Prim’s Algorithm

Problem Definition: Implement Prim’s Algorithm to find minimum cost spanning tree of a given weighted connected Graph(G, V, E) and demonstrate its working.

CSE@GIT

Objectives of the Experiment:

1. To understand Graph and its Spanning Tree
2. To understand and appreciate the Greedy strategy
3. Present the working of Prim’s Algorithm
4. Learn to write Algorithm in standard form
5. Analyze the Algorithm for its time complexity

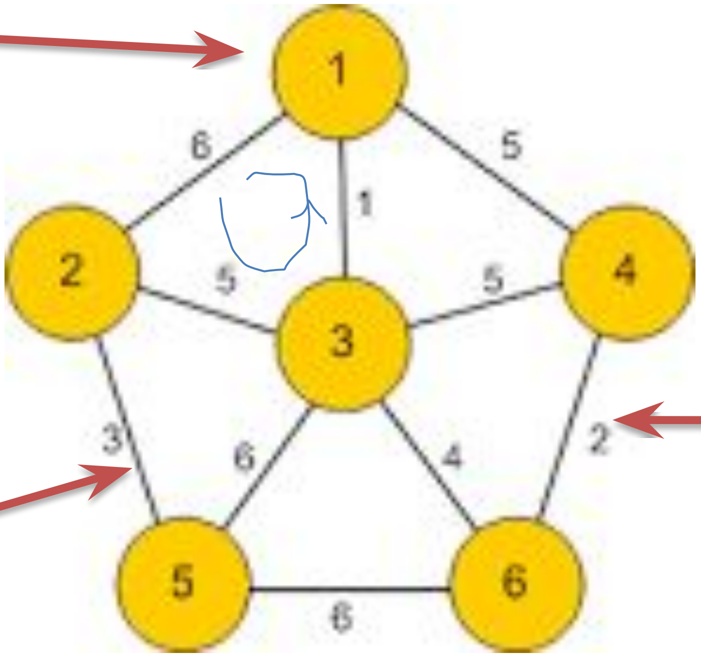
Graphs and their Properties

Wieght Connected Graph G(V, E)

G({n_1,n_2,n_3,…n_n}, {e_1},{e_2},{e_3}…………………{e_n})

G ({1,2,3,4,5,6}, {1,2},{1,4},{3,4},{{3,6}……..{5,6})

Node



Edge Weight

Edge

V – Node Vector

E -- Edge Vector

Cycles:.

1->2->3->1

1->4->3->1

4->3->6->4

5->6->4->3->5

1->2->5->6->4->1

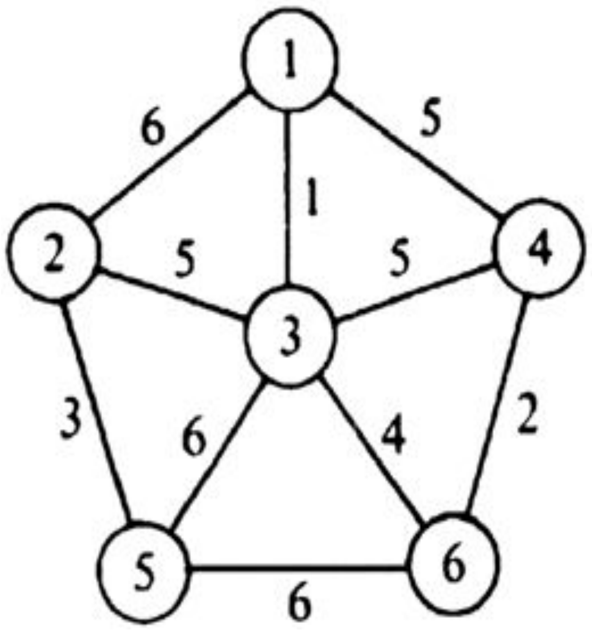
Minimum Spanning Tree

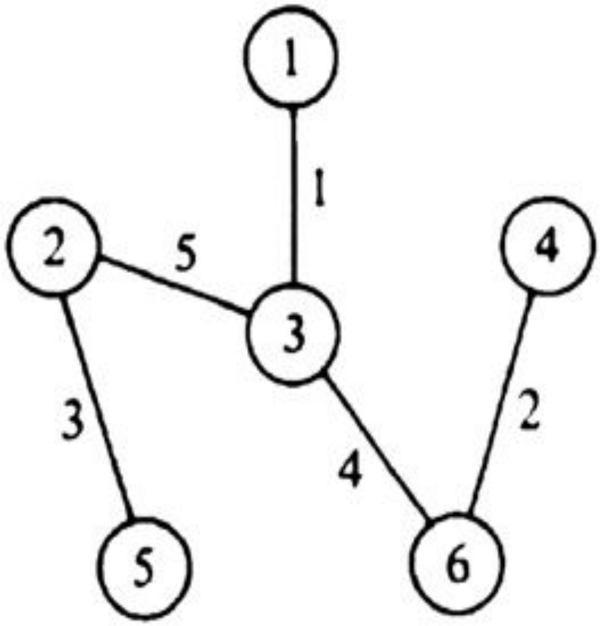
- A *spanning tree* of an undirected graph *G* is a subgraph of *G* that is a tree containing all the vertices of *G*.
- In a weighted graph, the weight of a subgraph is the sum of the weights of the edges in the subgraph.
- A *minimum spanning tree* (MST) for a weighted undirected graph is a spanning tree with minimum weight.

Graphs and its Spanning Tree

Graph G(V,E) T(V,E)

Spanning Tree

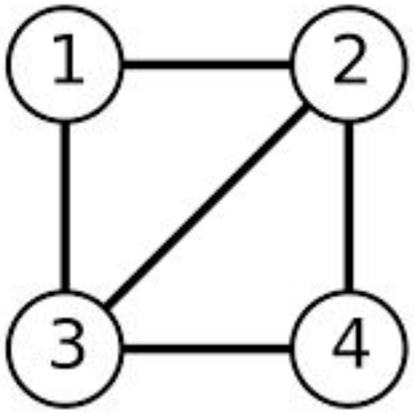


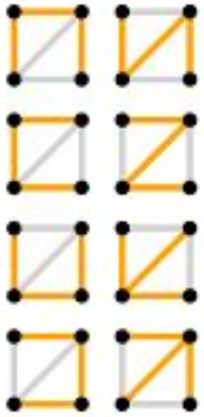


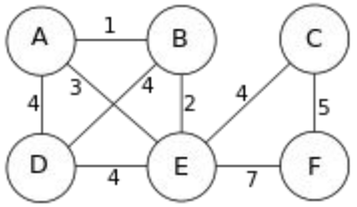
Graph may or may not have loops

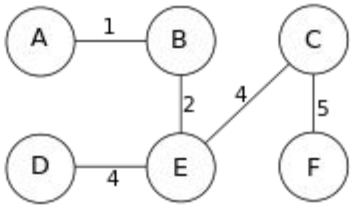
Tree will never have loops

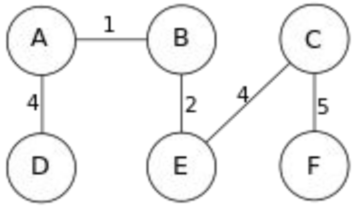
Graphs and its Spanning Tree More Examples…

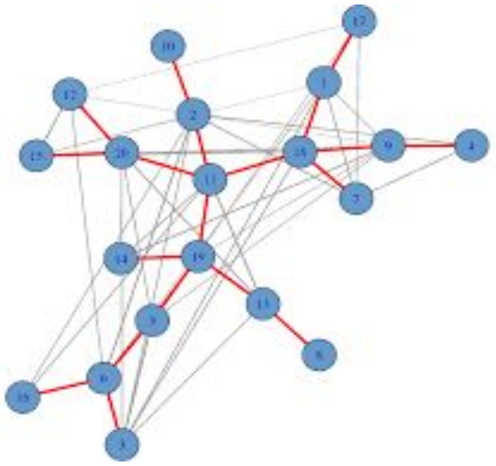












Prim’s Algorithm – Greedy Approach

- When we have a choice to make, make the one that looks best *right now*.

- Make a **locally optimal choice** in hope of getting a **globally optimal solution**.

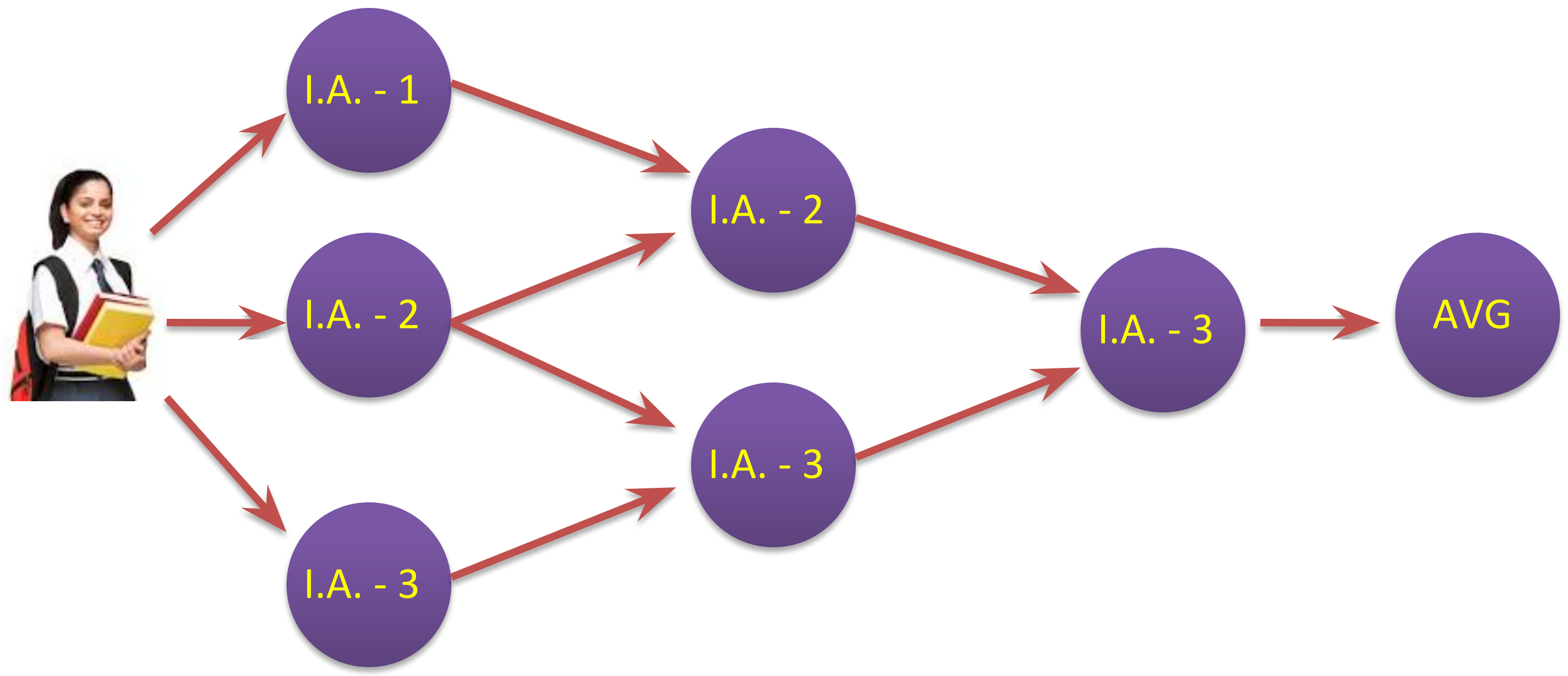
- **The choice that seems best at the moment is the one we go with.**

Example :

Score highest in the first I.A. Test so as to maximize average at the end of the semester

Greedy Choices

Maximize the Test Average



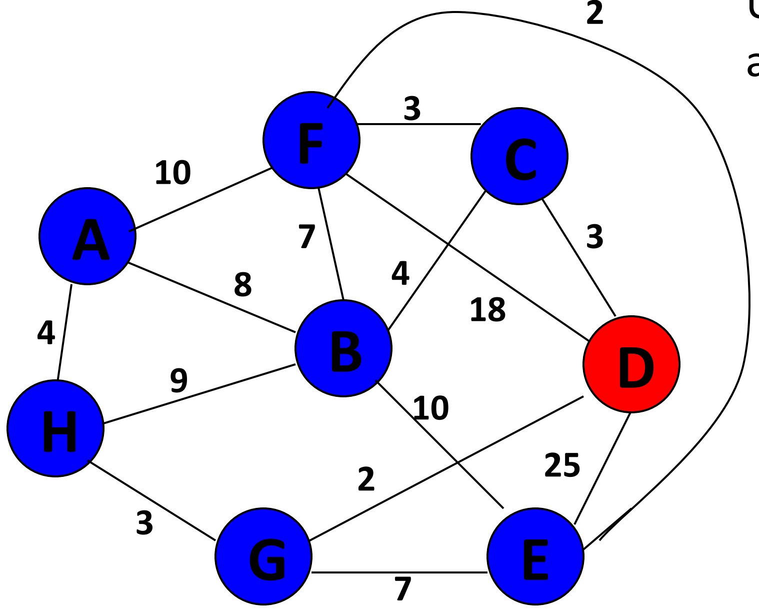
Typical Steps

- Cast the optimization problem as one in which we make a choice and are left with one sub-problem to solve.
- Prove that there's always an optimal solution that makes the greedy choice, so that the greedy choice is always safe.
- Make the greedy choice and **solve top-down**.
- May have to preprocess input to put it into greedy order.

General MST(Min. Cost Spanning Tree)

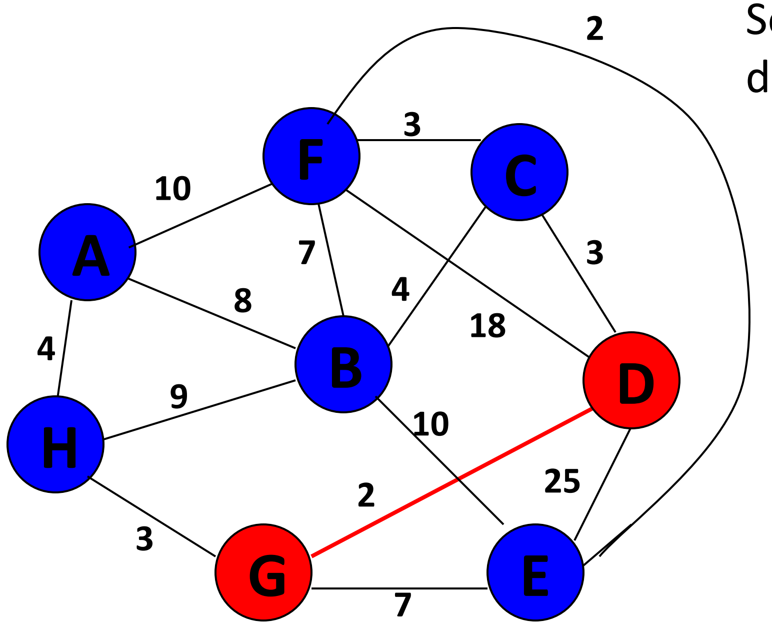
```
Generic-MST(G, w)

//Input :G a weighted connected Graph, w – weight matix
// Output : A the min. cost spanning tree

Let A=EMPTY;

Start at any node in the Graph…

while A does not form a spanning tree

    find an edge (u, v) that is safe for A,   add (u, v) to A

return A
```

Doesn't form a cycle



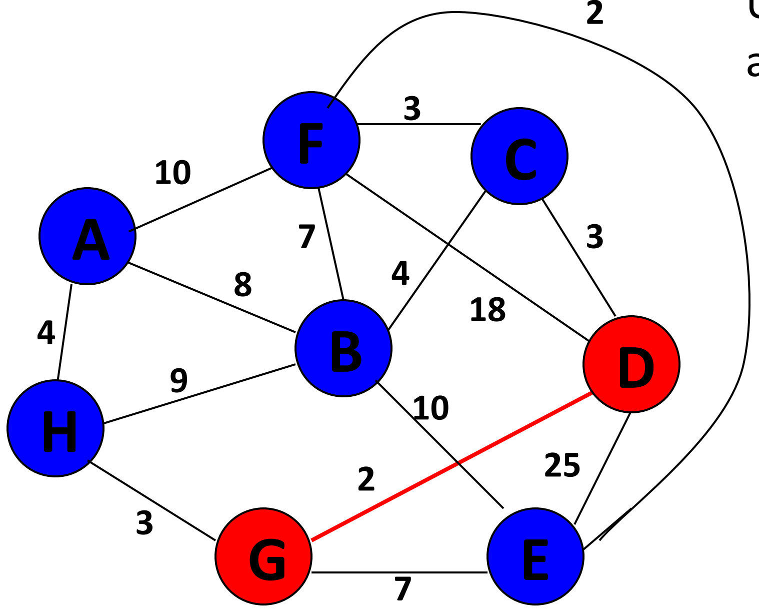
Update distances of adjacent, unselected nodes

	Visited	*U**	*V**
A			
B			
C		3	D
D	T	0	–
E		25	D
F		18	D
G		2	D
H			



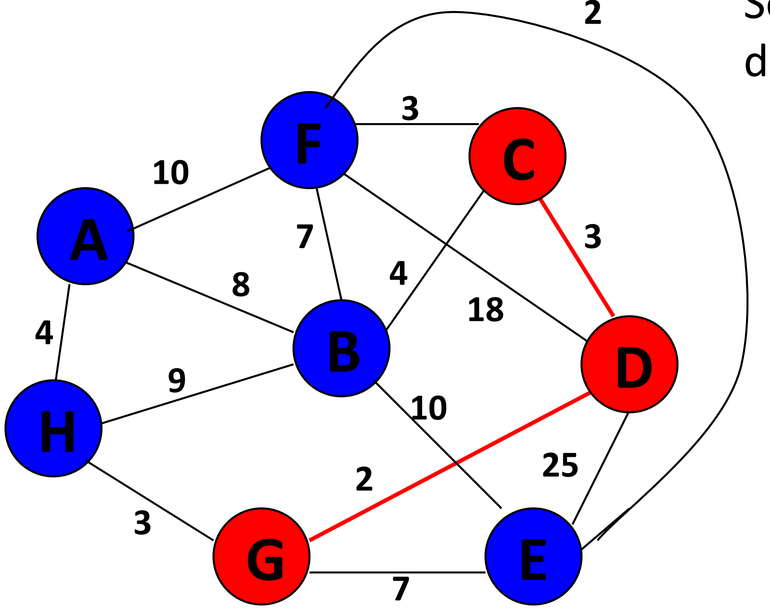
Select node with minimum distance

	Visited	*U**	*V**
A			
B			
C		3	D
D	T	0	–
E		25	D
F		18	D
G	T	2	D
H			



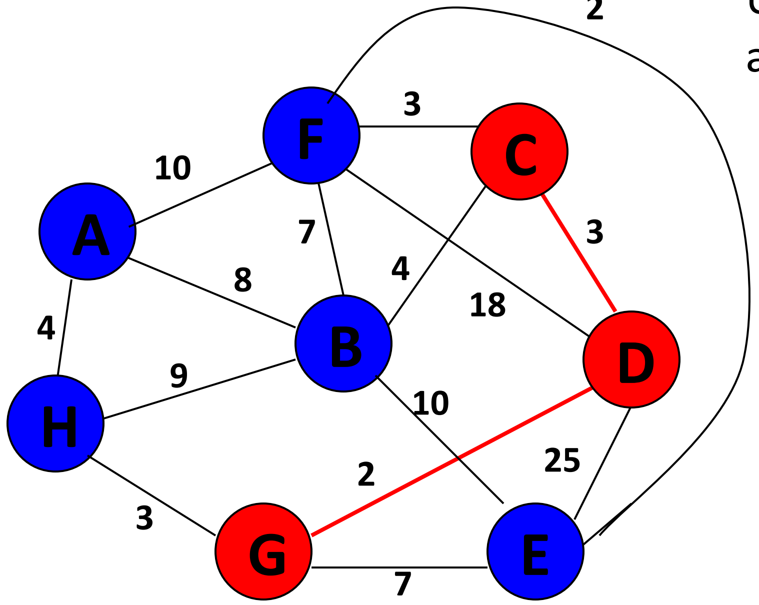
Update distances of adjacent, unselected nodes

	Visited	*U**	*V**
A			
B			
C		3	D
D	T	0	–
E		7	G
F		18	D
G	T	2	D
H		3	G



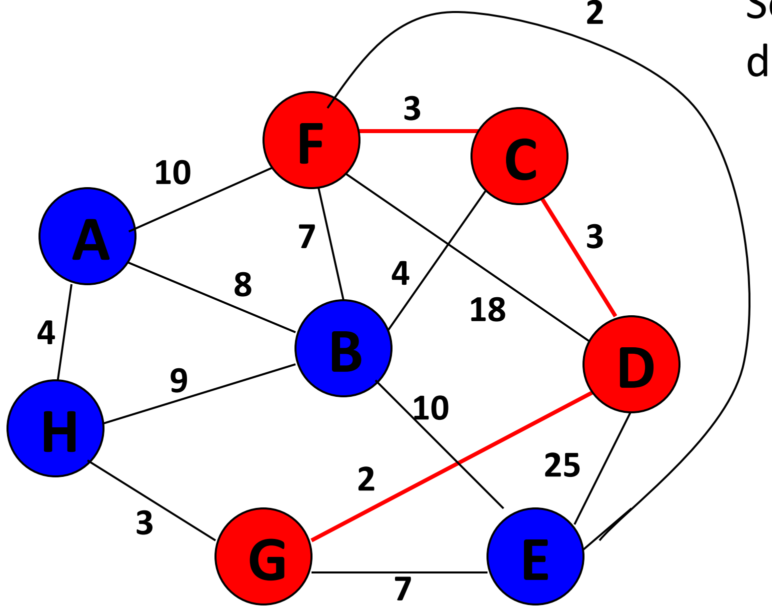
Select node with minimum distance

	Visited	*U**	*V**
A			
B			
C	T	3	D
D	T	0	–
E		7	G
F		18	D
G	T	2	D
H		3	G



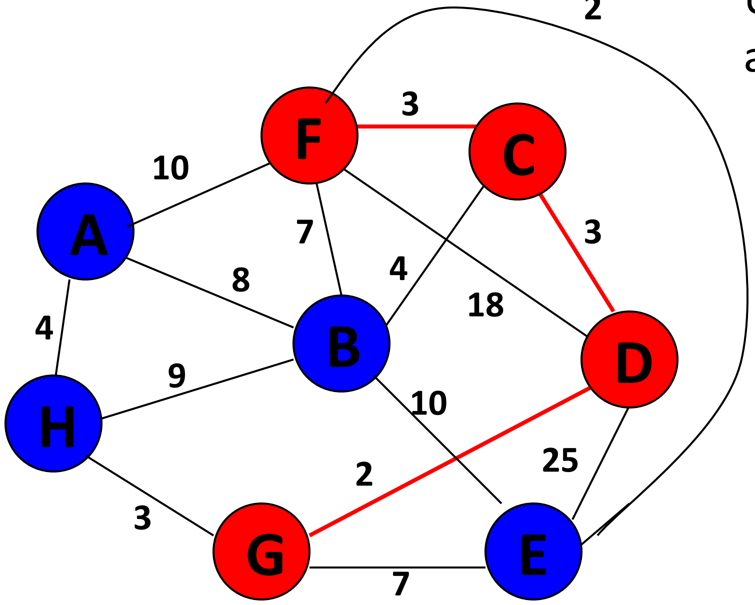
Update distances of adjacent, unselected nodes

	Visited	*U**	*V**
A			
B		4	C
C	T	3	D
D	T	0	–
E		7	G
F		3	C
G	T	2	D
H		3	G



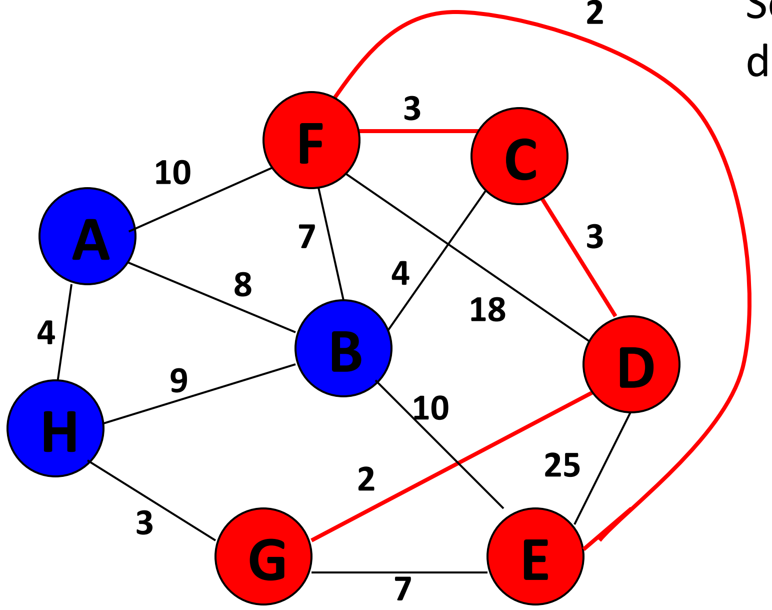
Select node with minimum distance

	Visited	*U**	*V**
A			
B		4	C
C	T	3	D
D	T	0	–
E		7	G
F	T	3	C
G	T	2	D
H		3	G



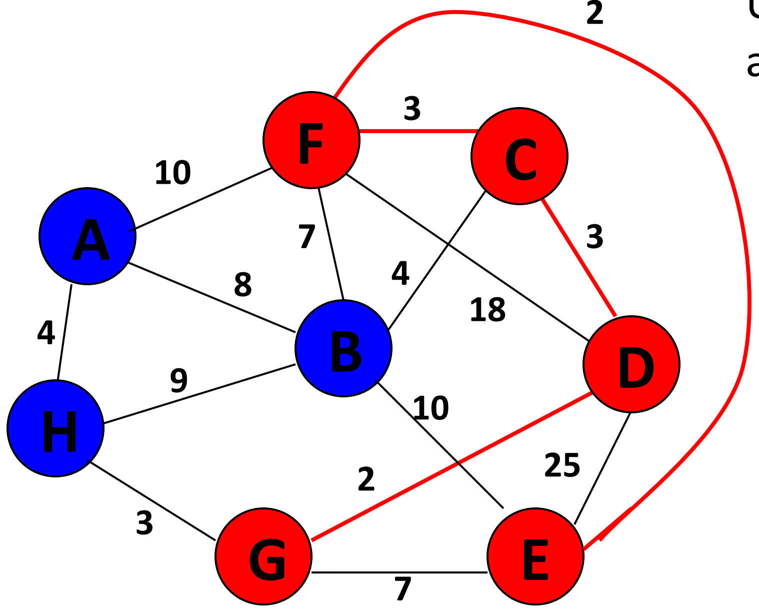
Update distances of adjacent, unselected nodes

	Visited	*U**	*V**
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E		2	F
F	T	3	C
G	T	2	D
H		3	G



Select node with minimum distance

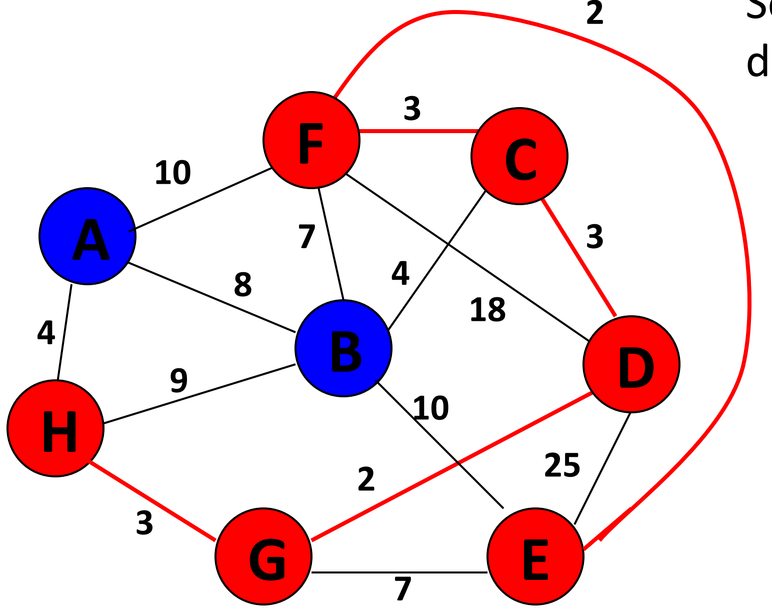
	Visited	*U**	*V**
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H		3	G



Update distances of adjacent, unselected nodes

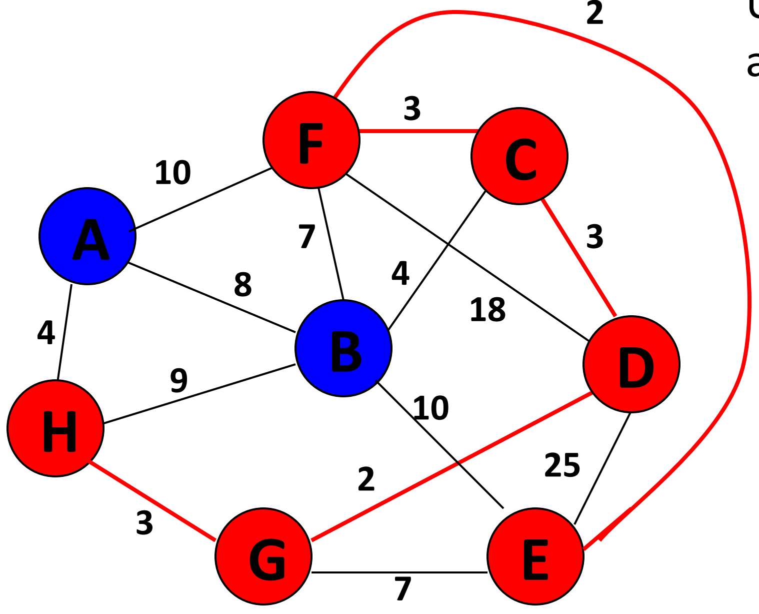
	Visited	*U**	*V**
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H		3	G

Table entries unchanged



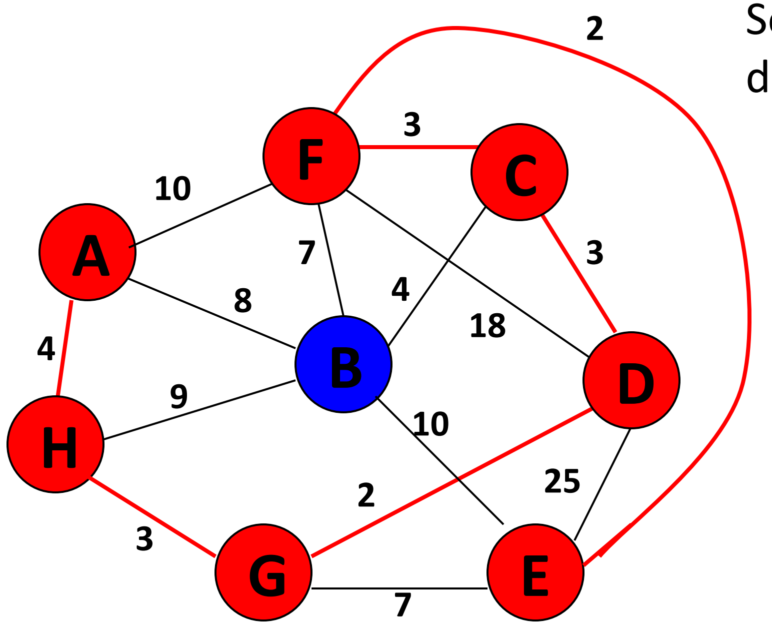
Select node with minimum distance

	Visited	*U**	*V**
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



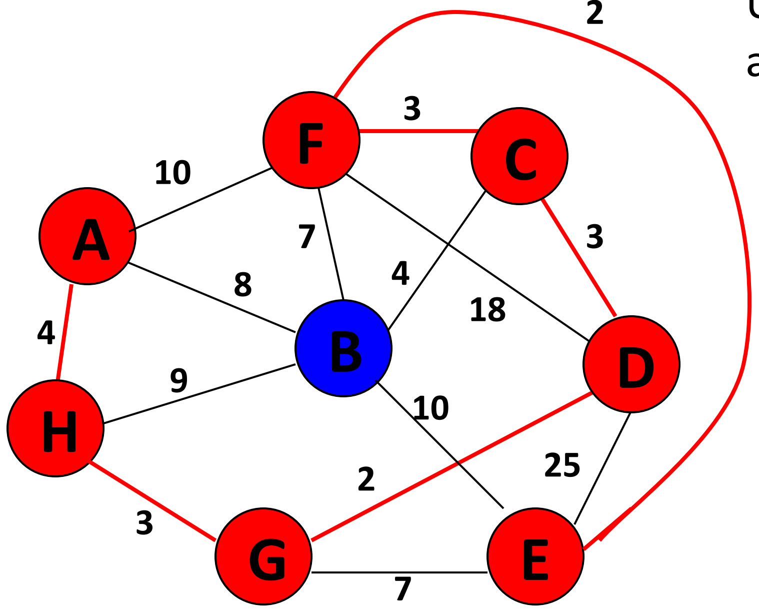
Update distances of adjacent, unselected nodes

	Visited	***U****	***V****
A		4	H
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



Select node with minimum distance

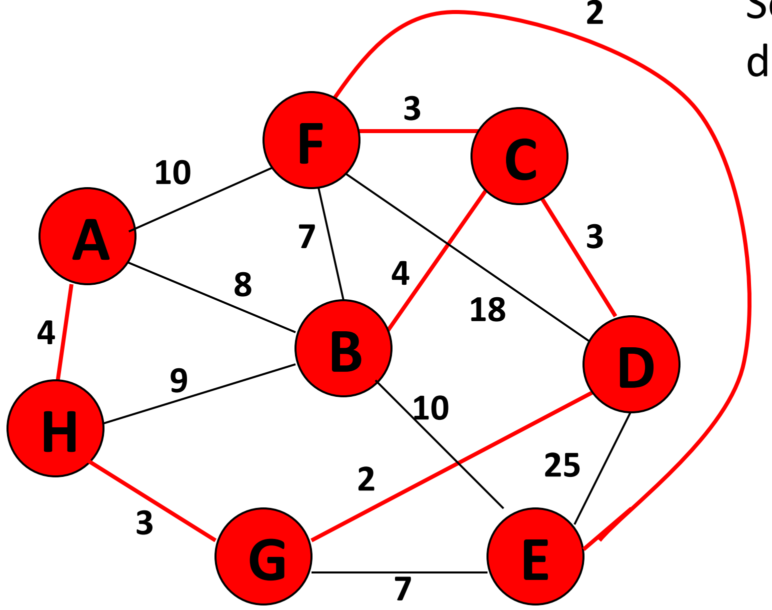
	Visited	*U**	*V**
A	T	4	H
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



Update distances of adjacent, unselected nodes

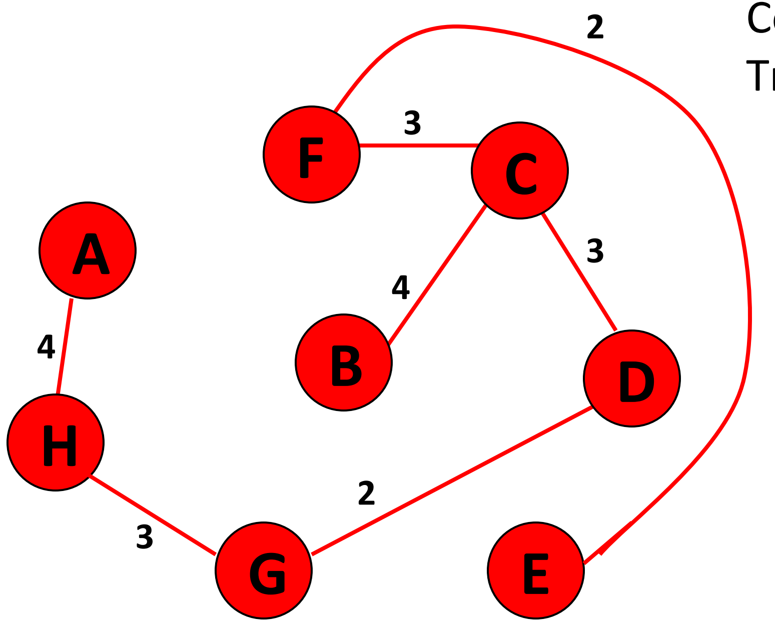
	Visited	*U**	*V**
A	T	4	H
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G

Table entries unchanged



Select node with minimum distance

	Visited	*U**	*V**
A	T	4	H
B	T	4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G



Cost of Minimum Spanning Tree = Σ d_v = 21

	Visited	*U**	*V**
A	T	4	H
B	T	4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G

Done

Prim’s Algorithm

```
ALGORITHM Prim(G)
//Prim’s algorithm for constructing a minimum spanning tree
    //Input: A weighted connected graph G = V,E //Output: ET ,
    the set of edges composing a minimum  spanning tree of G

  VT  ← {v0} //the set of tree vertices can be initialized with any vertex
  ET ← ∅
  for i ← 1 to |V | − 1 do
find a minimum-weight edge e∗ = (v∗, u∗) among all the edges (v,
    u) such that v is in VT and u is in V − VT
 VT ← VT ∪ {u∗}
 ET ← ET ∪ {e∗}
 return ET
```

Time and Space Complexity

- Correctness can be proven by induction on the number of vertices

- Applicable to both undirected and directed graphs
- Efficiency
    - **$O(|V|^2)$** for graphs represented by adj. matrix.
    - **$O(|E|\log|V|)$** for graphs represented by adj. lists.

Learning Outcome of the Experiment and Conclusion

At the end of the session, students should be able to :

1. Explain the working of Greedy Strategy. [L2, CO 2, PO1]
2. Demonstrate the working of Prim's algorithm on a given Graph G(V, w) [L3, CO 2, PO3]
1. Test the algorithm for a number of test cases and verify. [L5, CO 2, PO3]
2. Analyze the algorithm's time complexity [L4, CO 2, PO3]