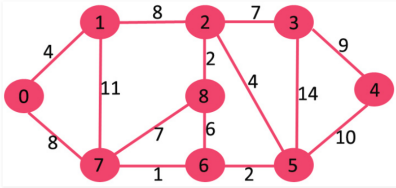
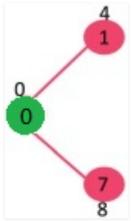
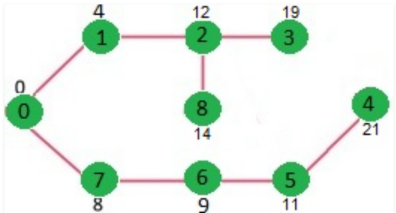


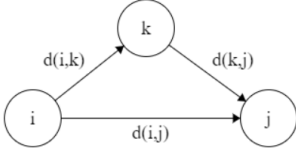
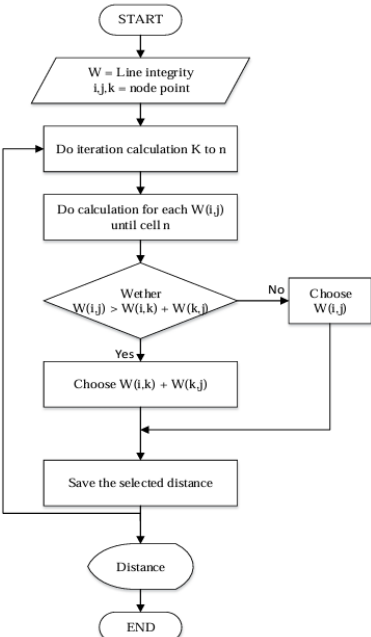
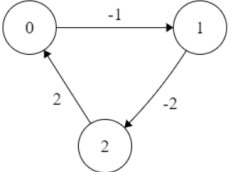
Assignment Computational Complexity

Group member: Tang Kuan Yew A18CS0261
 Toon Shu Hui A18CS0268
 Pang Wen Jie A18CS0232

| No | Paper | Application | Algorithm | Advantages | Disadvantages | What problem can be solved and what problem cannot be solved using this algorithm |
|----|---|--|---|--|--|---|
| 1 | <p>Title: Comparative Study On Bellman-Ford And Dijkstra Algorithms</p> <p>Authors: Fadhil Mukhlif, Abdu Saif</p> <p>Year published: 2020</p> <p>Published in: ResearchGate</p> | <ol style="list-style-type: none"> Traffic information systems use Dijkstra's algorithm in order to track the source and destinations from a given particular source and destination Open Shortest Path First (OSPF) used in Internet routing. Actually it uses a link-state in the individual areas that make up the hierarchy. However, the computation is based on Dijkstra's algorithm which is used to calculate the shortest path tree inside each area of the network | <ol style="list-style-type: none"> Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest-path tree. Initially, this set is empty. Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first. While sptSet doesn't include all vertices: <ol style="list-style-type: none"> Pick a vertex u which is not there in sptSet and has a minimum distance value. Include u to sptSet. Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v. <p>Example:</p> | <ol style="list-style-type: none"> Little complexity which is almost linear More efficient when the shortest path from one node to all other nodes needed to be calculated | <ol style="list-style-type: none"> Time consuming as it does a blind search Cannot handle negative edges where this leads to acyclic graphs and most often cannot obtain the right shortest path | <p>Problem can be solved:</p> <ol style="list-style-type: none"> Calculate the shortest path between a single node to all other nodes and a single source node to a single destination node by stopping the algorithm once the shortest distance is achieved for the destination node <p>Problem cannot be solved:</p> <ol style="list-style-type: none"> When the path is in a arc (not linear) Non applicable when edges have negative value |

| | | | | | | |
|---|---|--|---|---|---|--|
| | | |  <p>1. Initial: SptSet {} Distance assigned to edges {0, INF, INF, INF, INF, INF, INF, INF, INF}</p> <p>2. Pick the vertex with a minimum distance value. The vertex 0 is picked. SptSet {0}</p>  <p>3. SptSet {0, 1, 7} Distance assigned to edges {0, 4, 8, INF, INF, INF, INF, INF}</p>  <p>4. SptSet {0, 1, 7, 6, 5, 2, 8, 19, 21} Distance assigned to edges {0, 4, 8, 9, 11, 12, 14, 19, 21}</p> | | | |
| 2 | Title: Bellman–Ford algorithm for | Trapezoidal Picture Fuzzy Graph (TPZG) | Bellman-Ford Algorithm (BA) | 1. Able to operate on PFDGs with negative | 1. Large complexity and high time complexity: | Problem that can be solved: 1. Calculate shortest path with |

| | | | | | | |
|--|---|--|--|--|--|---|
| | <p>solving shortest path problem of a network under picture fuzzy environment</p> <p>Authors: Mani Parimala, Said Broumi, Karthikeyan Prakash, Selçuk Topal</p> <p>Year published: 2021</p> <p>Published in: Complex & Intelligent Systems (Volume 7) pp 2373-2381 by Springer</p> | <p>(for determining the transmission distance of multifunction sensors in computer electronics and wireless communication)</p> | <p>Pseudo Code:</p> <ol style="list-style-type: none"> Function of TPZG BA (G,s) $g(\alpha) = \begin{cases} \min_{\alpha < \beta} [g(\alpha) + d_{\alpha\beta}], & \text{otherwise} \\ 0, & \text{if } \alpha = 1 \end{cases}$ <ol style="list-style-type: none"> nranks[s] = 0 Ndist[s] = Empty Picture Fuzzy Number Add s into R (Relax(u,v)) For every node of i (excluding s) Rank [i] = infinity Add i into R End for Origin Node (u) = End Node (s) While Rank is not empty Eliminate vertex u from R For each adjacent vertex v from u Relaxed = false, temp_ndist[v] = ndist[u] + edge weight of {u,v} temp_nrank[v] = rank{temp_ndist[v]} If temp_nrank[v] < nrank[v] then nrank[v] = temp_nrank[v] ndist[v] = temp_ndist[v] prev[v] = u End if End for u = node in R with minimum rank value End while The PFN ndist[u] is the SP from the ON s to EN u <p>For n vertices, we relax the edges for n-1 times where n is the number of edges.</p> <p>Relax(u,v): if distance(v)>distance(u)+cost(u,v) : distance(v)=distance(u)+cost(u,v) Where u and v are edges.</p> <p>Step-by-step Algorithm:</p> <ol style="list-style-type: none"> Initialize distance from the source to destination as infinity. Select any two edges and make the list of edges. Relax the edges until the shortest | <p>weight arc length which cannot be dealt with the Dijkstra algorithm</p> <ol style="list-style-type: none"> Can be readily put in and extended to other fuzzy networks with the arc weight, such as: <ul style="list-style-type: none"> bipolar picture fuzzy numbers cubic picture fuzzy numbers interval bipolar picture fuzzy numbers single-valued picture fuzzy numbers trapezoidal picture fuzzy numbers triangular picture fuzzy numbers pentagonal fuzzy numbers and so on. | <p>O(VE)</p> <ol style="list-style-type: none"> When there is a network update, a slow response will be observed as this shift will propagate node-by-node. If the network contains routing loops, then there exists a node failure. | <p>non-negative weights for both directed and undirected graphs</p> <ol style="list-style-type: none"> Calculate shortest path of containing negative weights for directed graphs <p>Problem that cannot be solved:</p> <ol style="list-style-type: none"> When the graph with negative weights is undirected |
|--|---|--|--|--|--|---|

| | | | | | | |
|---|--|---|---|---|---|--|
| | | | distance starts repeating itself or in the worst-case scenario n-1 times. | | | |
| 3 | <p>Title: Floyd-warshall algorithm to determine the shortest path based on android</p> <p>Authors: Ramadiani, D Bukhori, Azainil, N Dengen</p> <p>Year published: 2018</p> <p>Published in: IOP Publishing Ltd</p> | Geographic Information System (GIS) that is capable of storing or documenting spatial information in digital form so that it can be used for navigation, GPS, more specific place/location search | <p>Floyd-Warshall Algorithm Steps:</p> <ol style="list-style-type: none"> 1. Delete all loop paths 2. Erase the parallel path and leave with the shortest-weighted path 3. Create a matrix with the Distance Table 4. Iterate $\min(d[i, j], d[i, k] + d[k, j])$   <p>Pseudo-code: for i = 1 to n do</p> | <p>Weight of the edge can be negative</p> <p>Return the results for all pairs of vertices in the form of matrix</p> | <p>Time Complexity: $O(V^3)$</p> <p>Space Complexity: $O(V^2)$ Where V = number of Vertices</p> <p>Graph cannot contains negative cycle</p>  | <p>Problem that can be solved:</p> <ul style="list-style-type: none"> - Find the shortest path of all possible pairs of vertices in any directed graph with positive cycle <p>Problem that cannot be solved:</p> <ul style="list-style-type: none"> - Find the shortest path with a graph that contains a negative cycle (requires Bellman-Ford algorithm to detect negative cycles) |

```

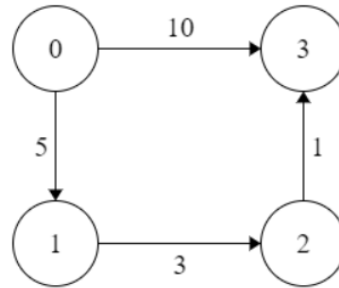
for j = 1 to n do
  if there exists an edge from i to j then
    d[i, j] = w[i, j]
    r[i, j] = i
  else
    d[i, j] = infinity
    r[i, j] = -1

```

```

for k = 1 to n do
  for i = 1 to n do
    for j = 1 to n do
      if d[i, k] + d[k, j] < d[i, j] then
        d[i, j] = d[i, k] + d[k, j]
        r[i, j] = k

```



| | | | |
|-----|-----|-----|-----|
| 0 | 5 | INF | 10 |
| INF | 0 | 3 | INF |
| INF | INF | 0 | 1 |
| INF | INF | INF | 0 |

| | | | |
|-----|-----|-----|---|
| 0 | 5 | 8 | 9 |
| INF | 0 | 3 | 4 |
| INF | INF | 0 | 1 |
| INF | INF | INF | 0 |