SHORTEST PATH ALGORITHMS

An algorithm to find the shortest distance path between the source and destination vertices is called the shortest path algorithm.

Types of shortest path problem

i. Single source shortest path

Given an input graph G = (V,E) and a distinguished vertex S, find the shortest path from S to every other vertex in G. Example: Dijkstra‟s algorithm (weighted graph and unweighted graph).

ii. All pairs shortest path problem

Given an input graph G = (V,E). Find the shortest path from each vertex to all vertices in a graph.

Dijkstra‟s algorithm

Weighted Graph

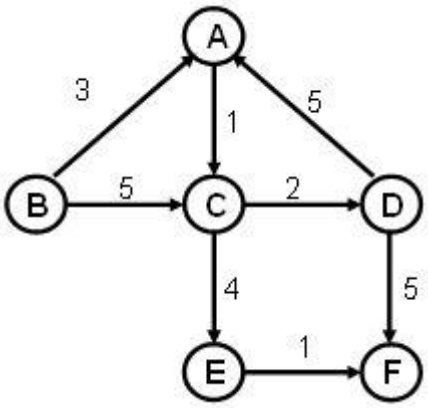
The general method to solve the single source shortest path problem is known as Dijkstra‟s algorithm. It applied to weighted graph.

Procedure

- It uses greedy technique.
- It proceeds in stages.
- It selects a vertex v, which has the smallest dv among all the unknown vertices and declares the shortest path from s to v is known.
- The remainder consists of updating the value of dw.
- We should set dw = dv + Cv, w, if the new value for dw would an improvement.

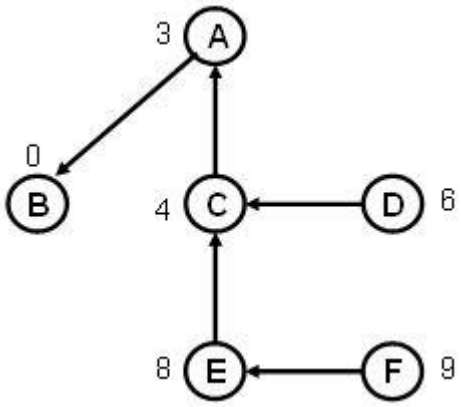
Example: Find the shortest path for the following graph.

Tracing Dijkstra's algorithm starting at vertex B:



Pass:	initially	1	2	3	4	5	6	Shortest distance	Predecessor
Active vertex:		B	A	C	D	E	F		
A	∞	3						3	B
B	0							0	-
C	∞	5	4					4	A
D	∞	∞	∞	6				6	C
E	∞	∞	∞	8	8			8	C
F	∞	∞	∞	∞	11	9		9	E

The resulting vertex-weighted graph is:



Algorithm Analysis

Time complexity of this algorithm O(|E| + |V|2) = O(|V|2)

Table Initialization routine

```
void InitTable(Vertex Start, Graph G, Table T)

{

int i;

ReadGraph(G,T);

for (i=0; i<NumVertex; i++)

{

T[i].known = False;

T[i]. Dist = Infinity;

T[i]. Path = NotAVertex;

}

T[Start]. Dist = 0;

}
```

Pseudocode for Dijkstra's algorithm

```
void Dijkstra(Table T)

{

Vertex v, w;

for( ; ;)

{

v = smallest unknown distance vertex;
```

```
if( v = = NotAVertex) break;

T[v]. kown = True;

for each w adjacent to v

if(!T[w].known)

 if(T[v].Dist + Cvw < T[w]. Dist)

{

/* update w*/ Decrease(T[w]. Dist to T[v].Dist + Cvw);

 T[w]. path = v;

}

 }

 }
```