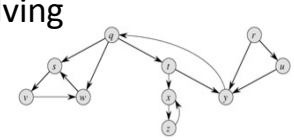
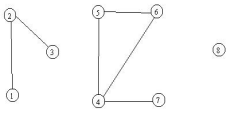
Graphs: Minimum Spanning Trees

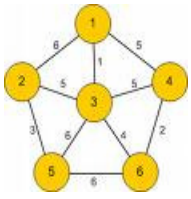
Atul Gupta

Graphs

- Most un-restricted form of data organization
- Final destination for problem solving
- Many variances
  - Directed and un-directed graphs
  - Connected and un-connected graphs
  - Weighted graphs



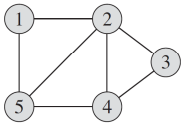




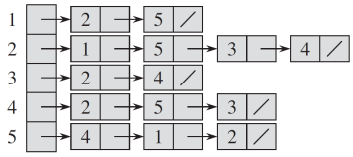
Representations of a Graph

Undirected Graph

- Adjacency List
- Adjacency Matrix



(a)



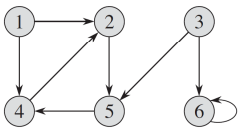
(b)

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

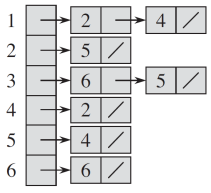
(c)

Representations (Di-Graph)

- Adjacency List
- Adjacency Matrix



(a)



(b)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

Problems Specific with Graphs

- Minimum Spanning Tree
- Path Problems
  - Simple Paths
  - Shortest Path Problem
    - Single source shortest paths
    - All-pair shortest paths
  - Find Cycles
  - Euler Path and Circuit Problem
  - Hamiltonian Path and Circuit Problem (or TSP)
- Graph Coloring
- Connected Components
- Isomorphic graphs
- Search Graphs

Minimum Spanning Tree

- Minimize the connecting media
- A tree with minimum weights
- Two Algorithms
  - Prim's MST
  - Kruskal's MST

A Generic Spanning Tree

GENERIC-MST(G, w)

1 $A = \emptyset$
2 **while** A does not form a spanning tree
3 find an edge (u, v) that is safe for A
4 $A = A \cup \{(u, v)\}$
5 **return** A

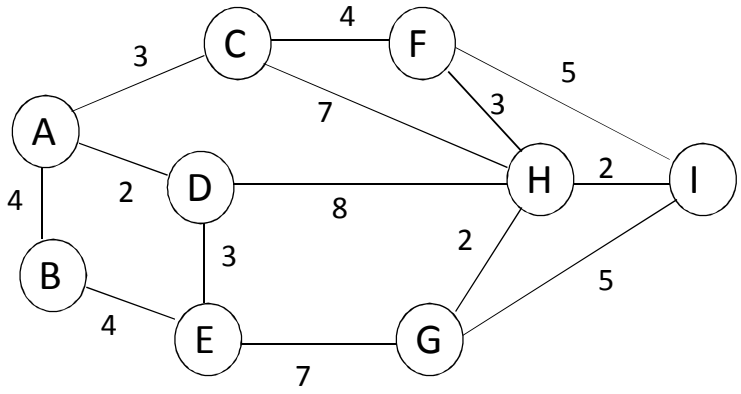
Prim's MST Algorithm

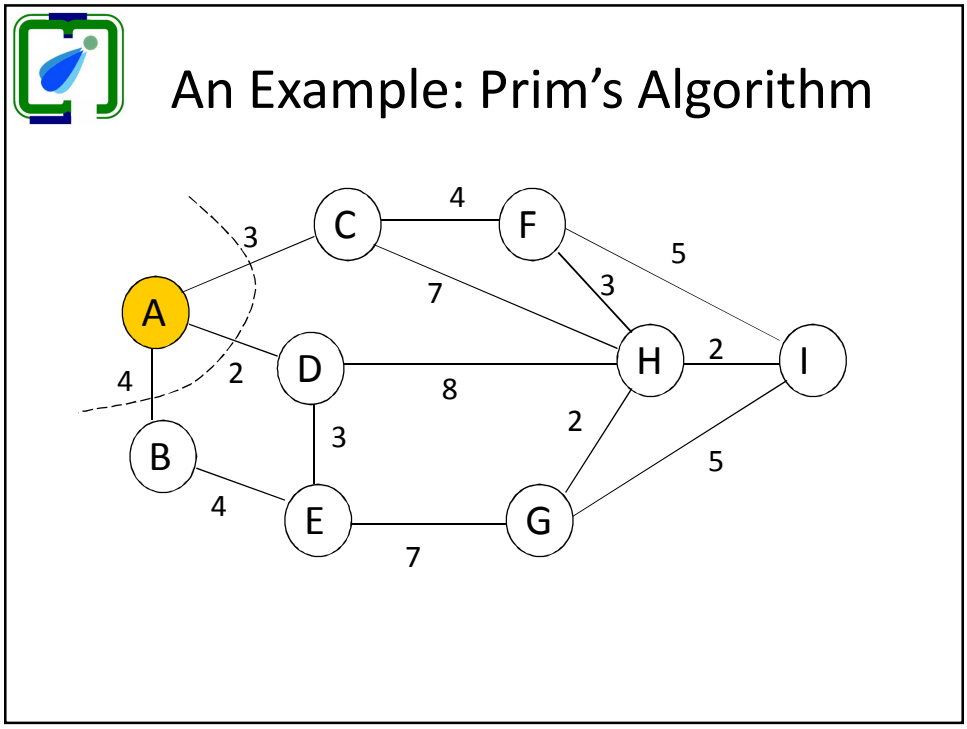
- The Idea:
  - Start with an empty set of vertices, A and S stores all the vertices of the graph
  - Initialize a key vector, key[u] for all u from 1 to n to ∞
  - Key[r] ← 0; A ← A union r
  - Put all the vertices in a priority Q except the starting one
  - while Q is not empty
    - u ← Extract-Min(Q)
    - for all v ϵ adj[u] and not in A
      update key[v]
  - Another example of a greedy approach

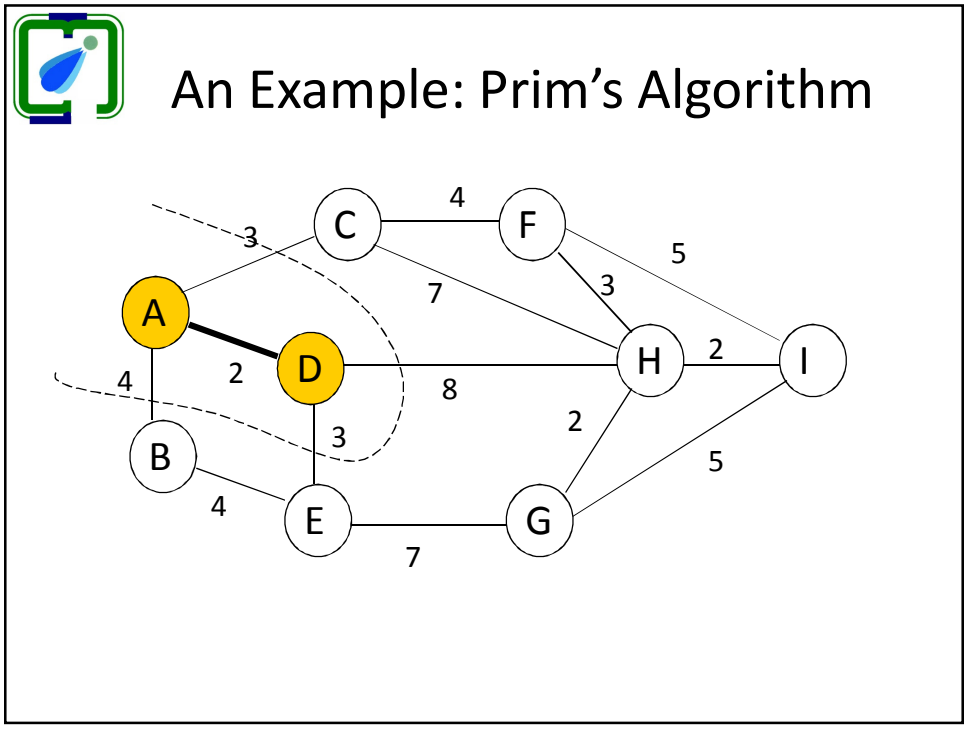
Prim's MST Algorithm

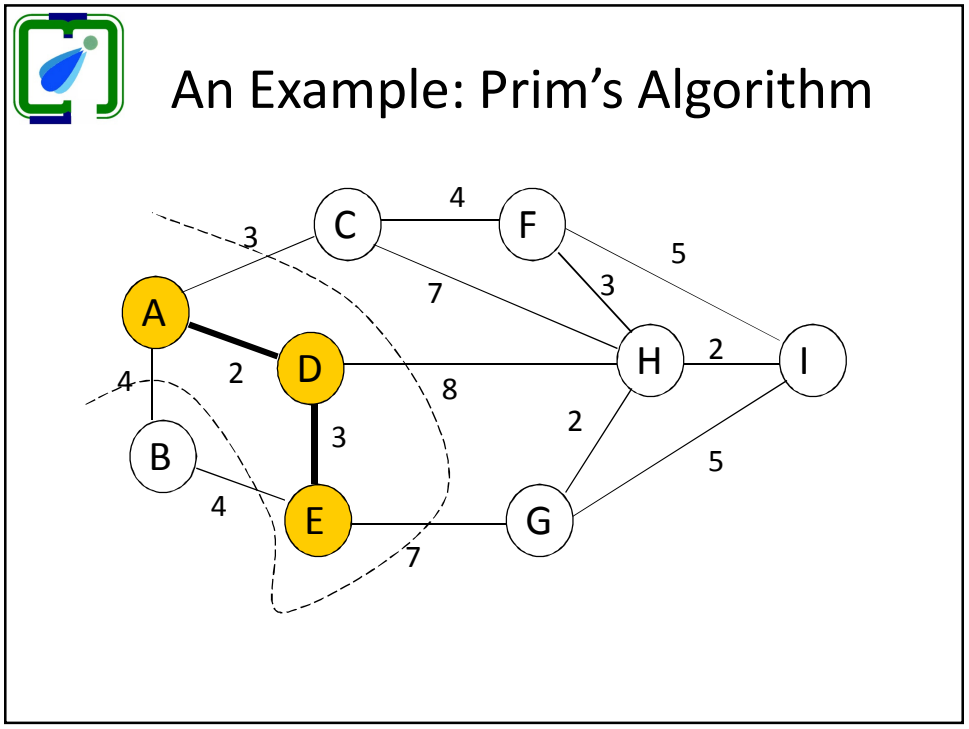
```
MST-PRIM(G, w, r)
1 for each u ∈ V [G] do
2         key[u] ← ∞
3         π[u] ← NIL
4 key[r] ← 0
5 Q ← V [G]
6 while Q ≠ Ø do
7         u ← EXTRACT-MIN(Q)
8         for each v ∈ Adj[u] do
9                 if v ∈ Q and w(u, v) < key[v]
10                then π[v] ← u
11                key[v] ← w(u, v)
```

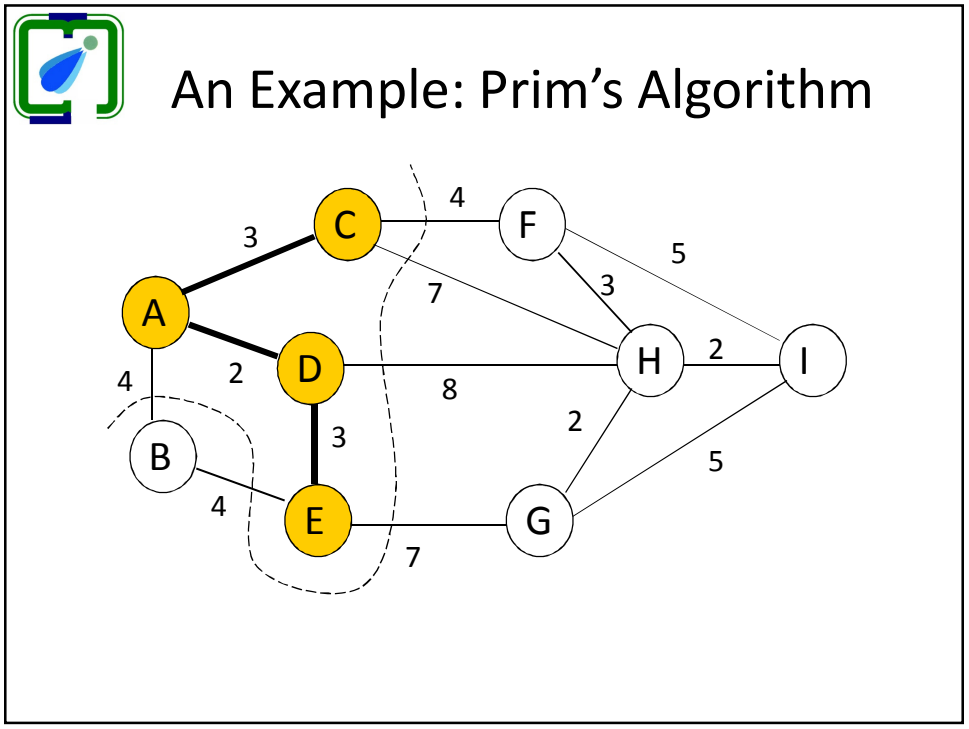
An Example: Prim's Algorithm

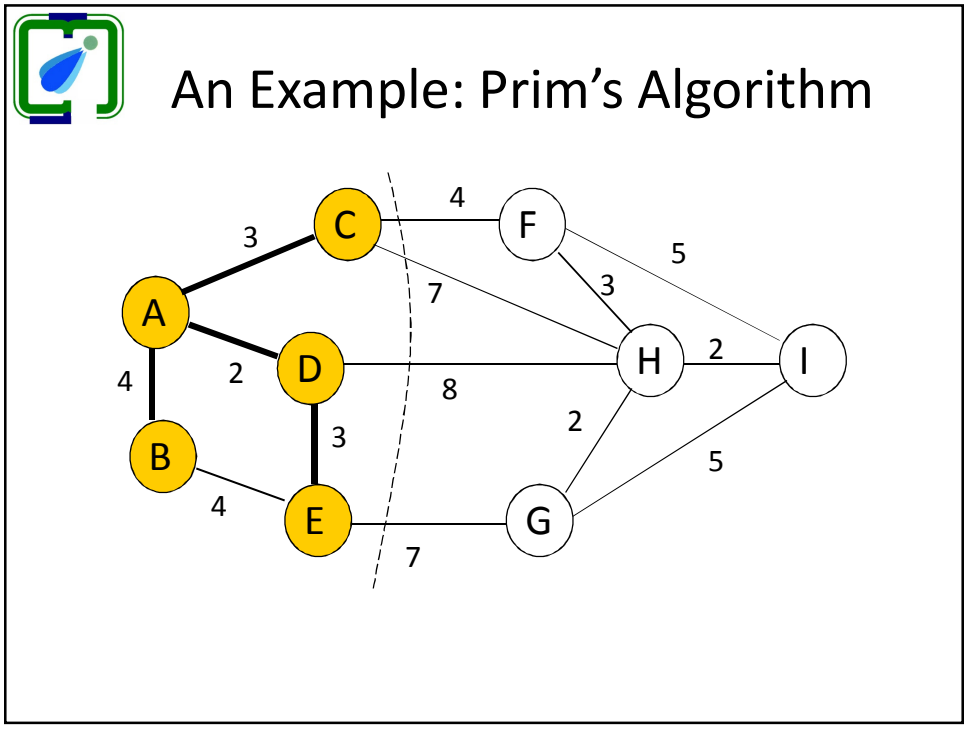


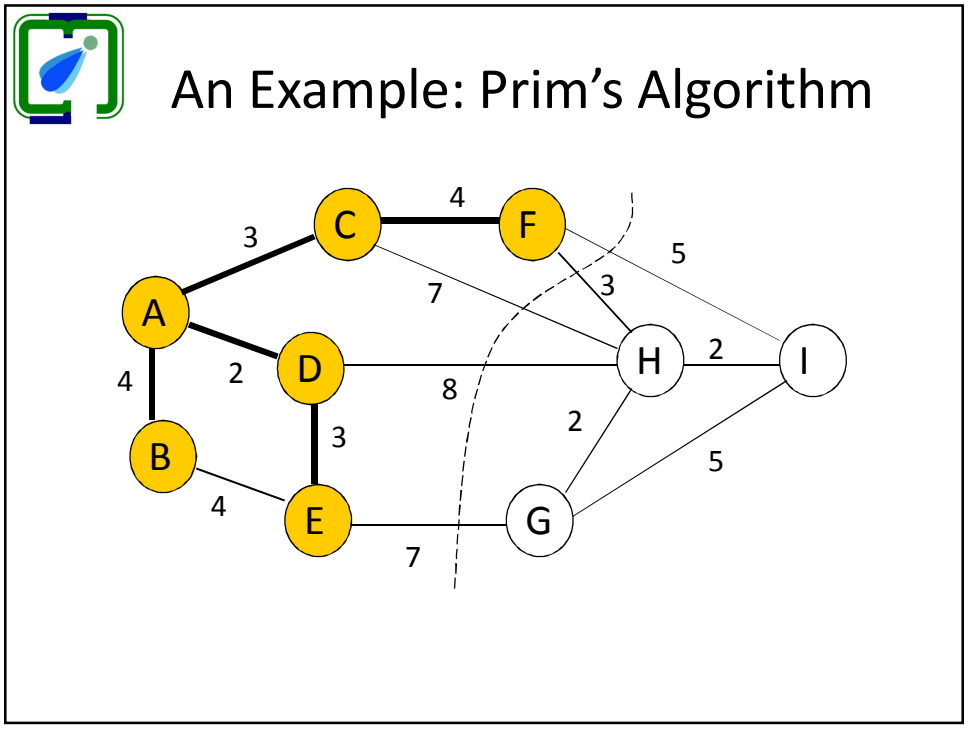

An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7

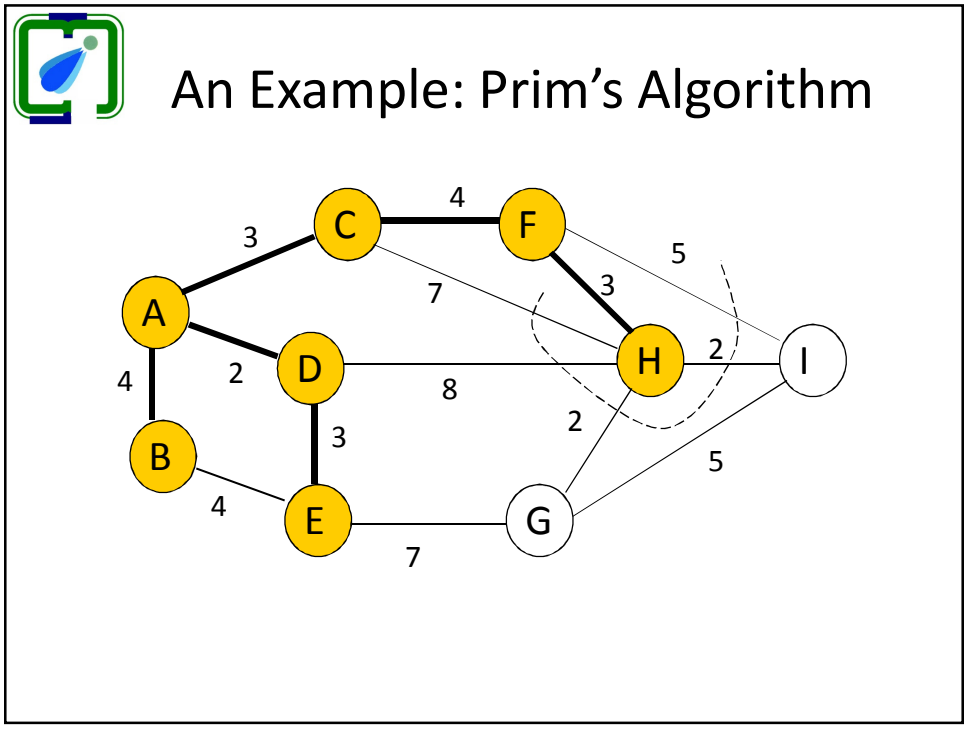

An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7

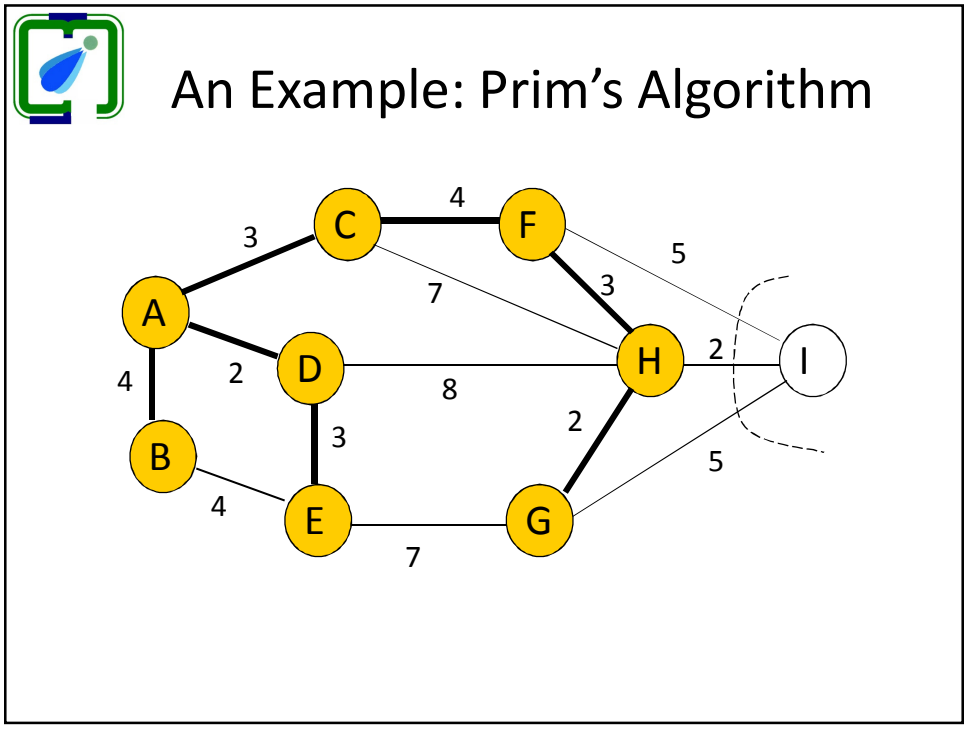

An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
2
4
8
2
3
5
4
7


An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
2
4
8
2
3
5
4
7

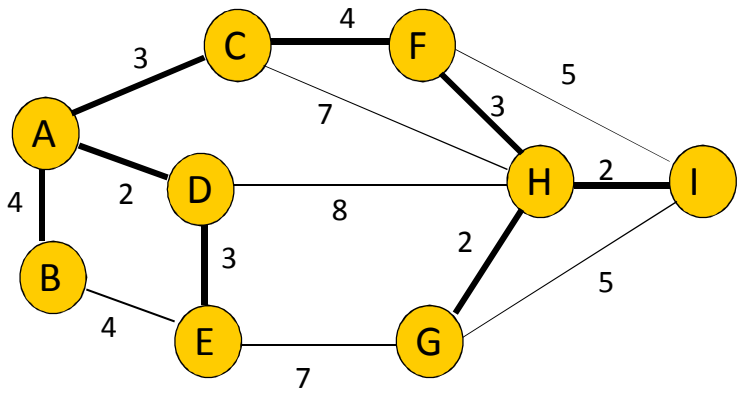

An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
2
4
8
2
3
5
4
7


An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
2
4
8
2
3
5
4
7


An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7


An Example: Prim's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7

An Example: Prim's Algorithm



Kruskal's MST Algorithm

- The Idea

Kruskal's MST Algorithm

MST-KRUSKAL(*G, w)*

1 *A* ← Ø

2 **sort (E)** //sort the edges of *E into nondecreasing order by weight w*

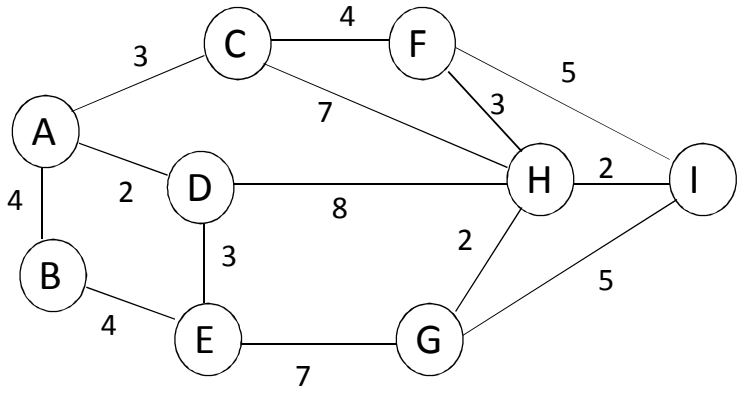
3 **for each edge (*u, v) Є E do***

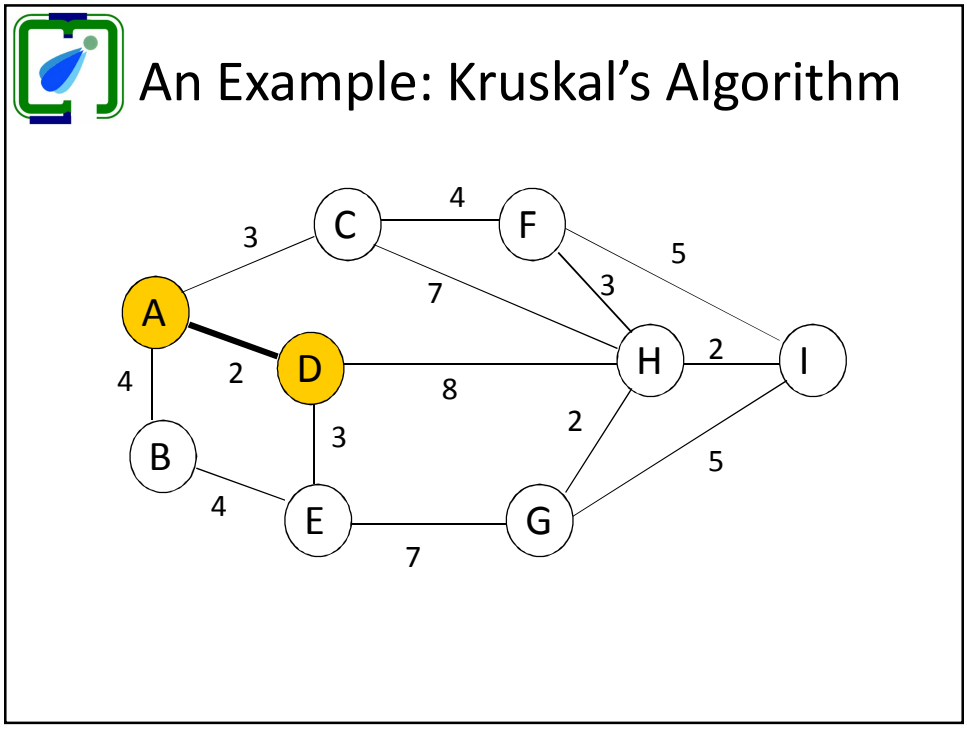
4 **if (A U (u, v) ≠ *cycle)***

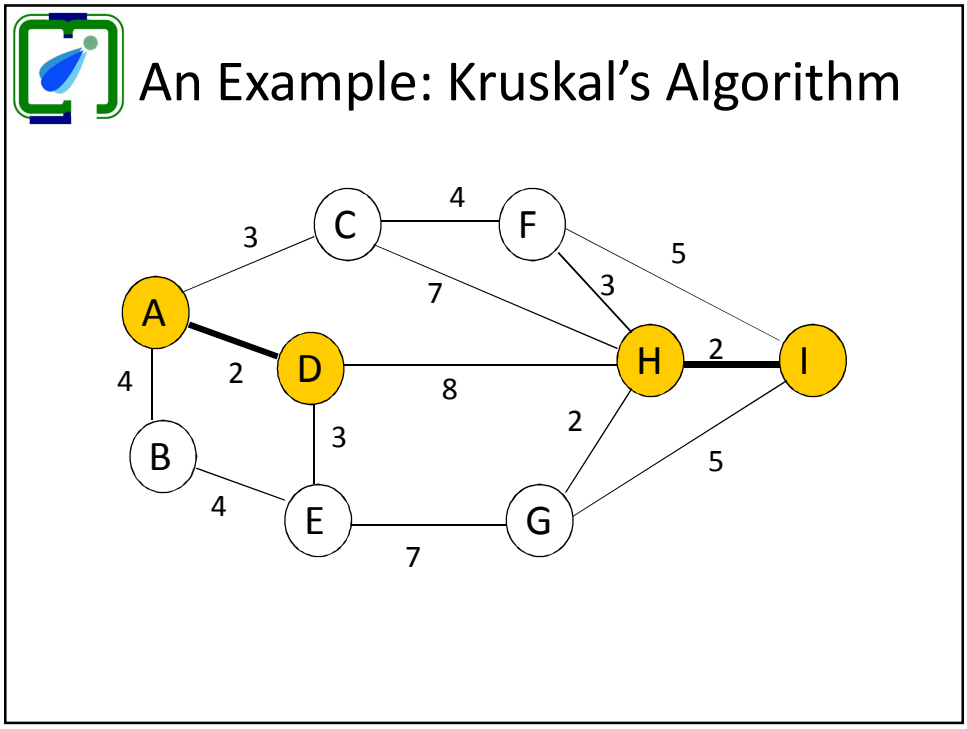
5 **then INSERT (*A, (u, v)*)**

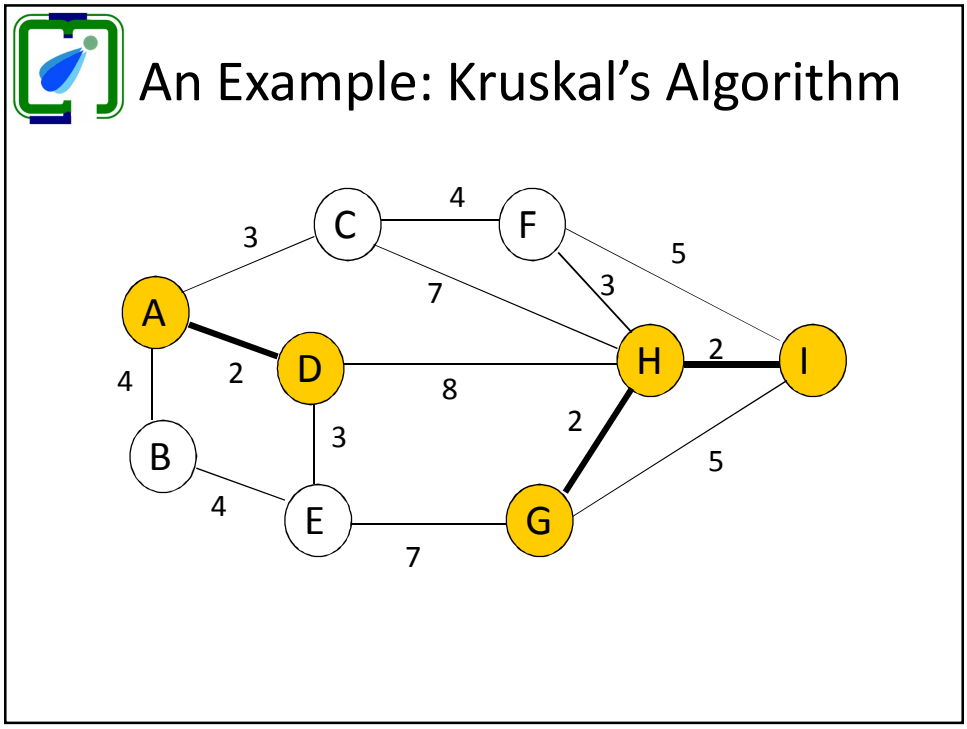
6 **return *A***

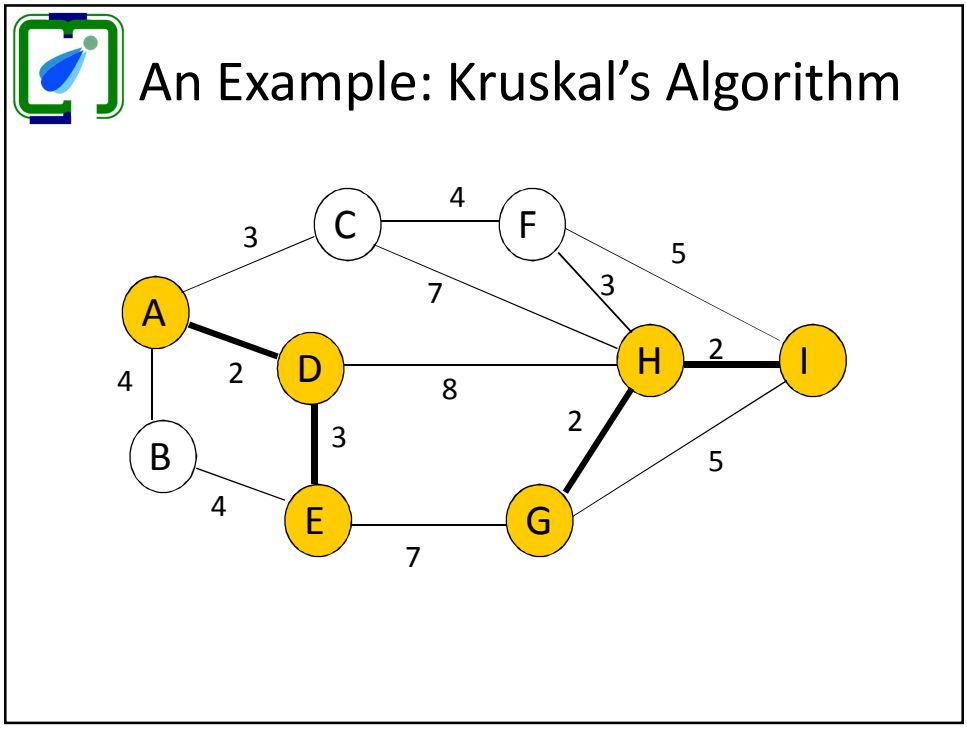
An Example: Kruskal's Algorithm

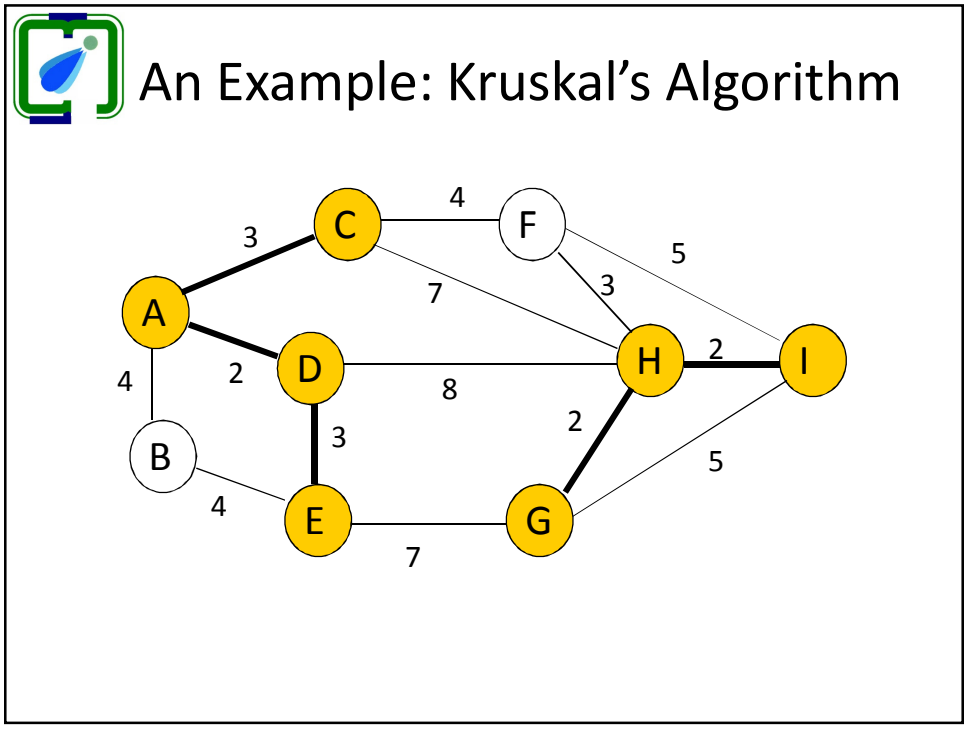


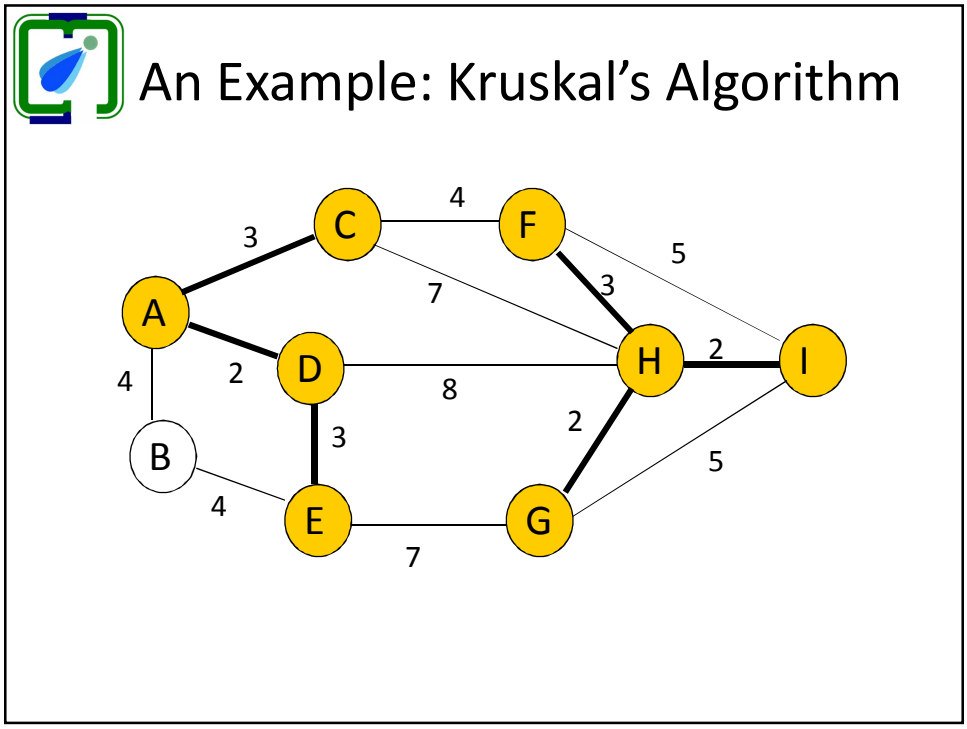

An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
2
4
2
8
3
2
5
4
7

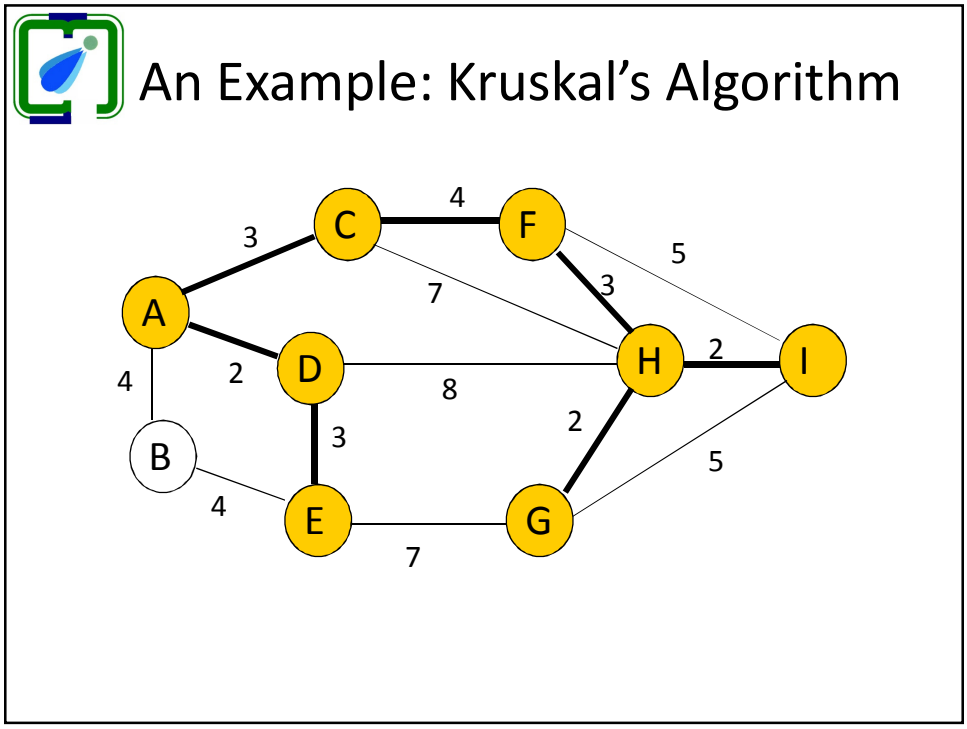

An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
2
4
2
8
3
2
5
4
7

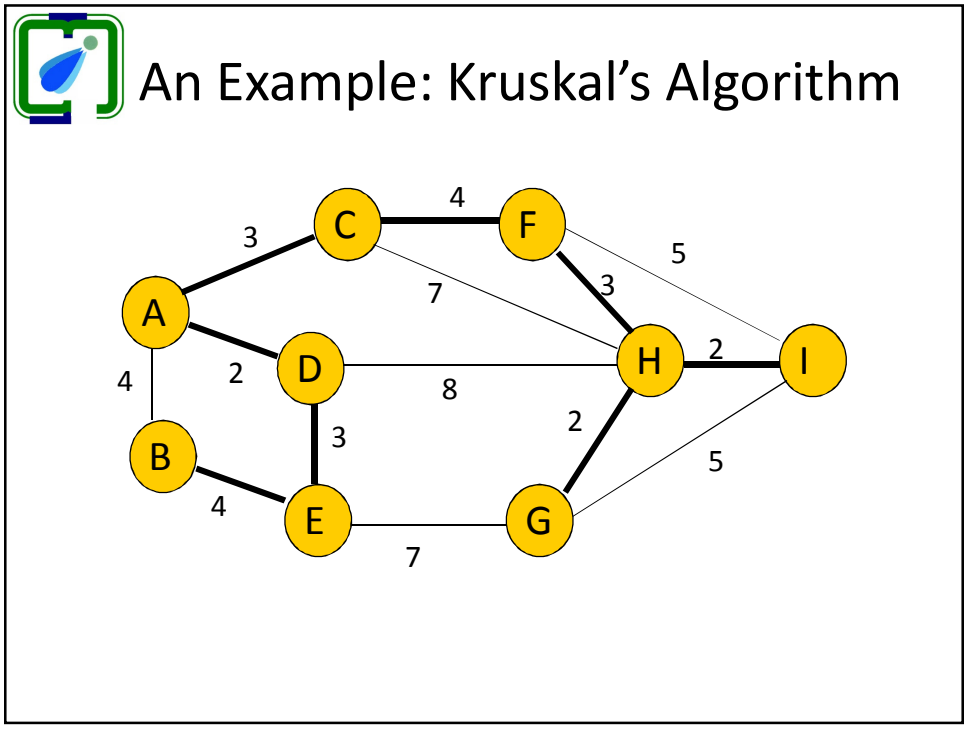

An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7


An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7

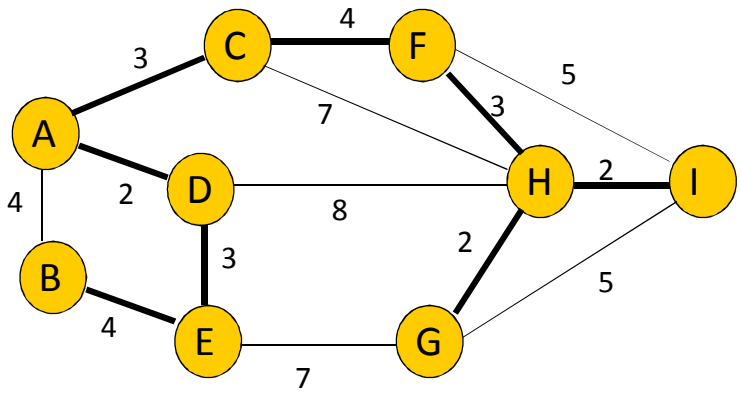

An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7


An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7


An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7


An Example: Kruskal's Algorithm
A
B
C
D
E
F
G
H
I
3
4
7
5
3
4
2
8
2
3
5
4
7

An Example: Kruskal's Algorithm



Minimum Spanning Trees

- A tree of the nodes of the graph with minimum total edge weight.
- Both Prim's and Kruskal's algorithms are examples of greedy approach of problem solving
- Applications
  - Reducing copper to connect multiple nodes in a electrical/electronic circuit
  - Minimizing network length (cable cost) to connect multiple routers/computers
  - and similar ….