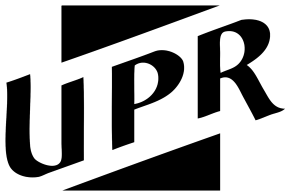
Universidad de Puerto Rico
Recinto Universitario de Mayagüez
Computer Science and Engineering

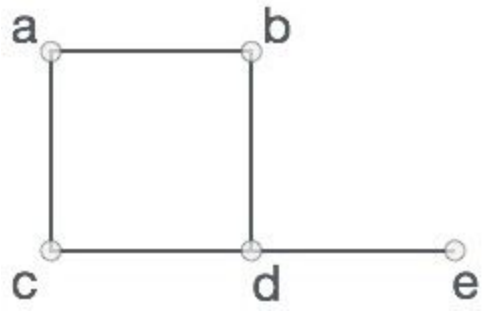


Shortest Path Routing:

Francisco Diaz
802-15-2172
Prof. Kejie Lu
Computer Science and Engineering

- Table of content
- Section 1: Introduction
    - Overview of the project
    - Link to youtube video.
    - Outline of the rest of this report
- Section 2: Basics of the shortest path routing problem
    - What is a graph (data structure)?
    - How to represent a network using a graph?
    - What is the shortest path routing problem?
- Section 3: Formulation of the shortest path routing problem
    - Define all necessary variables
    - Define an objective function
    - Define all constraints
    - This section will be evaluated using ABET performance indicator (1).
- Section 4: Shortest Path Algorithms
    - What is the Dijkstra algorithm?
    - What is the Floyd-Warshall algorithm?
    - What is the Bellman-Ford algorithm?
    - What is the algorithm you want to implement in this project?
    - Why did you choose this algorithm?
    - This section will be evaluated using ABET performance indicator (2) and indicator (3).
- Section 5: Implementation of a shortest path routing algorithm
    - The pseudo code of a shortest path algorithm
        1. The inputs include the network, the source nodes, and the destination node.
        2. The output is a sequence of nodes 0n the shortest path
    - The complexity analysis of the algorithm using big-O notation
    - This section will be evaluated using ABET performance indicator (3).
- Section 6: Conclusions
- References
    - Need **at least 15 references** for relevant software, standards, webpages, videos, research papers, etc.

I. Link to video
   A. https://www.youtube.com/watch?v=J51a-JkZ9lM&feature=em-lsp
II. Basics of the shortest path routing problem
   A. What is a graph data structure
      1. A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges. Formally, a graph is a pair of sets (V, E), where V is the set of vertices and E is the set of edges, connecting the pairs of vertices. Take a look at the following graph −



In the above graph, V = {a, b, c, d, e}, E = {ab, ac, bd, cd, de}

   B. How to represent a network using a graph?
      1. Adjacency matrix: a matrix where the columns and rows are vertices and an entry in the matrix represents the edge weight between a vertices (u,v)
      2. Adjacency list: a collection of lists where each list represents a node and the values of the list represents the edge weight between itself and the other node in the corresponding entry of the list.
   C. What is the shortest path problem
      1. Given a graph G=(V,A) with arc costs given by the matrix C=[c_{ij}], the shortest path problem is the problem of finding the shortest path from a specific starting vertex s∈V to a specific ending vertex t∈V, provided that such a path exists. c_{ij} can be positive, negative or zero, provided that no circuit of G exists whose total cost is negative. [2]

III. Formulation of the shortest path routing problem
   A. Formulation of the shortest path routing problem

1. When it comes to sending information through the network, we need to find the shortest path to send data between corresponding in order to minimize the travel time of each packet. For that we keep a table of a precomputed shortest path between nodes such that when a packet has to be sent we look at the entry for that node, know where to send the packet and that this location is optimal.

B. Define all necessary variables
   1. A graph is a data structure that contains the information of vertices, edges, and their respective weights between edges.
   2. A distance vector or map that associates a vertex and its distance to the source.
   3. A previous node vector that contains information of the nodes and how they connect
   4. A source node where we want to start

C. Define the objective function.
   1. The objective is to find the shortest distance between nodes u, v on all nodes in the collection Q.

```
alt ← dist[u] + length(u, v)
if alt < dist[v]:
    dist[v] ← alt
    prev[v] ← u
```

D. Define all constraints.
   1. For Dijkstra's Algorithm: $\forall e \in E, \{e \mid e \geq 0\}$
   2. For Bellman Ford it cannot contain a negative weight cycle

IV. Shortest Path Algorithms

A. What is the Dijkstra algorithm?

1. Given a graph and a source vertex in the graph, find shortest paths from source to all vertices in the given graph. Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a shortest path tree with a given source as root. We maintain two sets, one set contains vertices included in the shortest path tree, the other set includes vertices not yet included in the shortest path tree. At every step of

the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.[3]

2. Time complexity:
    a) Array $O(V^2)$ [13}
    b) Min Priority Queue $O(V + E\ log(V))$ [13]

```
1  function Dijkstra(Graph, source):
2
3      create vertex set Q
4
5      for each vertex v in Graph:
6          dist[v] ← INFINITY
7          prev[v] ← UNDEFINED
8          add v to Q
10     dist[source] ← 0
11
12     while Q is not empty:
13         u ← vertex in Q with min dist[u]
14
15         remove u from Q
16
17         for each neighbor v of u:           // only v that are still in Q
18             alt ← dist[u] + length(u, v)
19             if alt < dist[v]:
20                 dist[v] ← alt
21                 prev[v] ← u
22
23     return dist[], prev[]
```

B. What is the Floyd-Warshall algorithm?
    1. Floyd Warshall is an algorithm for finding shortest paths in a weighted graph with positive or negative edge weights, but with no negative cycles. A single execution of the algorithm will find the lengths, summed weights, of shortest paths between all pairs of vertices. Although it does not return details of the paths themselves, it is possible to reconstruct the paths with simple modifications to the algorithm.[5]
    2. Floy Warshall time complexity:
        a) $O(V^3)$ [13]

1. **function** floyd-warshall(Graph,source):

```
2.  let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
3.  for each edge (u, v) do
4.      dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
5.  for each vertex v do
6.      dist[v][v] ← 0
7.  for k from 1 to |V|
8.      for i from 1 to |V|
9.          for j from 1 to |V|
10.             if dist[i][j] > dist[i][k] + dist[k][j]
11.                 dist[i][j] ← dist[i][k] + dist[k][j]
12.             end if
13.
```

C. What is the Bellman-Ford algorithm?

1. Bellman Ford algorithm works by overestimating the length of the path from the starting vertex to all other vertices. Then it iteratively relaxes those estimates by finding new paths that are shorter than the previously overestimated paths. By doing this repeatedly for all vertices, we can guarantee that the result is optimized.[6]
2. Bellman Ford time complexity:
    a) $O(V^2E)$ [13]

```
1.  function BellmanFord(list vertices, list edges, vertex source)
2.      distance[], predecessor[]
3.      for each vertex v in vertices do
4.          distance[v] := inf
5.          predecessor[v] := null
6.      distance[source] := 0
7.      for i from 1 to size(vertices)−1 do
8.          for each edge (u, v) with weight w in edges do
9.              if distance[u] + w < distance[v] then
10.                 distance[v] := distance[u] + w
11.                 predecessor[v] := u
12.
13.     for each edge (u, v) with weight w in edges do
14.         if distance[u] + w < distance[v] then
15.             error "Graph contains a negative-weight cycle"
16.     return distance[], predecessor[]
```

D. What is the algorithm you choose to implement in this project
    1. Dijkstra's Algorithm
E. Why did you choose this algorithm?
    1. Disjktra's algorithm is the fastest of the three with the only constraint that there cannot be negative edges. In the case of routing this is a reasonable assumption since euclidean distances cannot be negative.

V. Implementation
    A. See notebook

VI. Conclusion
    A. The shortest path routing problem helps modern computer networks send information with the shortest distance possible. This helps send information faster, which in turn helps in the overall performance and reliability of the network.

References

1. http://www.eisber.net/unidocs/Algorithmen%20auf%20Graphen/Shortest_paths.pdf
2. https://www.tutorialspoint.com/data_structures_algorithms/graph_data_structure.htm
3. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
4. https://www.programiz.com/dsa/bellman-ford-algorithm
5. https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
6. https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
7. https://www.geeksforgeeks.org/shortest-path-problem-between-routing-terminals-implementation-in-python/
8. https://www.youtube.com/watch?v=-pFYJuJvqPY
9. https://www.youtube.com/watch?v=wVu5qN0E4ww
10. https://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf
11. https://www.codingame.com/playgrounds/1608/shortest-paths-with-dijkstras-algorithm/dijkstras-algorithm
12. https://medium.com/basecs/finding-the-shortest-path-with-a-little-help-from-dijkstra-613149fbdc8e
13. https://cs.stackexchange.com/questions/2942/am-i-right-about-the-differences-between-floyd-warshall-dijkstra-and-bellman-fo
14. https://en.wikipedia.org/wiki/Routing
15. https://www.researchgate.net/publication/323411017_Minimizing_Flow_Completion_Times_using_Adaptive_Routing_over_Inter-Datacenter_Wide_Area_Networks