# CMP(N)302: Design and Analysis of Algorithms

## Lecture 07: Minimum Spanning Trees

Ahmed Hamdy

Computer Engineering Department

Cairo University

Fall 2019

# Minimum Spanning Trees (MST)

- Problem arouse from many applications
- Given distances between cities, choose which roads to construct in order for all cities to be reachable with minimum construction cost.

| | Alexandria | Cairo | Matrouh | Aswan | Assiut | Hurghada |
|---|---|---|---|---|---|---|
| Alexandria | 0 | 220 | 320 | 1,080 | 580 | 680 |
| Cairo | 220 | 0 | 450 | 860 | 360 | 450 |
| Matrouh | 320 | 450 | 0 | 1,300 | 800 | 900 |
| Aswan | 1,080 | 860 | 1,300 | 0 | 500 | 400 |
| Assiut | 580 | 360 | 800 | 500 | 0 | 300 |
| Hurghada | 680 | 450 | 900 | 400 | 300 | 0 |

# Definition

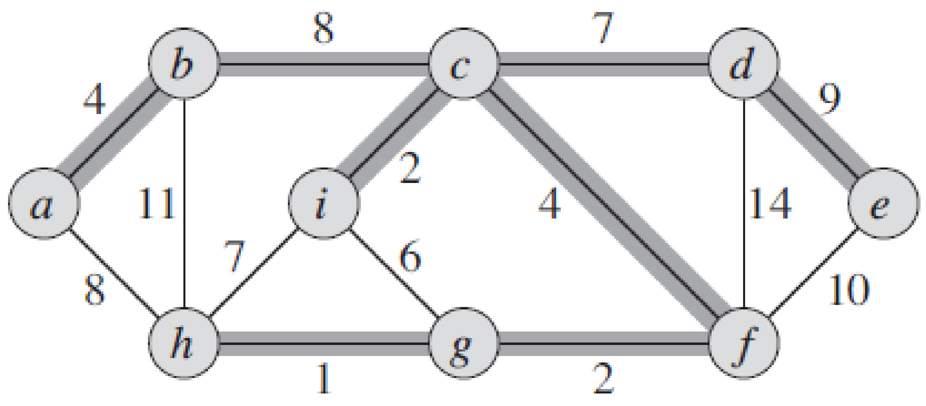



**Figure 23.1** A minimum spanning tree for a connected graph. The weights on edges are shown, and the edges in a minimum spanning tree are shaded. The total weight of the tree shown is 37. This minimum spanning tree is not unique: removing the edge $(b, c)$ and replacing it with the edge $(a, h)$ yields another spanning tree with weight 37.

- What is the use of this?!!
  - In electronic circuit design, we need to wire the electric components together

# Definition

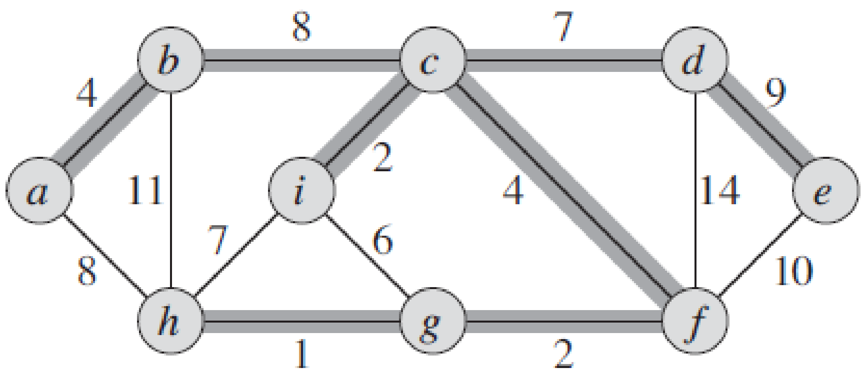



**Figure 23.1** A minimum spanning tree for a connected graph. The weights on edges are shown, and the edges in a minimum spanning tree are shaded. The total weight of the tree shown is 37. This minimum spanning tree is not unique: removing the edge $(b, c)$ and replacing it with the edge $(a, h)$ yields another spanning tree with weight 37.

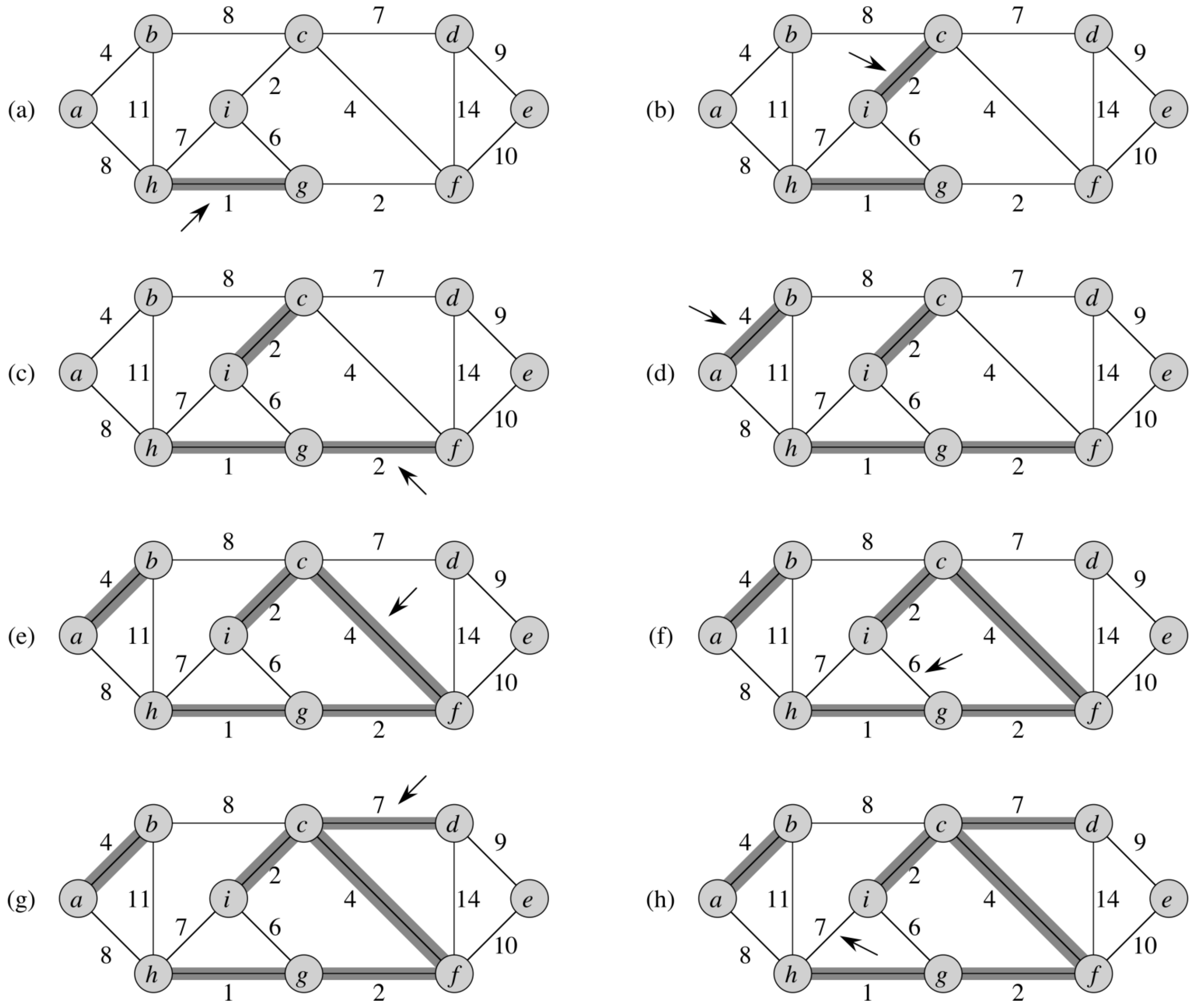
- How to write it as a definition for the problem?
  - Find an acyclic subset $T \subseteq E$ that connects all the vertices with minimum $w(T) = \sum_{(u,v)\in T} w(u, v)$

# Main concept

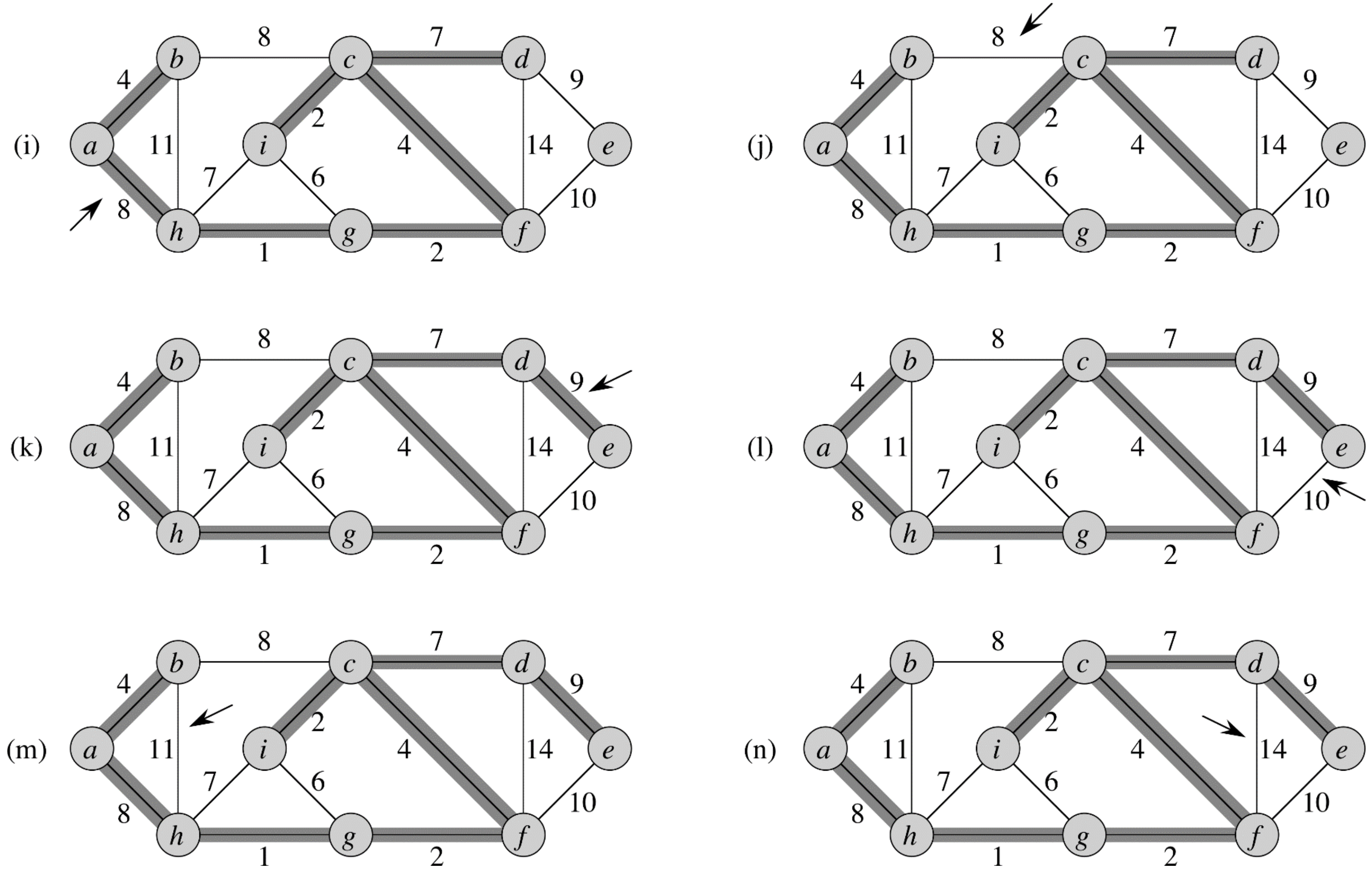
```
GENERIC-MST(G, w)
A = ∅
while A does not form a spanning tree
    find an edge (u, v) that is safe for A
    A = A ∪ {(u, v)}
return A
```

- Follows which approach??
  - Greedy approach

# Kruskal's algorithm



# Kruskal’s algorithm




- Each iteration: a) have a forest and b) add the least-weight safe edge connecting two different components

# Kruskal's algorithm

- Algorithm:

MST-KRUSKAL$(G, w)$

$O(1)$ → 1 $A = \emptyset$

$O(V)$ → 2 **for** each vertex $v \in G.V$

3 MAKE-SET$(v)$

$O(E \log E)$ → 4 sort the edges of $G.E$ into nondecreasing order by weight $w$

$O(E \log E)$ Lines 5-8 → 5 **for** each edge $(u, v) \in G.E$, taken in nondecreasing order by weight

6 **if** FIND-SET$(u) \neq$ FIND-SET$(v)$

7 $A = A \cup \{(u, v)\}$

8 UNION$(u, v)$

9 **return** $A$

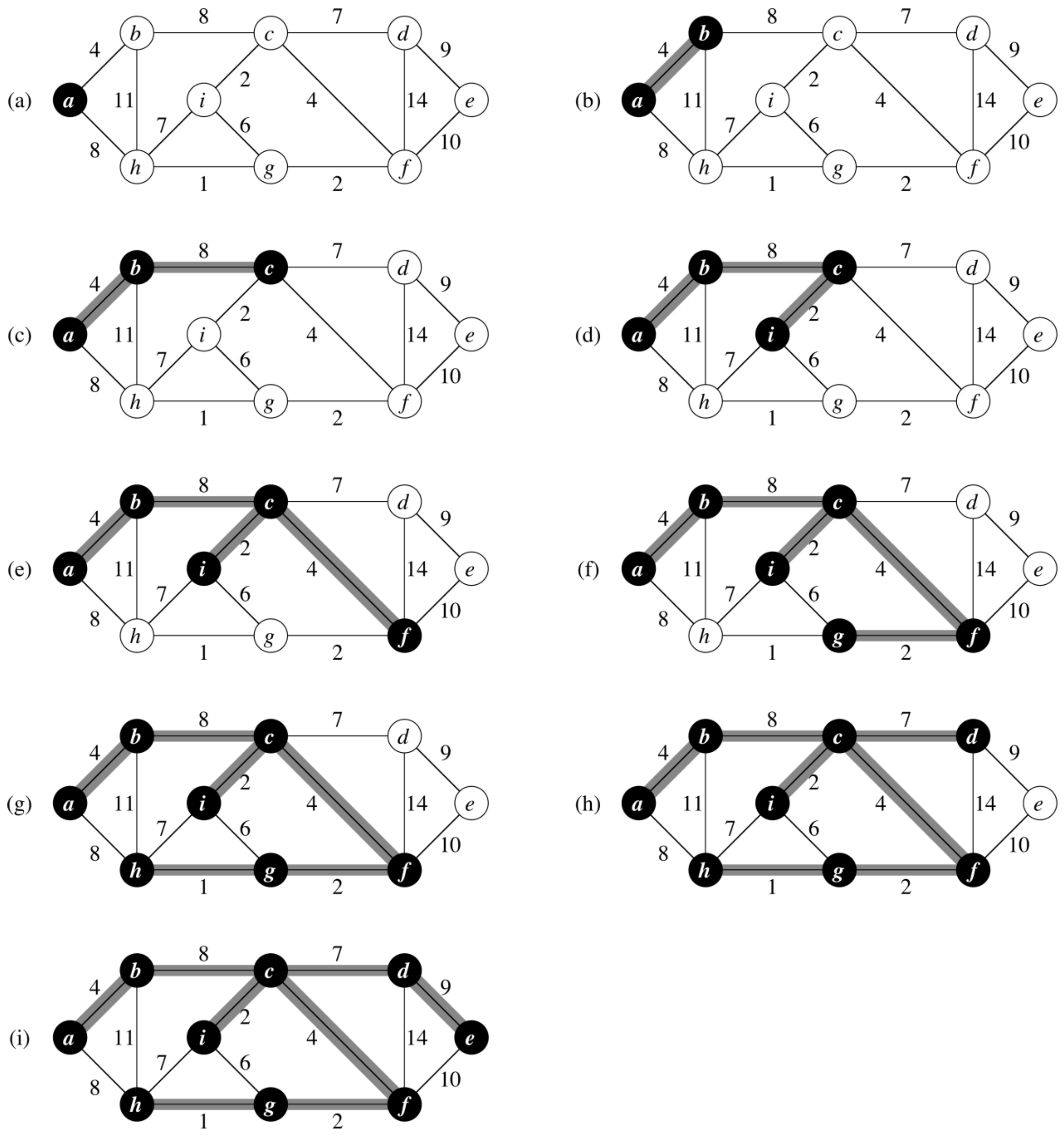
- Complexity: $O(E \log E) = O(E \log V)$
- Read disjoint-sets (Chapter 21)

# Prim's algorithm

During each iteration:

a) have a tree

b) add the least-weight safe edge connecting the tree to vertex not in tree



# Prim's algorithm

- Algorithm:

MST-PRIM$(G, w, r)$

| | | |
|---|---|---|
| $O(V)$ → | 1 | **for** each $u \in G.V$ |
| | 2 |     $u.key = \infty$ |
| | 3 |     $u.\pi = \text{NIL}$ |
| | 4 | $r.key = 0$ |
| | 5 | $Q = G.V$ |
| $O(V)$ → | 6 | **while** $Q \neq \emptyset$ |
| $O(\log V)$ → | 7 |     $u = \text{EXTRACT-MIN}(Q)$ |
| $O(E)$ → | 8 |     **for** each $v \in G.Adj[u]$ |
| Lines 6 – 8 | 9 |         **if** $v \in Q$ and $w(u, v) < v.key$ |
| | 10 |             $v.\pi = u$ |
| $O(\log V)$ → | 11 |             $v.key = w(u, v)$ |

| Procedure | Binary heap (worst-case) | Fibonacci heap (amortized) |
|---|---|---|
| MAKE-HEAP | $\Theta(1)$ | $\Theta(1)$ |
| INSERT | $\Theta(\lg n)$ | $\Theta(1)$ |
| MINIMUM | $\Theta(1)$ | $\Theta(1)$ |
| EXTRACT-MIN | $\Theta(\lg n)$ | $O(\lg n)$ |
| UNION | $\Theta(n)$ | $\Theta(1)$ |
| DECREASE-KEY | $\Theta(\lg n)$ | $\Theta(1)$ |
| DELETE | $\Theta(\lg n)$ | $O(\lg n)$ |

- Complexity: $O(V \log V + E \log V) = O(E \log V)$
  - Using Fibonacci heaps: $O(E + V \log V)$