent apex a, if sum of extengt value of u (from genesis) and weight of edge u-a, is less than the extent value of a, then update the extent value of a.

IV. ALGORITHM

A. Bellman-Ford Algorithm

Input: Edges edges[], int edgecalculate[], int nodecalculate, int genesis, int terminus

Output: Routing table

```
{
        int *extent;    // Should be allocated
        int i, j;
        if (extent== NULL) then{
    printf (stderr, "malloc () failed\n");
        exit (EXIT_FAILURE);
        }
        for (i← 0; i < nodecalculate; ++i)
          extent[i] = INFINITUDE;
        extent[genesis] =0;
        for (i←0; i < nodecalculate; ++i){
           for (j←0; j < edgecalculate; ++j){
                   if (extent [edges[j]. genesis]!=
        INFINITY){
                        int new_extent←distance[edges[j].
        genesis] +edges[j]. weight;
                              if (new_extent <
                extent[edges[j]. terminus])
                                   extent[edges[j].
                      terminus] = new_extent;
                   }
              }
        }
        for (i= 0; i < edgecalculate; ++i){
           if
           (extent[edges[i].terminus]>extent[edges[i].gene
           sis] edges[i].weight){
                     puts ("Negative edge weight cycles
        detected!");
                     free (extent);
                      return;
              }
}
```

B. Dijsktra Algorithm

```
Dijkstra(Visual Representation, genesis){
  Generate apogee set Q
  for each apogee a in Visual Representation{
    ext[v]←INFINITUDE
    prior[v]←UNDETERMINED
    add up  a to Q
  }
  ext[genesis]←0
  while Q is not vacant{
        u←vertex in Q with min ext[u]
    detach u from Q
    for every adjacent v of u{
      alt←ext[u]+stretch(u,a)
      if alt<ext[v]{
        ext[v]←alt
        prior[v]←u
      }
    }
  }
  return ext[],prior[]
}
```

V. TIME COMPLEXITY

Algorithm	Time Complexity
Bellman-Ford	O(VE)
Dijkstra	O(V^2) using linear array for priority queue O((V+E)logV) using binary heap O(VlogV+E)using fibonacci heap

VI. RESULT

Results from two algorithms agree –

A. Bellman-Ford

1. Calculation for node n needs link cost to neighbor nodes plus total cost to each neighbor from vertex.
2. Each node can maintain set of costs and paths for every other node
3. Can exchange information with direct neighbors.
4. Can update costs and paths based on information from neighbors and knowledge of link costs.

B. DIJKSTRA

1. Each node needs complete topology.
2. Must know link costs of all links in network.
3. Must exchange information with other nodes.

VII. CONCLUSION

As the analysis shows the Dijkstra's algorithm has a lower running time than that of Bellman-ford for a same problem, but requires the edges to be non-negative. Thus, Bellman-Ford is usually used only when there are negative edges. Both of these algorithms solve the single source shortest path problem. The major difference between the two algorithms is that Bellman-Ford can handle negative edges whereas Dijkstra cannot handle negative edges. But, it is to be remembered that when there is a negative cycle present in the graph then there is no shortest path.

REFERENCES

1. en.wikipedia.org/
2. Jin Y.Yen. "An algorithm for finding shortest routes from all source notes to given destination in general network", Quart.Appl.Math., 27,1970,526-530.
3. Thomas H.Cormen, Charles E.Leiserson, Ronalad L.Rivest and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 24.1: The Bellman-Food algorithm, pp.588-592. Problem 24-1, pp.614-615.
4. "A Study on Contrast and Comparison between Bellman-Ford algorithm and Dikstra's algorithm" by Thippeswamy.K, Hanumanthappa.J, Dr.Manjaiah D.H.
5. Geeks for Geeks

AUTHORS PROFILE

Pooja Ravi is a pursuing undergraduate course in computer science at RN Shetty Institute of Technology ISE Department. She completed her primary education from Jyothy Kendriya Vidyalaya, Bangalore in the year 2015.She then did her pre-univeristy education from Sri Kumaran Pre-University College, Bangalore in the year 2015-2017. She will graduate in the year 2021.Fields of Interest: Artificial Intelligence, Machine Learning, Data Science.
Email-id:ravibsu@gmail.com

Pragna B Rao is a student at RN Shetty Institute of Technology, currently pursuing undergrad in Bachelors of Information Science & Engineering. She did her schooling at Carmel, Bangalore followed by pre-university education at Sri Kumaran Pre-University college. She will graduate in the year 2021. She's an inquisitive person, enjoy learning, and is passionate about Data Science, Machine Learning, Software Development and Technology Management. Email-id:balajiraopragna@gmail.com