# DIJKSTRA ALGORITHM

EXPERIMENT: No. (GREEDY ALGORITHM) PAGE No. DATE

## SINGLE SOURCE SHORTEST PATH

### I] Problem Definition

- It solves the single source shortest path problem on weighted graph (directed as well as undirected) $G = (V, E)$, for which all edge weights are non-negative

  i.e $w(u,v) \geq 0 \quad \forall (u,v) \in E$

- Given a digraph with non-negetive edge weights $G = (V, E)$ and a distinguished source vertex, $s \in V$, determine the distance and a shortest path from the source vertex to every vertex in the digraph.

### II] Algorithm

A] Notations: (i) $G(V, E)$ → Graph with set of vertices V & set of Edges E

(ii) for each vertex $v \in V$,

(a) $v.\pi$, predecessor, is either a vertex or NIL

(b) $v.d$, weight of shortest path ian upperbound on the weight of shortest path from s to v.

Shortest path estimate

Sundaram

Teacher's Sign.:

DIJKSTRA Algorithm uses following two procedures

B] Initialization :

In this, for every $v \in V$, its shortest path estimate i.e $v.d$ & predecessor i.e $v.\pi$ is initialized.

Algorithm

```
Initialize_Single_Source (G, s)
{
1.  for each vertex v ∈ V
    {
2.      v.d = ∞
3.      v.π = NIL
    }
4.  s.d = 0
}
```

Since for loop runs for each vertex, complexity of this procedure is $\Theta(|V|)$

C] Relaxation

Process of relaxing an edge $(u, v)$ consists of testing whether shortest path of $v$ can be improved.

If so, update $v.d$ & $v.\pi$

PROGRAMMING

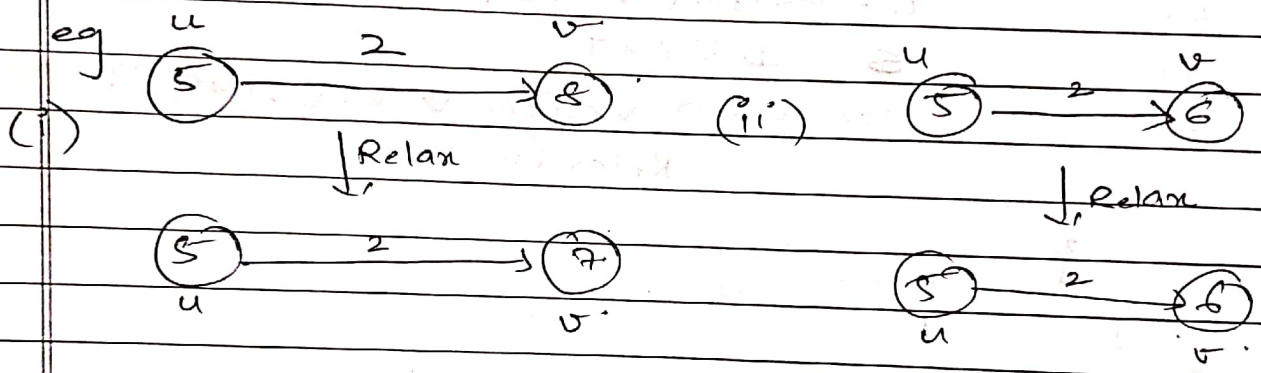
EXPERIMENT : No.

PAGE No.
DATE

Algorithm :

```
Relax (u, v, w)
{
1.    if v.d > u.d + w(u,v)
2.    {   v.d = u.d + w(u,v)
3.        v.π = u
      }
```

eg




## 5] DIJKSTRA'S ALGORITHM

- Algorithm maintains a set S of vertices whose final shortest path weights from source s is already determined.
- Algorithm repeatedly selects vertex $u \in V-S$ with shortest path estimate, adds u to S.
- After adding u, it relaxes all edges leaving u.
- Algorithm also use min-priority queue, Q of vertices, keyed by their shortest path estimate d.

## Algorithm

```
DIJKSTRA (G, w, s)
{
1.  Initialize_Single_Source (G, s)
2.  D = ϕ                    // Empty set D
3.  Q = G.V                  // Min priority Queue.

4.  while Q ≠ 0
    {
5.      u = Extract_Min (Q)
6.      D = D ∪ {u}
7.      for each vertex v ∈ G.Adj[u]
8.          Relax (u, v, w)

    }
}
```

## III] Analysis

- Algo maintains MIN-PRIORITY Q by using 3 operations
  (i) INSERT (implicitly on line 3)
  (ii) Extract_Min (line 5)
  (iii) DECREASE_KEY (implicitly in Relax i.e line 8)

- Since each vertex $u \in V$ is added to set D exactly once, so INSERT & EXTRACT_MIN operation is called once per vertex i.e **|V| times.**

- Since each edge $e \in E$ is relaxed only once, therefore for loop is executed |E| times i.e DECREASE_KEY operation runs for **|E| times**.

EXPERIMENT : No.

PAGE No.
DATE

- Running time of Dijkstra's algorithm depends on how min priority queue is implemented.

| Data Structure | Running time of each operation | | |
|---|---|---|---|
| | INSERT | DECREASE_KEY | EXTRACT_MIN |
| Array | O(1) | O(1) | O(V) |
| Binary Heap | time to build entire heap is O(V) | O(log V) | O(log V) |
| Fibonacci Heap | —11— | O(1) amortized time | O(log V) |

For Array :-

Running time complexity = $O(\underbrace{V}_{\text{line 4}}[\underbrace{V}_{\text{line 5}}] + \underbrace{E}_{\text{line 7 \& 8}})$

$= O(V^2 + E) = O(V^2)$

For Binary Heap :-

Time complexity = $O((\underbrace{V}_{\text{line 4}} + \underbrace{E}_{\text{line 7}}) \log V)$ (line 5 & line 8)

$= O(E \log V)$ if all vertices are reachable from source

Sundaram

Teacher's Sign.:

## IV. Printing Path

The following procedure prints out the vertices on a shortest path from $s$ to $v$ assuming that SSSP algorithm has already computed. $v.\pi$

### Algorithm

```
Print_Path (G, s, v)
{
1.  if  v == s
2.          print s

3.  elseif  v.π == NIL
4.          print "No path from 's' to 'v' exist.

5.  else
5.  {       Print_Path (G, s, v.π)
6.          print v
    }
```

### Drawbacks

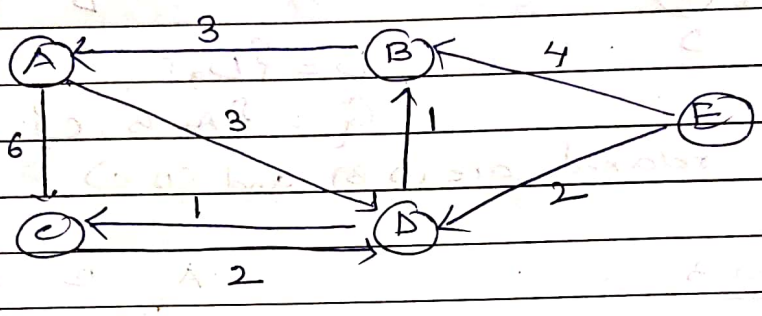
1. Blind search so waste lot of time while processing.
2. It cannot handle negative edges.
3. Sometime leads to cyclic graph, & cannot obtain right shortest path.

EXPERIMENT : No.

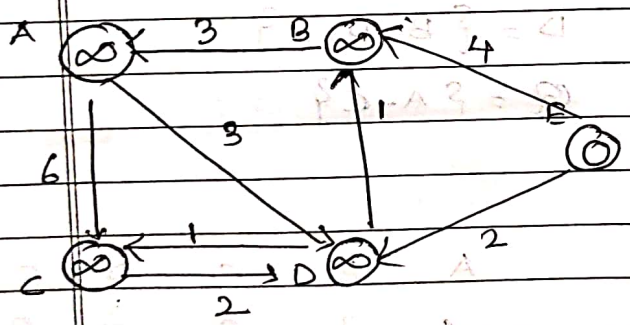
PAGE No.
DATE

## V] Example

Q1. Run DIJKSTRA's algorithm on the directed graph shown using vertex E as source vertex.




⇒ **Step I : Initialization**



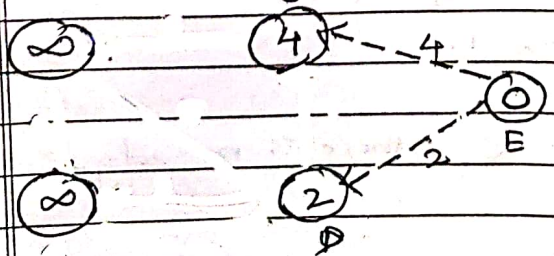

| | A | B | C | D | E |
|---|---|---|---|---|---|
| d | ∞ | ∞ | ∞ | ∞ | 0 |
| π | — | — | — | — | — |

$D = \phi$

$Q = \{A, B, C, D, E\}$

**Step II : Relaxation.**

**Iteration 1**




| | A | B | C | D | E |
|---|---|---|---|---|---|
| d | ∞ | 4 | ∞ | 2 | 0 |
| π | — | E | — | E | — |

$D = E$

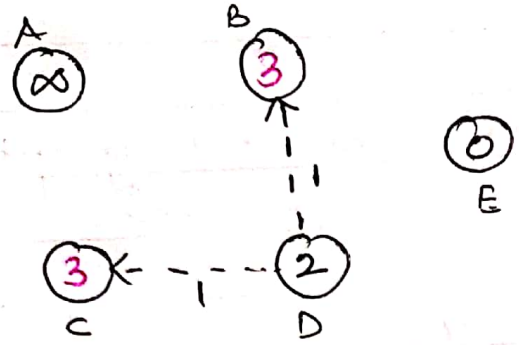
$Q = \{A, B, C, D\}$

Edges relaxed are (E, B) & (E, D)

Sundaram

Teacher's Sign.: _____

## Iteration 2:



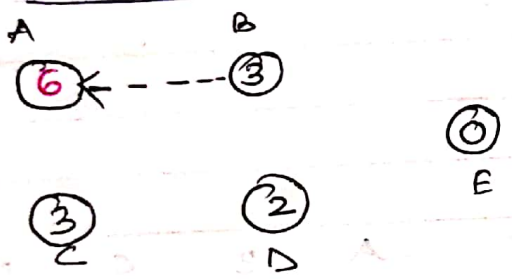

| | A | B | C | D | E |
|---|---|---|---|---|---|
| d | ∞ | 3 | 3 | 2 | 0 |
| π | — | D | D | E | — |

$D = \{D, E\}$

$Q = \{A, B, C\}$

Edges relaxed are (D, B) and (D, C)

## Iteration 3

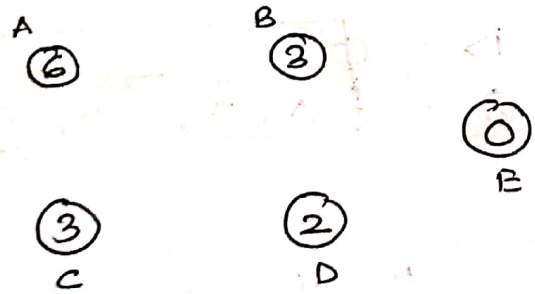



| | A | B | C | D | E |
|---|---|---|---|---|---|
| d | 6 | 3 | 3 | 2 | 0 |
| π | B | D | D | E | — |

$D = \{B, D, E\}$

$Q = \{A, C\}$

## Iteration 4




| | A | B | C | D | E |
|---|---|---|---|---|---|
| d | 6 | 3 | 3 | 2 | 0 |
| π | B | D | D | E | — |

$D = \{B, D, E, C\}$

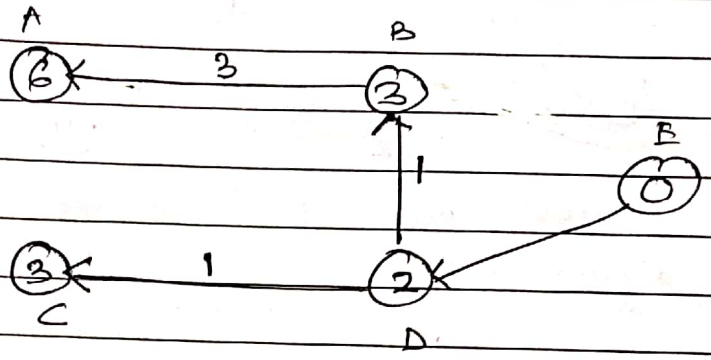
$Q = \{A\}$

No edges would be relaxed.

Similarly in Iteration 5, there is no edge relaxation.

EXPERIMENT : No.

PAGE No.
DATE




Step III : Printing Path

1) E → A

PP(E, A) → PP(E, B) → PP(E, D) → PP(E, E)

A ← B ← D ← E

2) E → C

E → D → C

3) E → B

E → D → B.

4) E → D

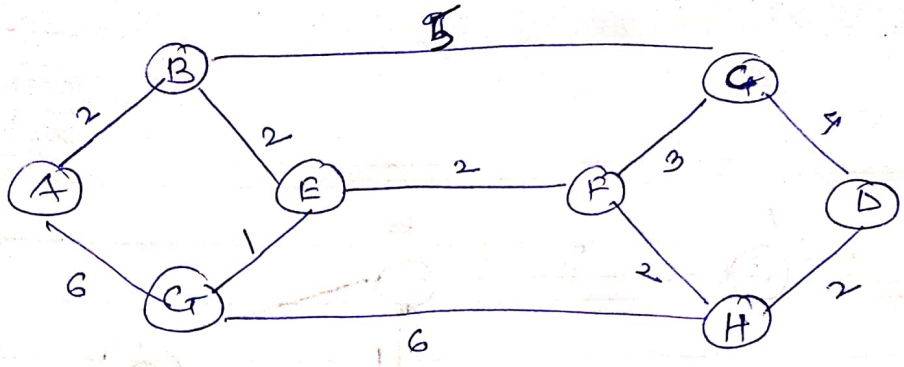
APPLICATIONS

1] Used in geographical maps - Any GIS application.
2] Routing application - Web can change based on traffic.
3] Telephone network.
4] In Games, it is used as "Pathfinding", used by AI to plot routes or by game engines to assist user
5] Social Network Analysis - To calculate similarity among users. Algo are such that one with shortest path may have many things in common.

Sundaram

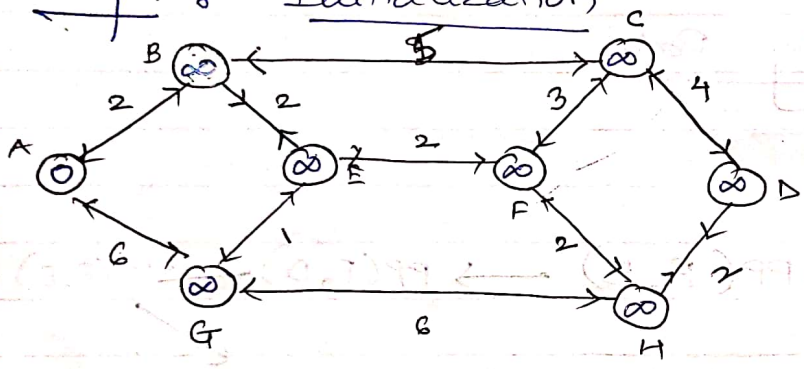
Teacher's Sign.:

2]




Find the shortest path from source vertex A using DIJKSTRA's algorithm.

Step 1 : Initialization



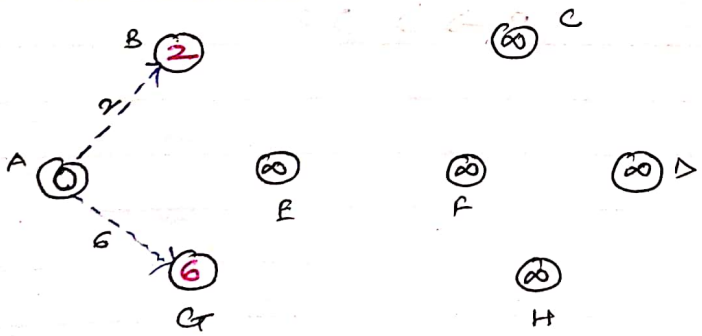

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| π | - | - | - | - | - | - | - | - |

$D = \phi$

$Q = \{A\ B\ C\ D\ E\ F\ G\ H\}$

Step 2 : Relaxation

Iteration 1



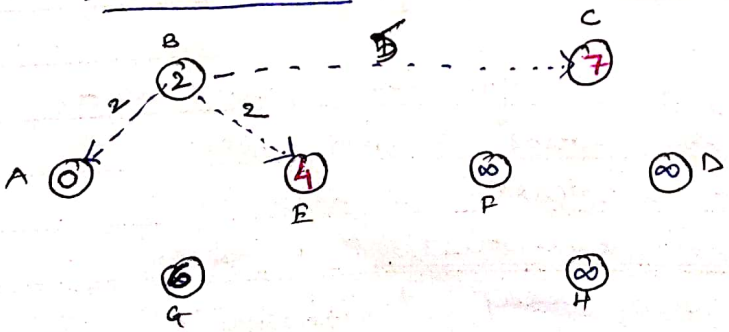

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | 2 | ∞ | ∞ | ∞ | ∞ | 6 | ∞ |
| π | - | A | - | - | - | - | A | - |

$D = \{A\}$

$Q = \{B\ C\ D\ E\ F\ G\ H\}$

Edges relaxed are (A, B) (A, G)

Iteration 2




| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | 2 | 7 | ∞ | 4 | ∞ | 6 | ∞ |
| π | - | A | B | - | B | - | - | - |

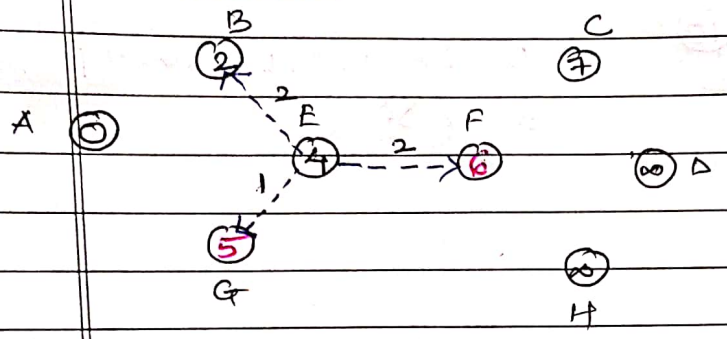
$D = \{A, B\}$

$Q = \{C\ D\ E\ F\ G\ H\}$

EXPERIMENT : No.

PAGE No.
DATE

## Iteration 3 :

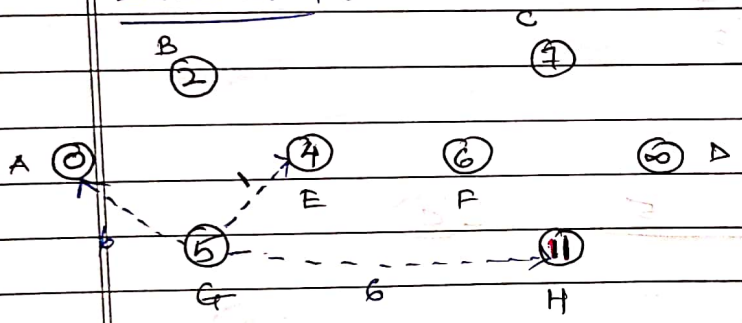



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | 2 | 7 | ∞ | 4 | 6 | 5 | ∞ |
| π | – | A | B | – | B | E | E | – |

D = { A, B, E }

Q = { C, D, F, G, H }

## Iteration 4 :

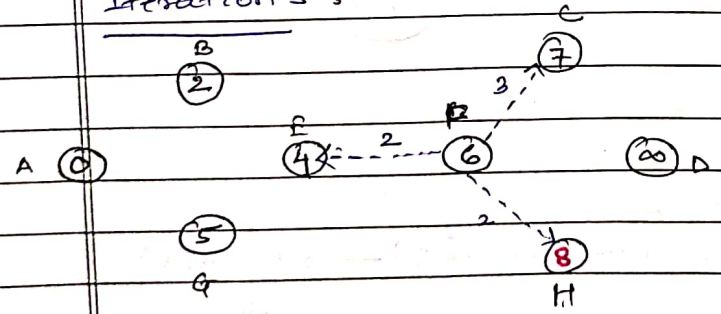



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | 2 | 7 | ∞ | 4 | 6 | 5 | 11 |
| π | – | A | B | – | B | E | E | G |

D = { A, B, E, G }

Q = { C, D, F, H }

## Iteration 5 :

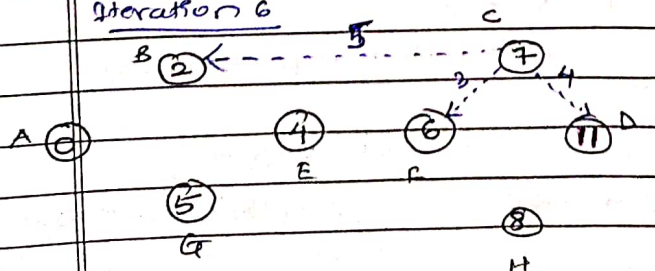



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | 2 | 7 | ∞ | 4 | 6 | 5 | 8 |
| π | – | A | B | – | B | E | E | F |

D = { A, B, E, G, F }

Q = { C, D, H }

## Iteration 6




| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | 2 | 7 | 11 | 4 | 6 | 5 | 8 |
| π | – | A | B | C | B | E | E | F |

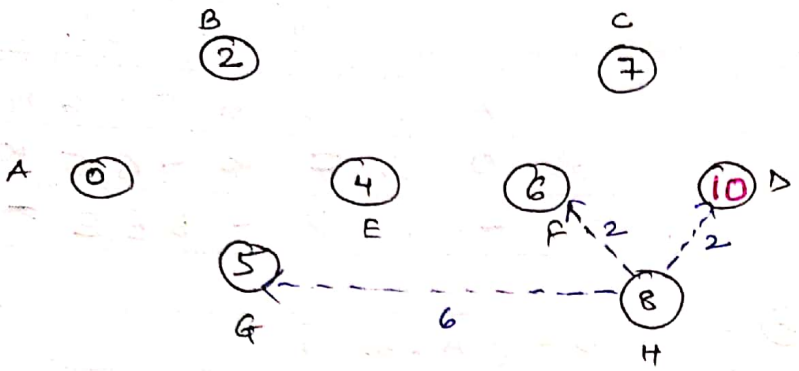
D = { A, B, C, E, F, G }

Q = { D, H }

Sundaram

Teacher's Sign.: _____

## Iteration 1

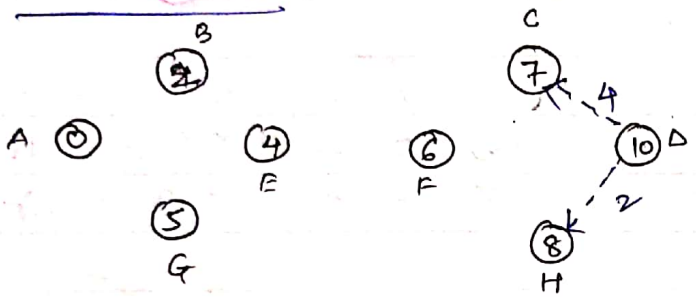



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| d | 0 | 2 | 7 | 10 | 4 | 6 | 5 | 8 |
| π | — | A | B | H | B | E | E | F |

D = {A, B, C, E, F, G, H}

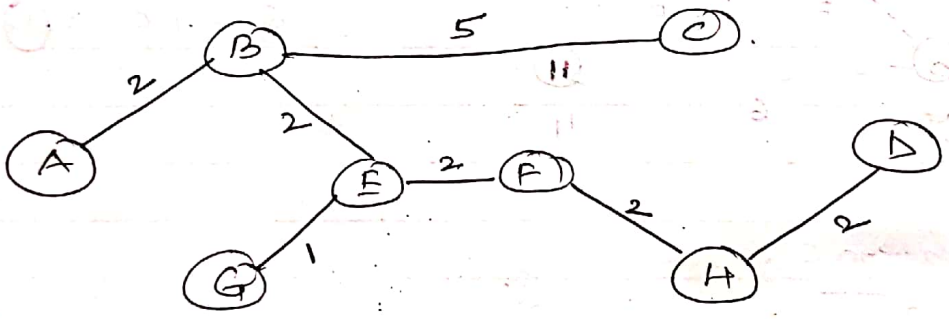
Q = {}

## Iteration 8




No edge relaxation.

∴




## Step III printing path

1] ~~A~~ → B

2] A → C

PP(A, C) → PP(A, B) → PP(A, A)

C ← B ← A

3] A → G

A → B → E → G

4] A → D

A → B → E → F → H → D

5] A → E

A → B → E

6] A → F

A → B → E → F

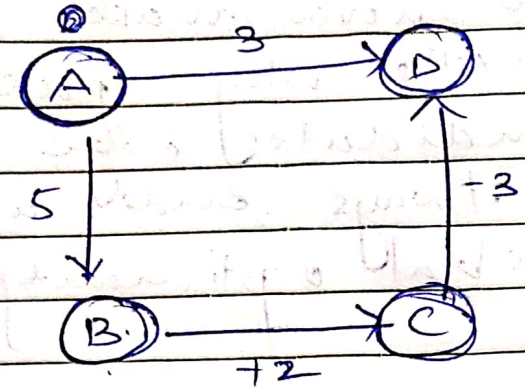
7] A → H

A → B → E → F → H

PAGE NO.
DATE / /

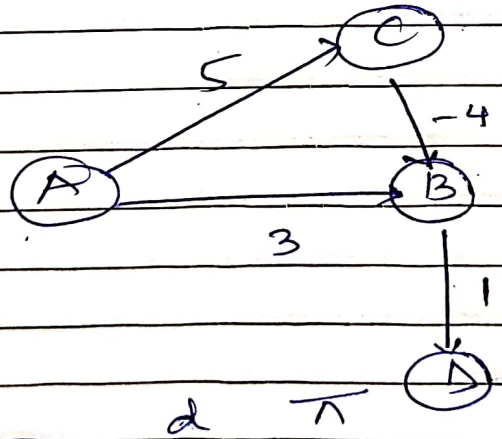
## Ex. With Negative edges

1]




| | d | π |
|---|---|---|
| A | 0 | – |
| B | ∞ 5 | A |
| C | ∞ 7 | B |
| D | ∞ 3 | A |

2]




| | d | π |
|---|---|---|
| A | 0 | – |
| B | ∞ 3 1 | A C |
| C | ∞ 5 | A |
| D | ∞ 4 | B |

Dijkstra fails.

# MINIMUM SPANNING TREE

EXPERIMENT : No.

PAGE No.
DATE

## (GREEDY ALGORITHM)

### I] What is spanning Tree?

- Given an undirected and connected graph $G \equiv (V, E)$,
- "A spanning tree of the graph G is a tree that spans G (i.e includes every vertex of G) and is a subgraph of G (i.e every edge in spanning tree belongs to G)".
- A spanning tree of G is a subgraph T, that is,
  i) Connected.
  ii) Acyclic
  iii) Includes all of the vertices.

### II] What is MST?

- Cost of spanning tree is the sum of the weights of all the edges in the tree.
  - There can be many spanning trees.
- MST is the spanning tree with minimum cost.
- There can be many MST,

Sundaram

Teacher's Sign.: