FINDING THE SHORTEST PATH USING THREE DIFFERENT ALGORITHMS

CMSC 142

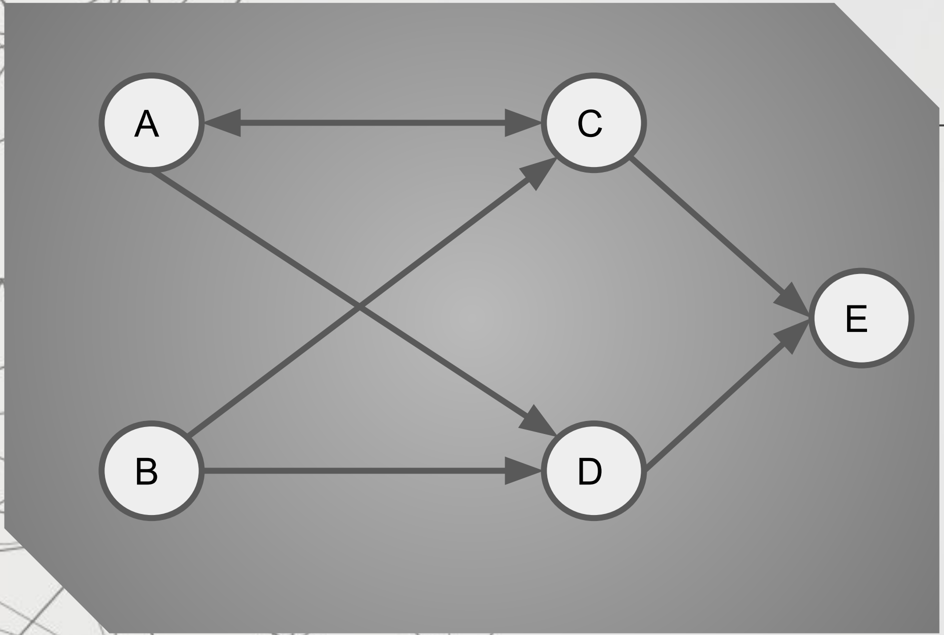
Lolibeth Domer

TABLE OF CONTENTS

01

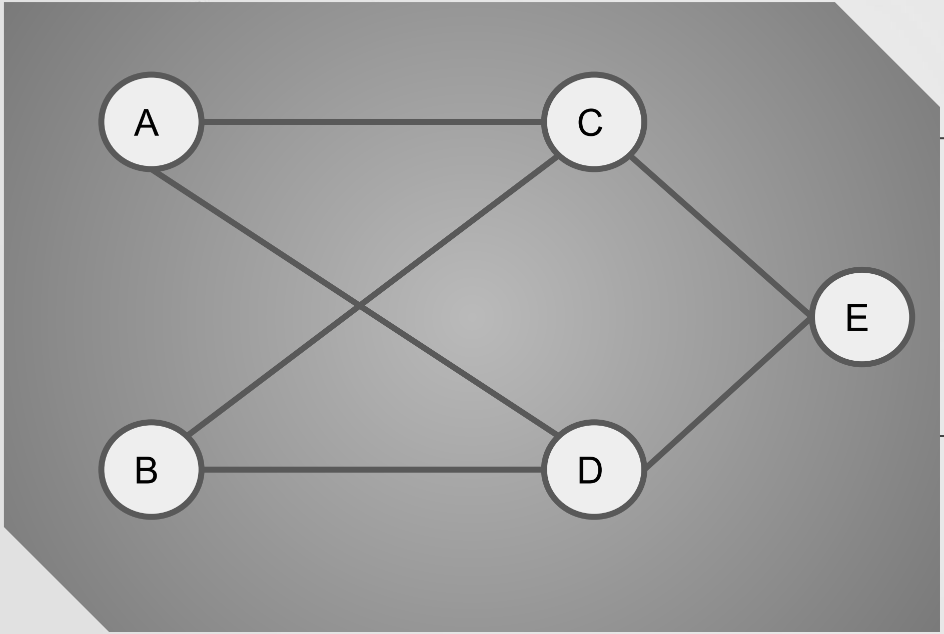
INTRODUCTION

Graph, Vertex, Edges, Undirected, Unweighted, etc.



GRAPH

It is a mathematical representation of a network and describes the relationship between lines and points.



Vertex

Vertex or node is a point in the graph represented by a circle.

Edge

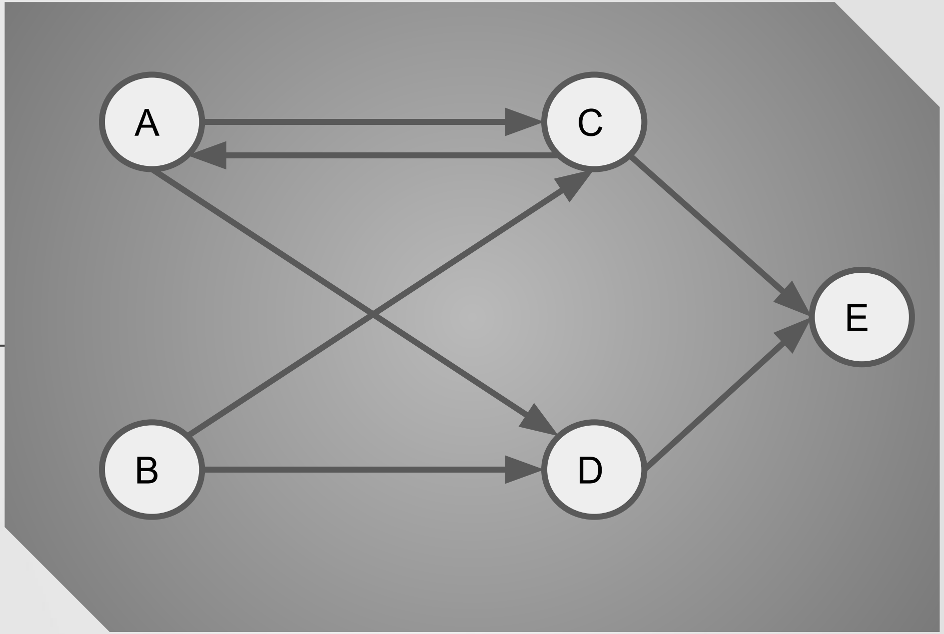
Edge is a line that serves as a link that connects the vertices together.

TYPE OF GRAPHS

Directed
VS.
Undirected

Weighted
VS.
Unweighted

DIRECTED VS. UNDIRECTED



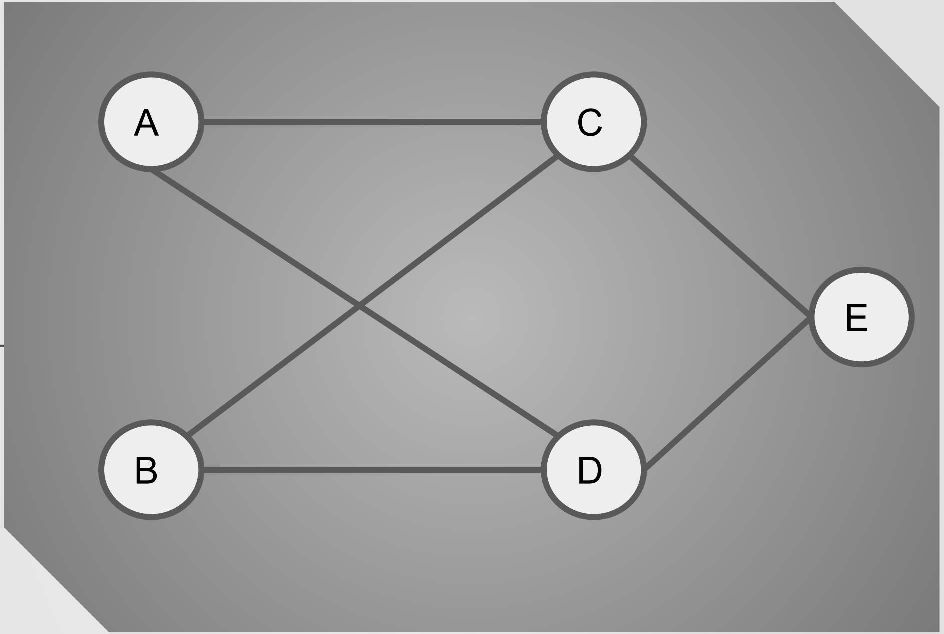
Directed Graph

It consists of edges or links of vertices that represents a one way relationship.

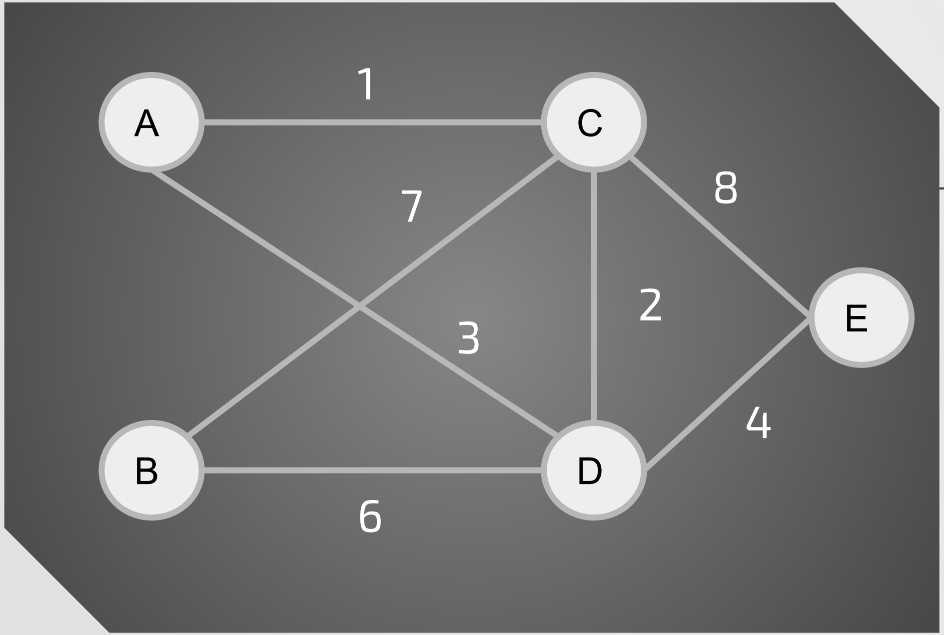
DIRECTED VS. UNDIRECTED

Undirected Graph

It consists of edges that are bidirectional, or a relationship from a node to another node and backwards.



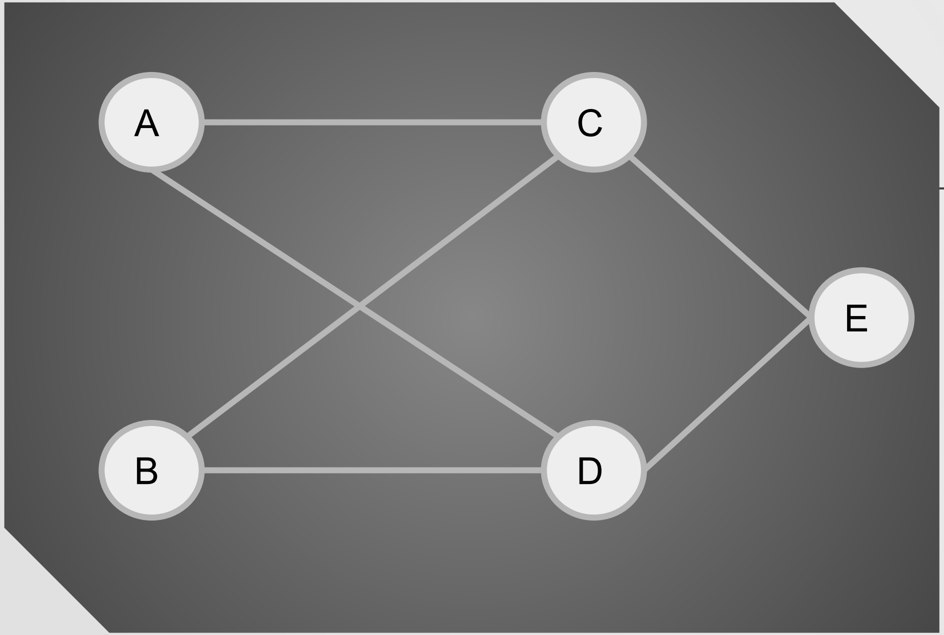
WEIGHTED VS. UNWEIGHTED



Weighted Graph

It is a graph whose edges have been labeled with weights or numbers.

WEIGHTED VS. UNWEIGHTED



Unweighted Graph

It is a graph with no weight or no numerical values attached.

Source is V(A)
Target is V(E)

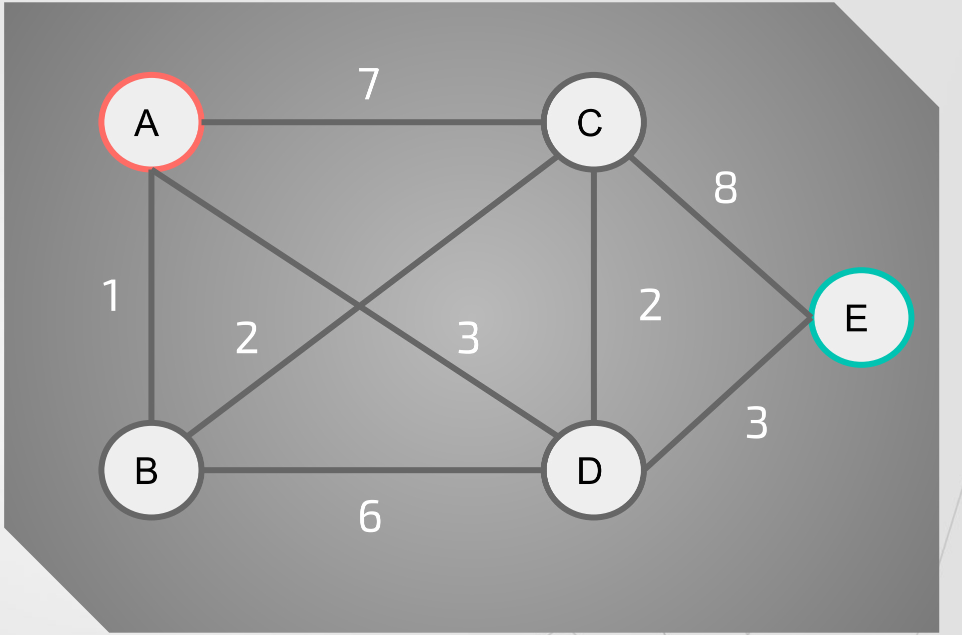
E(A,C) → E(C,E) ; 7 + 8 = **15**

E(A,B) → E(B,D) → E(D,E) ; 1 + 6 + 3 = **10**;

E(A,B) → E(B,C) → E(C,D) → (D,E) ; 1 + 2 + 2 + 3 = **8**

E(A,D)→ E(D,E) ; 3 + 3 = **6**

E(A,B) → E(B,C) → E(C,E); 1 + 2 + 8 = **11**



Finding the Shortest Path

02

Algorithms

Dijkstra’s Algorithm ; Bellman-Ford Algorithm;
Floyd-Warshall Algorithm

Algorithms for Finding the Shortest Path

Dijkstra's Algorithm

Created and published by Dr. Edsgar Dijkstra in 1959.

Proposed by Alfonso Shimbell in 1955. But named Richard Bellman and Lester Ford Jr.

Bellman-Ford Algorithm

Floyd-Warshall Algorithm

Proposed by Robert Floyd and Stephen Warshall in the same year 1962

DIJKSTRA'S ALGORITHM

It is an algorithm for finding the shortest path with a minimum costs between nodes in any weighted graph with nonnegative weights. It was created and published by a computer scientist and software engineer, Dr. Edsger Dijkstra in 1959.

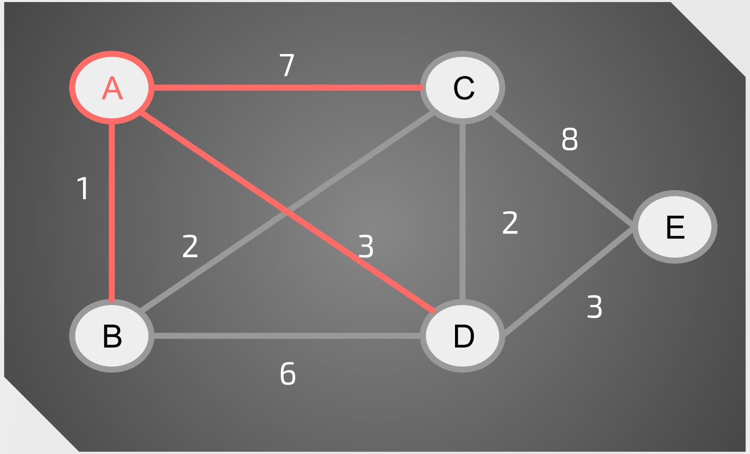
Time Complexity: O(E log V) or O (V^2)
Space Complexity: O(V);

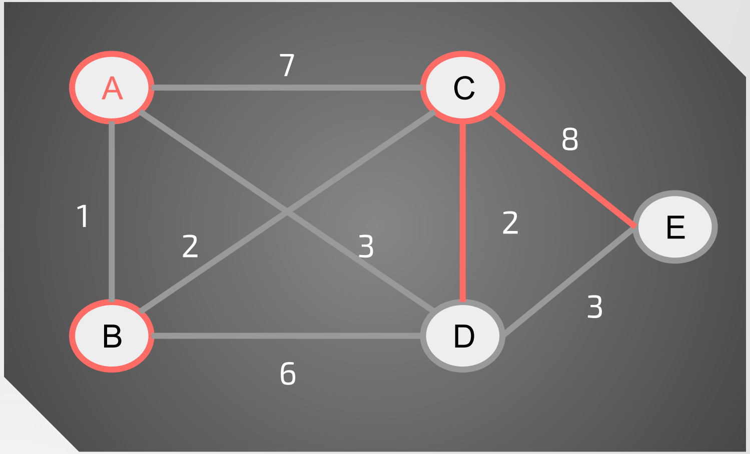
DIJKSTRA'S ALGORITHM

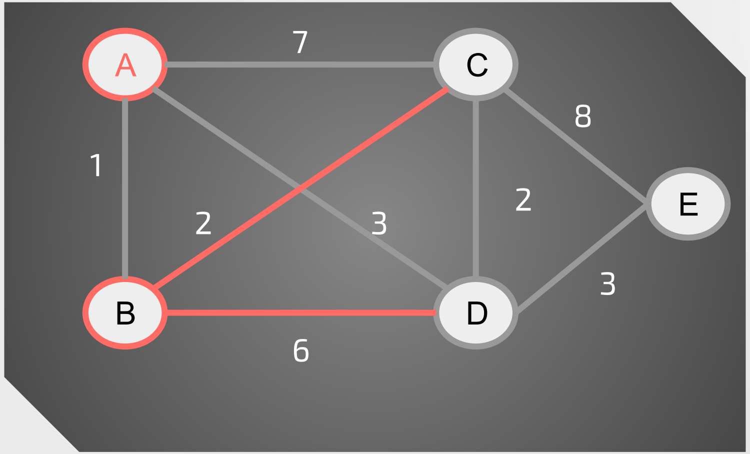
1. **Initialize** graph[n][n], distance [n], prevVertex[n], visitedVertex[n], int source;
2. **Calculation of the temporary distances** of all neighbour nodes of the active vertex by summing up the distance with the weights of the edges.
3. If the calculated distance of a node is smaller as the current one, **update the distance** and set the current node as antecessor.
4. **Setting of the node** with the minimal temporary distance as active and set the previous node in a prevVertex list.
5. **Repeating of steps 4 to 7** until there aren't any nodes left with a permanent distance, which neighbours still have temporary distances.

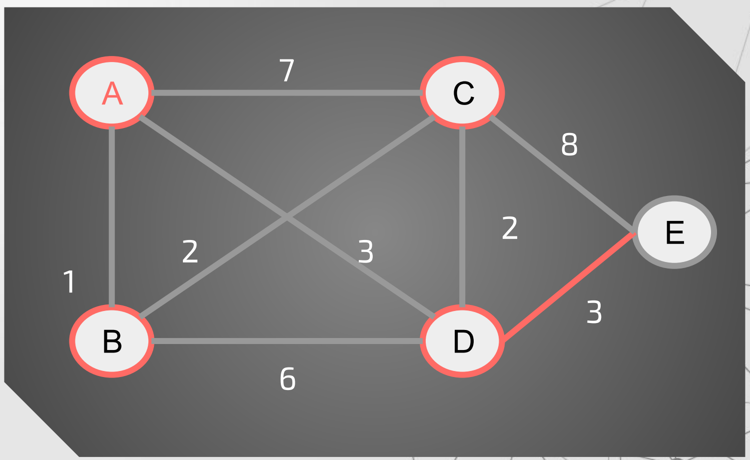
DIJKSTRA'S ALGORITHM

```
for (int i = 0; i < count; i++) {
    // Update the distance between neighbouring vertex and source vertex
    int u = findMinDistance(distance, visitedVertex);
    visitedVertex[u] = true;

    // Update all the neighbouring vertex distances
     for (int v = 0; v < count; v++) {

          if (!visitedVertex[v] && graph[u][v] != 0 && (distance[u] + graph[u][v] < distance[v])) {
               int temp_d=distance[v];
               distance[v] = distance[u] + graph[u][v];

              //Check the if the current cost is less than the stored cost
               if (v==0) prevVertex[0]=0;
               else if (temp_d>distance[v])
                    prevVertex[v]=u;
```

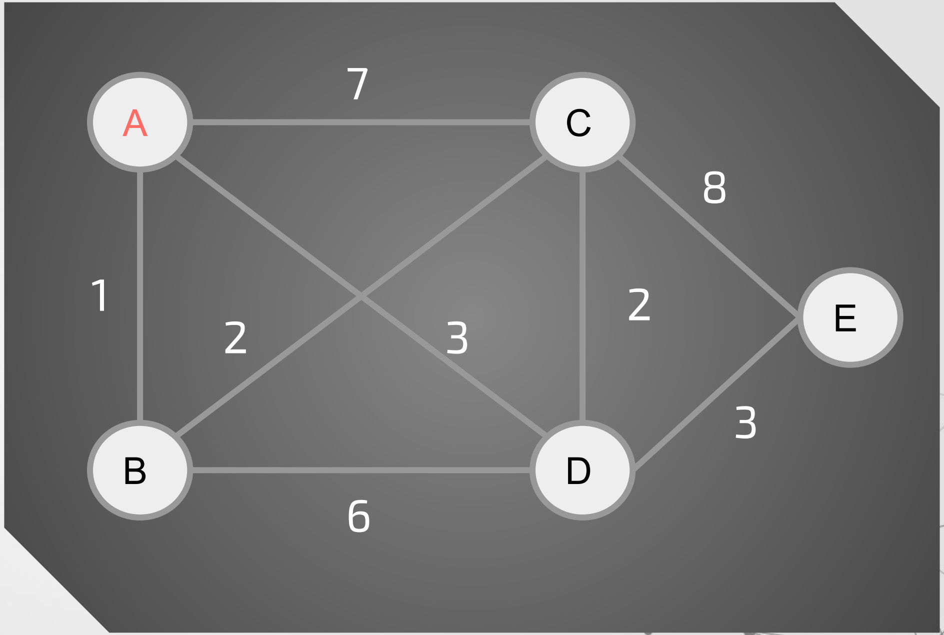

A
7
C
1
2
3
2
8
E
3
B
D
6


A
7
C
1
2
3
2
8
E
3
B
D
6

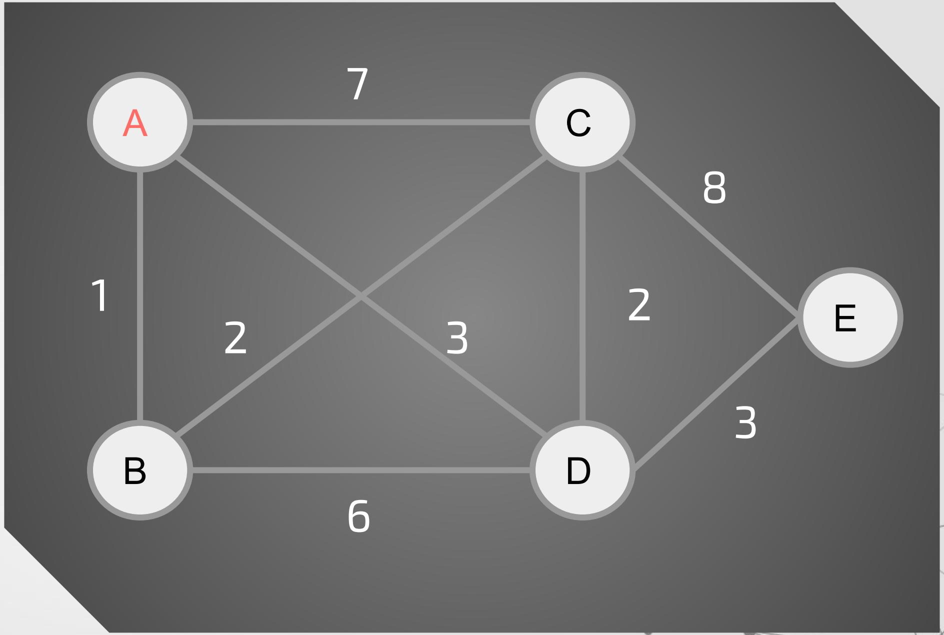

A
7
C
1
2
3
2
8
E
3
B
D
6


A
7
C
1
2
3
2
8
E
3
B
D
6

Vertex	Cost/ Distance	Previous Vertex
A	∞	
B	∞	
C	∞	
D	∞	
E	∞	

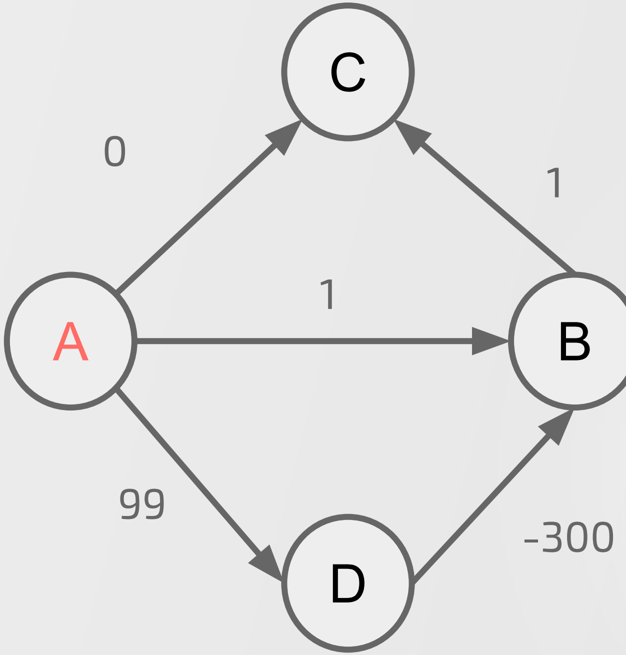


Vertex	Cost	Previous Vertex
A	0	None
B	1	A
C	3	B
D	3	A
E	6	D



SHORTEST PATH: v(A) ⟶ v(D) ⟶ v(E) or e(A,D) ⟶ e(D,E)

Dijkstra’s Algorithm does not work with negative-weighted graph.



Vertex	Cost/Distan ce	Previous Vertex
A	0	none
B	1	A
C	0	A
D	99	A

Updating Vertex B? 1>-201

BELLMAN-FORD'S ALGORITHM

It is also a single-source shortest path algorithm, similar to Dijkstra's Algorithm. However, Bellman-Ford can work on graphs with negative- weighted edges.This was proposed by Alfonso Shimbell in 1955, however it was named after Richard Bellman and Lester Ford Jr.

Time Complexity: O(VE)
Space Complexity: O(V);

BELLMAN-FORD'S ALGORITHM

1. **Initializes** distances from the source to all vertices as infinite and distance to the source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.
2. This step **calculates shortest distances**. Do following **|V|-1 times** where |V| is the number of vertices in given graph.
    a. Do following for each edge u-v

       If dist[v] > dist[u] + weight of edge uv, then update dist[v]

       dist[v] = dist[u] + weight of edge uv
3. Loop **reports** if there is a negative weight cycle in graph. Do following for each edge u-v

   If dist[v] > dist[u] + weight of edge uv, then "Graph contains negative weight cycle"

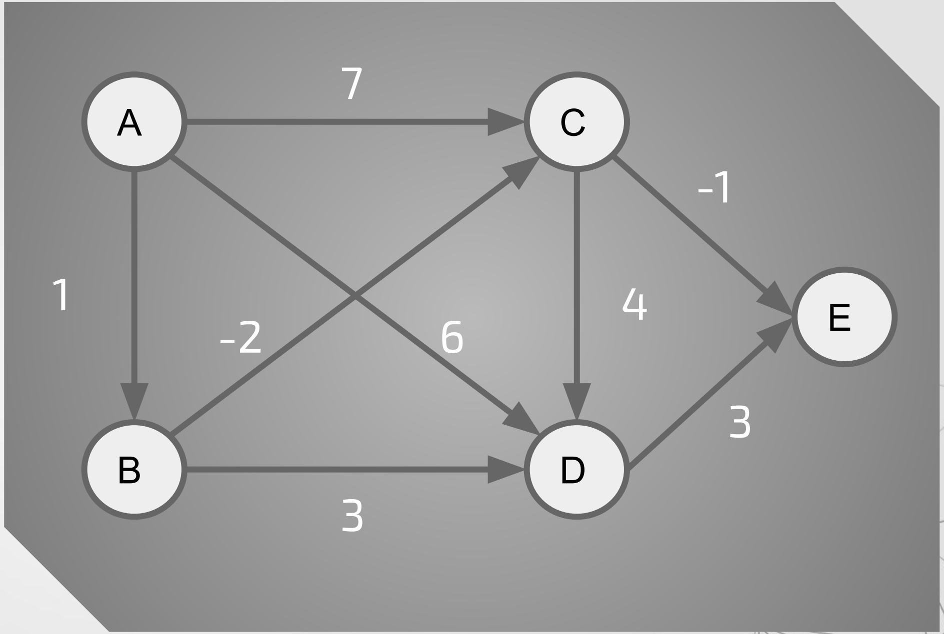
Note :3 Guarantees shortest distances if graph doesn't contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle

BELLMAN-FORD’S ALGORITHM

```
//V-1 iteration to ensure the distance from the source to nodes are steady
for (int i = 1; i < V; ++i) {
        for (int j = 0; j < E; ++j) {
                int u = graph.edge[j].src;
                int v = graph.edge[j].dest;
                int weight = graph.edge[j].weight;
                int temp_d=dist[v];

                //Updating distance if the current one is smaller than the recorded
                if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v]) {
                        dist[v] = dist[u] + weight;

                        if (v==0)prevVertex[0]=0;
                        else if (temp_d>dist[v])
                                prevVertex[v]=u;
                }
        }
```

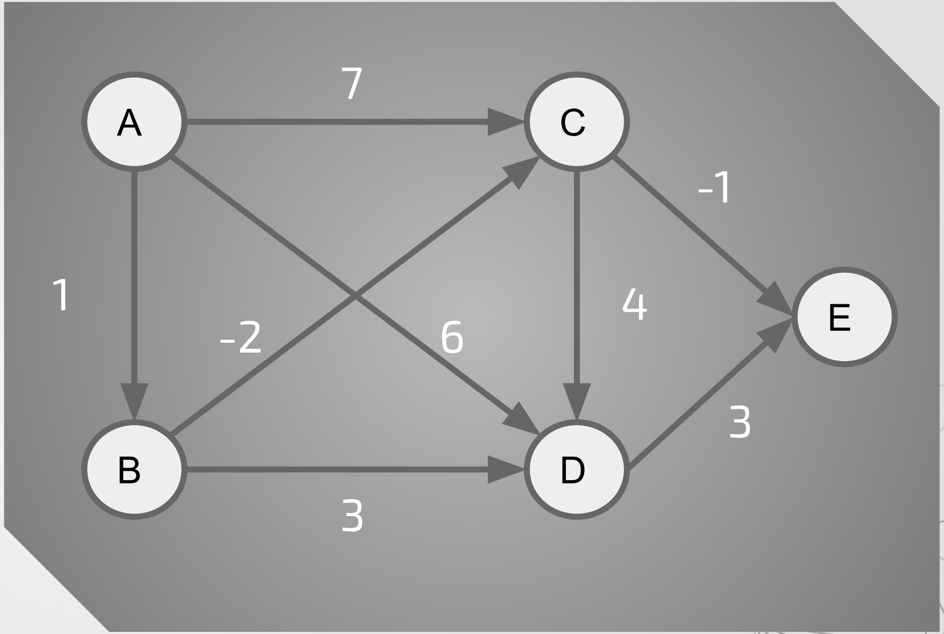
Vertex	Cost/ Distance	Previous Vertex
A	∞	
B	∞	
C	∞	
D	∞	
E	∞	

Edges:(A,B), (A,C), (A,D), (B,C), (B,D), (C,D), (C,E), (D,E)



Vertex	Cost/ Distance	Previous Vertex
A	0	None
B	1	A
C	-1	B
D	3	C
E	-2	C

Edges:(A,B), (A,C), (A,D), (B,C), (B,D), (C,D), (C,E), (D,E)



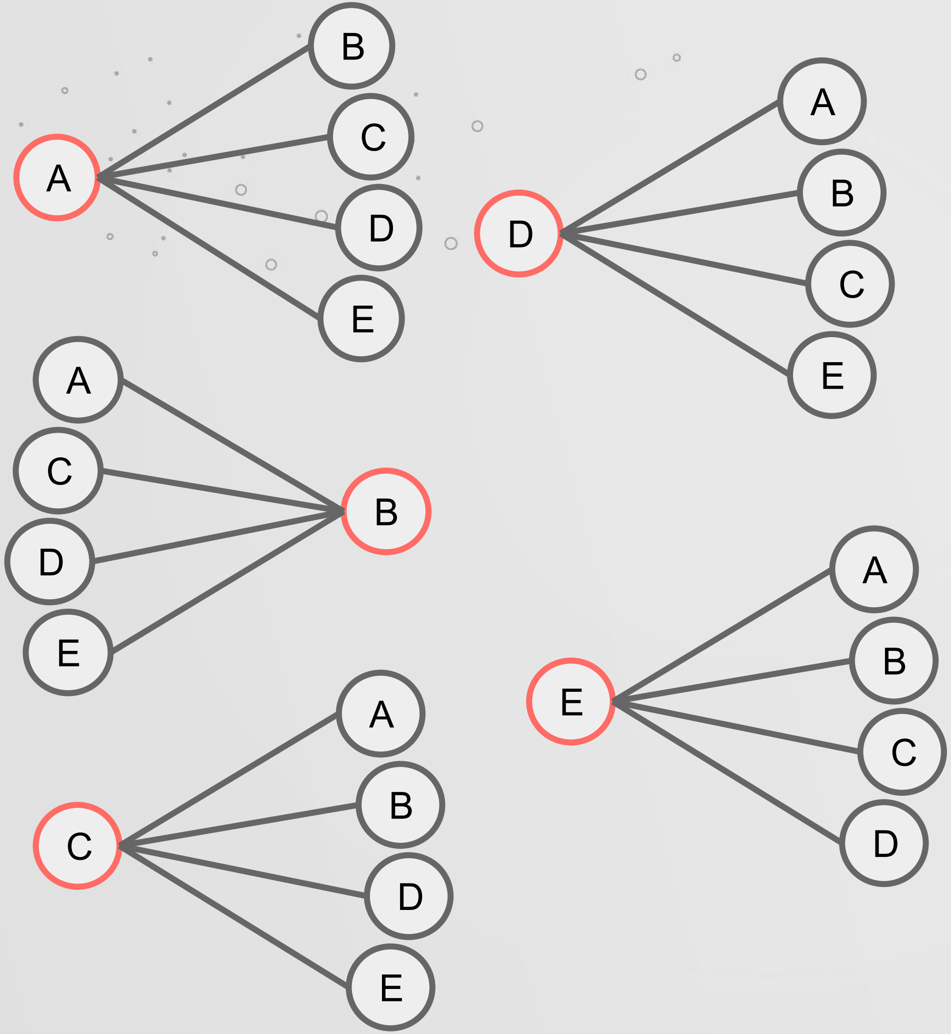
SHORTEST PATH: v(A) →v(B) → v(C) → v(E) or e(A,B) → e(B,C) → e(D,E)

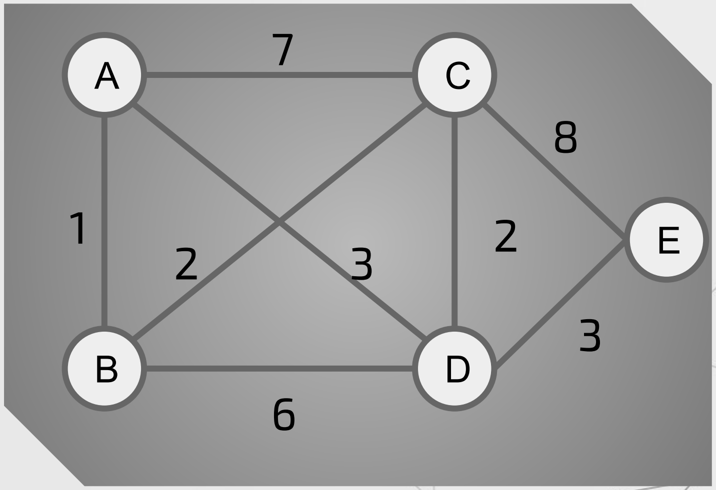
How about finding the all pairs shortest path in a graph?

FLOYD-WARSHALL'S ALGORITHM

This is algorithm is very similar to Dijkstra's, however the difference is that it computes all the pairs shortest distance of each node or vertex. Also, it do not have a boolean Visited_vertex variable where it strictly can visit the node only once. Thus, this updates distance whenever it changes and also applicable to negative-weighted graphs.This was proposed by Robert Floyd and Stephen Warshall in the same year 1962.

Time Complexity: $O(V^3)$
Space Complexity: $O(V^2)$;





Time Complexity: $(V^2) \times (V) = (\mathbf{V^3})$

FLOYD-WARSHALL'S ALGORITHM

1. **Declare** graph[v][v], distance [v][v], prevVertex[v][v];
2. **Initialization** of the distance variable by getting the initial distance of the edges.
3. Getting the first loop as the source for the entire row of the matrix and the second loop as the next destination and the third loop for scanning if the node is within the path of the shortest path from the source node.
   a. **Updating of the distance** in the distance matrix if the new computed distance is lesser than the current one

FLOYD-WARSHALL'S ALGORITHM

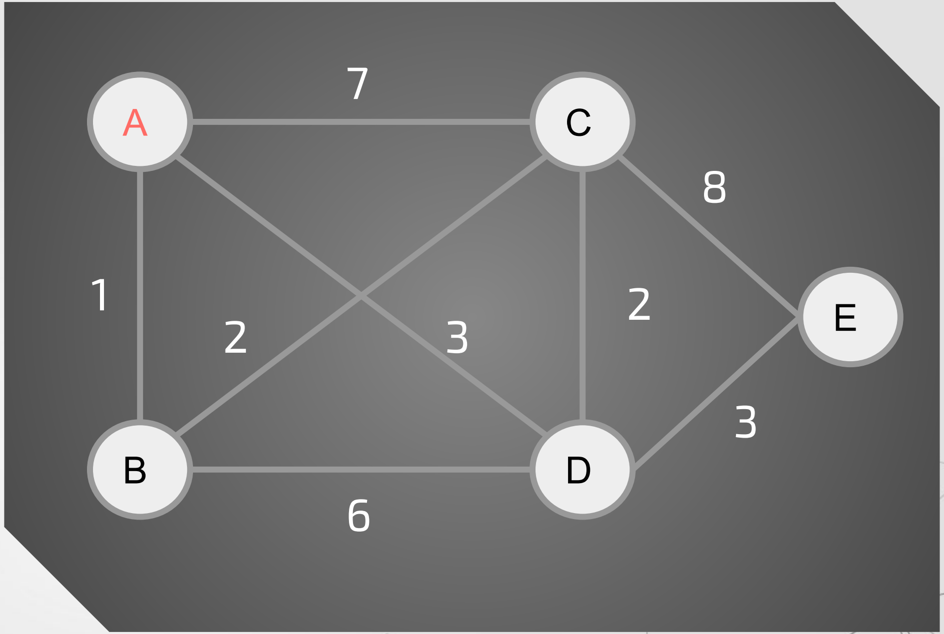
```
// Initialize the distance given
for (i = 0; i < V; i++)
    for (j = 0; j < V; j++)
        dist[i][j] = graph[i][j];

// Picking the source
 for (k = 0; k < V; k++){
    // Pick all the destination of the remaining vertices beside the source
    for (i = 0; i < V; i++){
            // Updating the distance if the vertices is included in the Shortest path
            for (j = 0; j < V; j++){
                if (dist[i][k] + dist[k][j] < dist[i][j])
                        dist[i][j] = dist[i][k] + dist[k][j];
    }}}
```

Shortest Distance from the Source

S/T	A	B	C	D	E
A	0	1	3	3	6
B	1	0	2	4	7
C	3	2	0	2	5
D	3	4	2	0	3
E	6	7	5	3	0

Shortest Path (Prev Vertex)

S/T	A	B	C	D	E
A	none	A	B	A	D
B	B	none	B	C	D
C	B	C	none	C	D
D	D	A	D	none	D
E	D	A	D	E	none



Time Complexity: $O(V^3)$

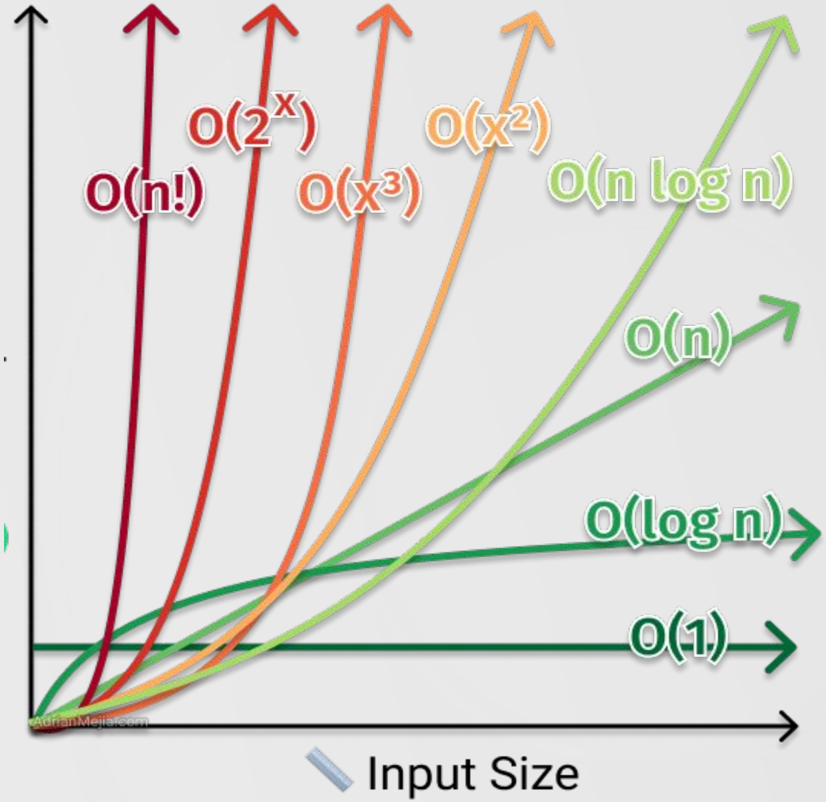
03

Comparison

Dijkstra vs. Bellman-Ford vs. Floyd Warshall

Comparison

Algorithm	Time Complexity	Source	Negative Weights	Method	Directed /Undirected
Dijkstra	O (E log V) or O (V^2)	Single	No	Greedy	Both
Bellman -Ford	O (VE)	Single	Yes	Dynamic Programming	Both
Floyd- -Warshall	O (V^3)	All	Yes	Dynamic Programming	Both



(Mejia, 2020)

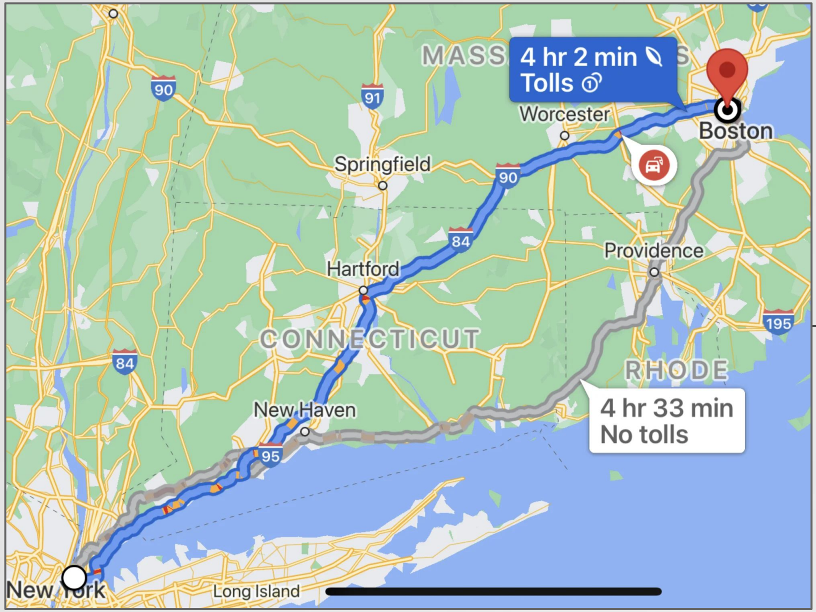
TIME COMPLEXITY

04

Applications

Real life applications

Digital Mapping Services



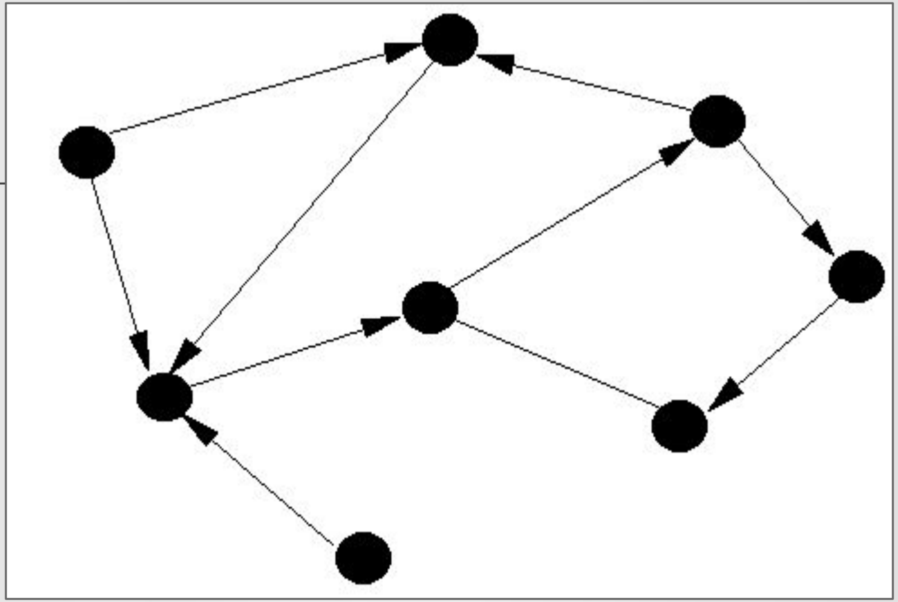
Google Map

This was based in Dijkstra's Algorithm which compute the minimum distance in various routes and paths from the source to the target location.

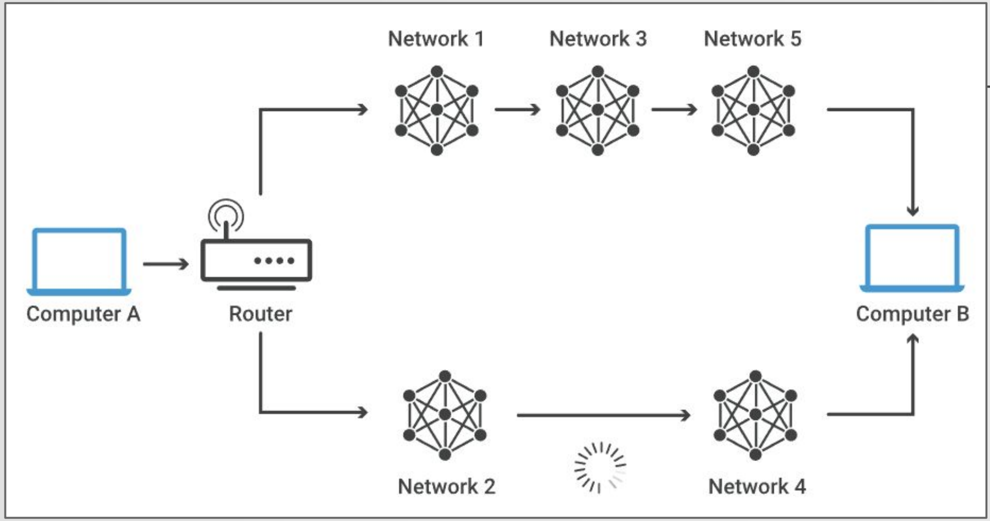
Communication Network

Telephone Network

In telephone network the bandwidth represents the amount information that can be transmitted by the line. Transmission line represent as edges and vertices are the stations.

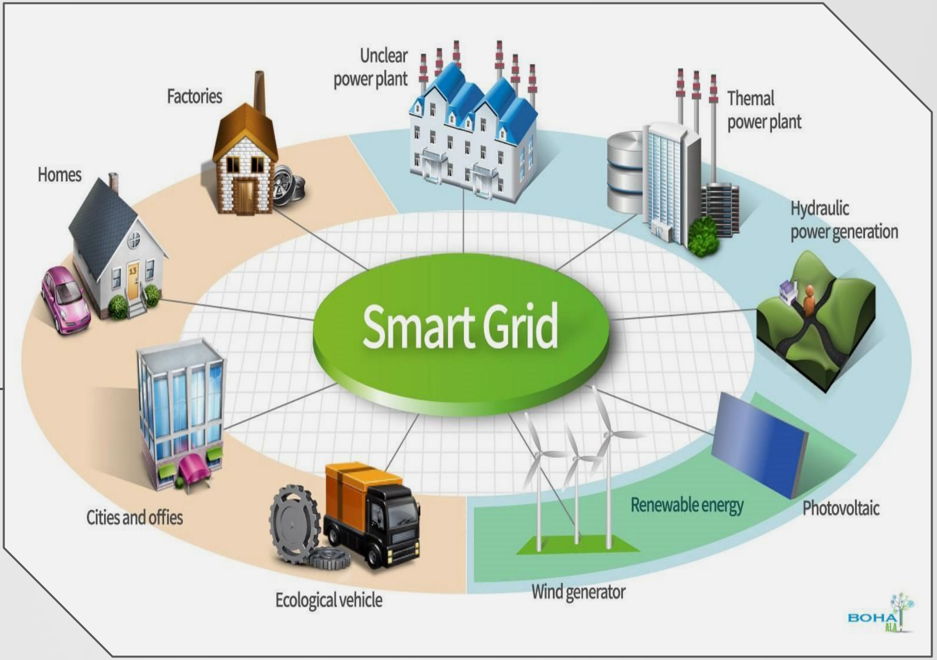


IP Routing



Internet have an Open Shortest Path First (OPSF) Protocol that is used to find the best shortest cost path between the source router and destination router. Also, Dijkstra’s algorithm is widely used in routing protocols.

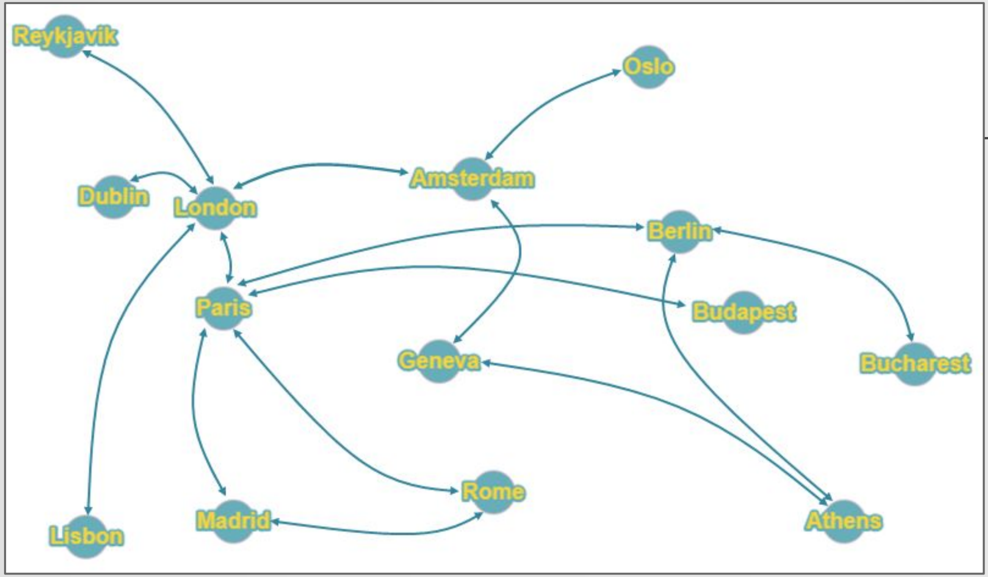
Smart Grid



Power System

This is a two way communication from the producers to consumers. In which the sources controls and automates the needs of the customers. This portrays the shortest problem path to which transmission line it will use towards the consumers destination.

Flighting Agenda



The flight agenda have certain access to database on flights such as the arrival and departure time that computes the earliest arrival time for the destination from the origin airport. Therefore, it can determine which flights are the shortest and have minimum cost towards the customer's destination.

05

DEMONSTRATION

Java and Gama Platform (Agent-Based Modelling)

RESOURCES

- GeeksforGeeks.(2020).Application of Dijkstra's Shortest Path. https://www.geeksforgeeks.org/applications-of-dijkstras-shortest-path-algorithm/
- Statistics.com.(2021).Directed Vs. Undirected Network. https://www.statistics.com/glossary/directed-vs-undirected-network/
- Indiatimes.(2021).Definition of Graph Theory. https://economictimes.indiatimes.com/definition/graph-theory
- Javatpoint.().Type of Graphs.https://www.javatpoint.com/graph-theory-types-of-graphs
- GITTA.(2016).Dijkstra Algorithm: Short terms and Pseudocode. http://www.gitta.info/Accessibiliti/en/html/Dijkstra_learningObject1.html
- ·GeeksforGeeks.(2021).Bellman-Ford Algorithm.https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/
- GeeksforGeeks.(2021). Floyd Warshall Algorithm. https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/
- Abdul Bari. (n.d.).Dijkstra's Algorithm; Bellman Ford; Floyd Warshall https://www.youtube.com/channel/UCZCFT11CWBi3MHNIGf019nw