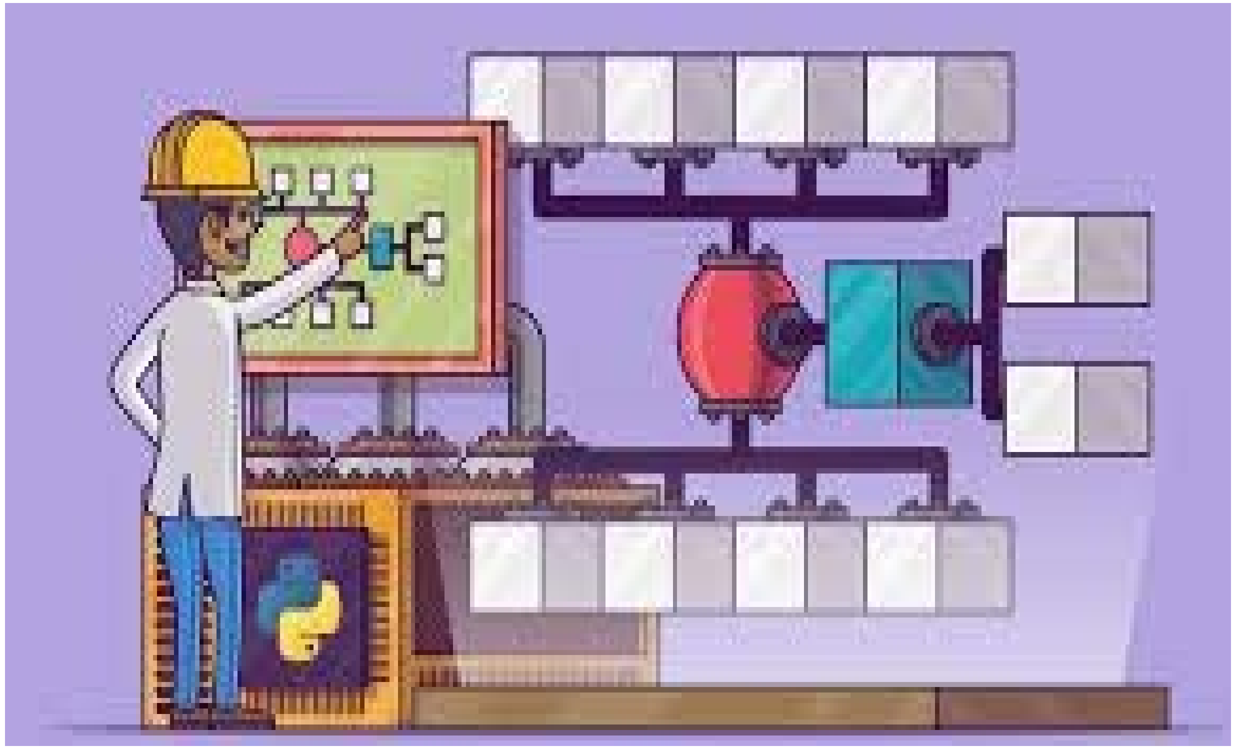
ساختمان داده ها



مدرس:
سمانه حسینی سمنانی

دانشگاه صنعتی اصفهان- دانشکده برق و کامپیوتر

Elementary Graph Algorithms

- Graph representation
- graph-searching algorithm
  - breadth-first search
  - depth-first search
- minimum-spanning-tree
  - Kruskal
  - Prim
  - Sollin

Growing a minimum spanning tree

$G = (V, E)$

Weight function: $w : E \to \mathbb{R}$,

GENERIC-MST(G, w)

```
1  A = ∅
2  while A does not form a spanning tree
3        find an edge (u, v) that is safe for A
4        A = A ∪ {(u, v)}
5  return A
```

The tricky part is, of course, finding a safe edge in line 3.

The algorithms of Kruskal, Prim and Sollin

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

each use a specific rule to determine a safe edge in line 3 of GENERIC-MST.

Kruskal’s algorithm

- Kruskal’s algorithm finds a safe edge to add to the growing forest by finding, of all the edges that connect any two trees in the forest, an edge (u, v) of least weight.

```
MST-KRUSKAL(G, w)
A = Ø
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

- The operation FIND-SET(u) returns a representative element from the set that contains u

Kruskal’s algorithm

- Kruskal’s algorithm finds a safe edge to add to the growing forest by finding, of all the edges that connect any two trees in the forest, an edge (u, v) of least weight.

```
MST-KRUSKAL(G, w)
A = Ø
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

Thus, we can determine whether two vertices u and belong to the same tree by testing whether FIND-SET(u) equals FIND-SET(v)

- The operation FIND-SET(u) returns a representative element from the set that contains u

Kruskal’s algorithm

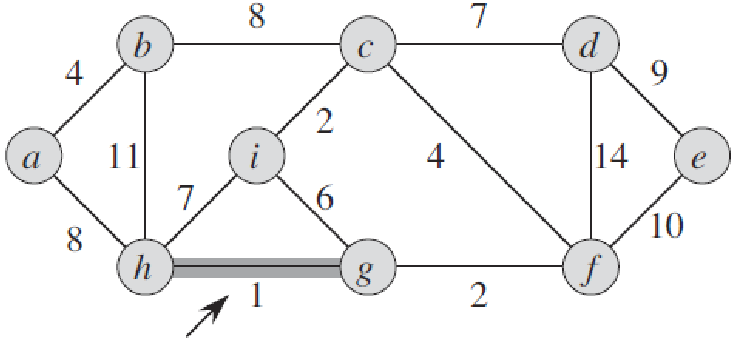
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3      MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6      if FIND-SET(u) ≠ FIND-SET(v)
7          A = A ∪ {(u, v)}
8          UNION(u, v)
9  return A
```

Lines 1–3 initialize the set A to the empty set and create |V| trees, one containing each vertex

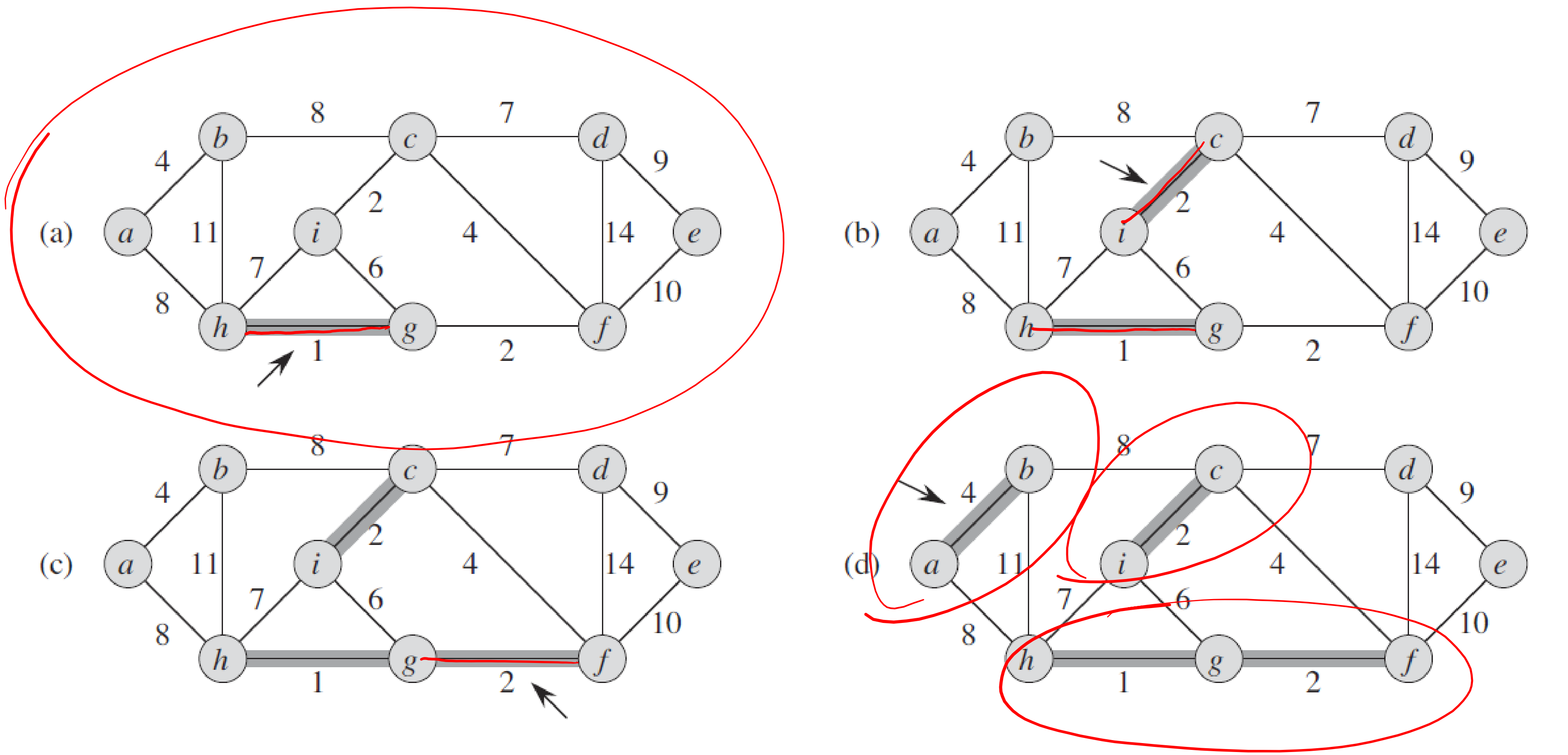
for each edge (u, v) whether the endpoints u and v belong to the same tree.

If they do, then the edge (u, v) cannot be added to the forest without creating a cycle, and the edge is discarded.

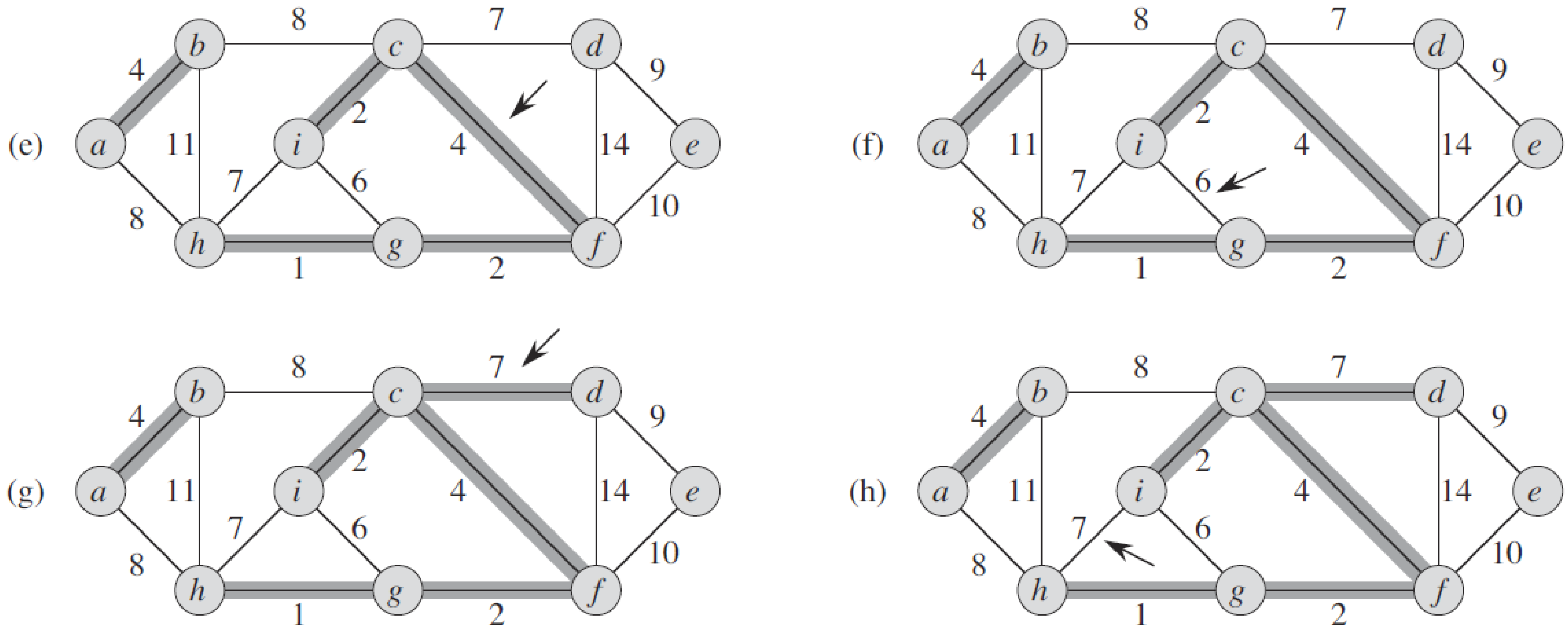
(a)



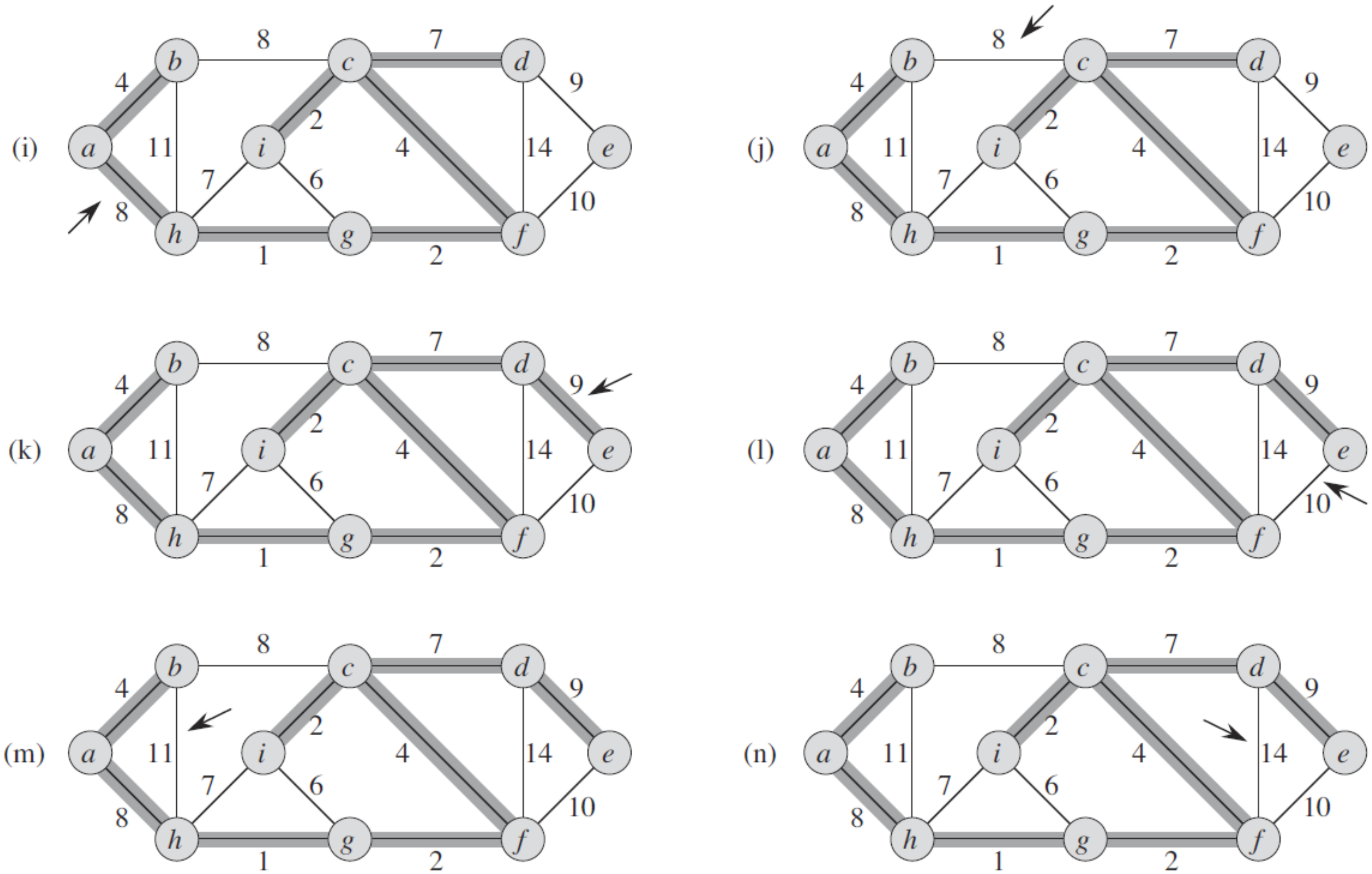
Kruskal’s algorithm



Kruskal's algorithm



Kruskal’s algorithm



Running time of Kruskal's algorithm

- Initializing the set A in line 1 takes O(1) time
- time to sort the edges in line 4 is $O(E \lg E)$
- The for loop of lines 5–8 : $O(E \log E)$
- running time of Kruskal's algorithm is $O(E \log E)$
- $|E| < |V|^2$

$\log |E| < \log |V|^2$

$2 \log |V|$

- $\lg |E| = O(\lg V)$
- running time of Kruskal's algorithm is

$$O(E \lg V).$$

MST-KRUSKAL(G, w)

1 $A = \emptyset$
2 **for** each vertex $v \in G.V$
3 MAKE-SET(v)
4 sort the edges of $G.E$ into nondecreasing order by weight w
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

$O(V)$

if((u,v) is safe)

log E

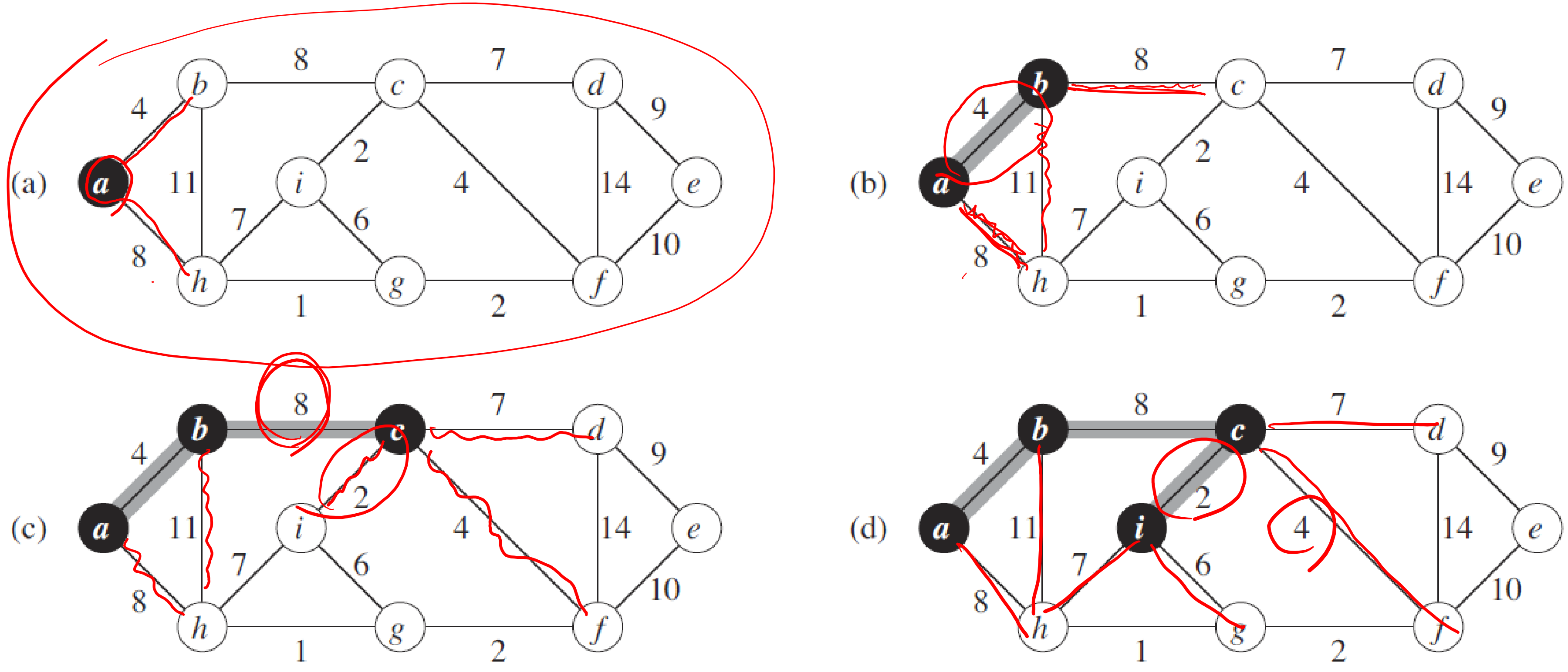
$|V|(V-1)$

Prim’s algorithm

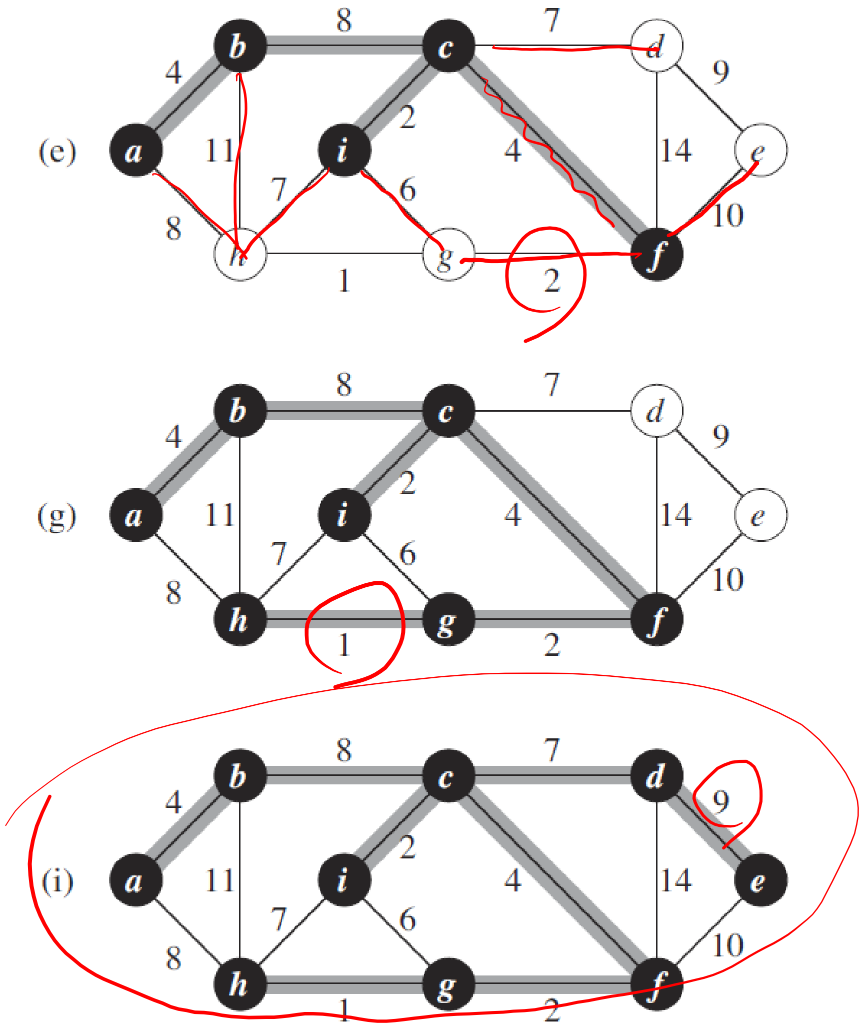
- Like Kruskal’s algorithm, Prim’s algorithm is a special case of the generic minimum- spanning-tree method
- Prim’s algorithm has the property that the edges in the set A always form a single tree.

Prim's algorithm

- The tree starts from an arbitrary root vertex r and grows until the tree spans all the vertices in V .

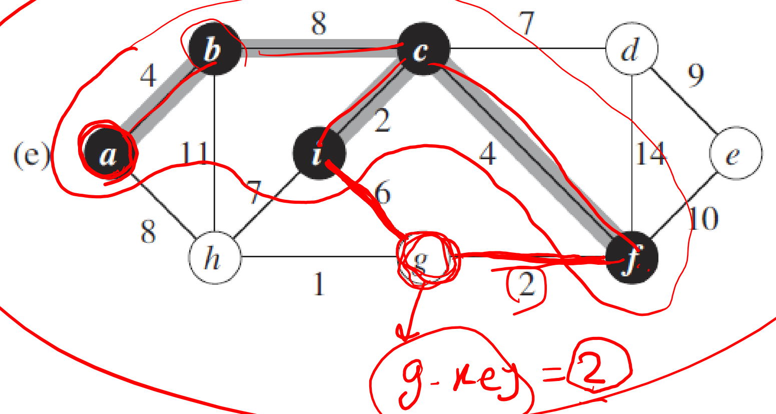


Prim’s algorithm



Prim's algorithm

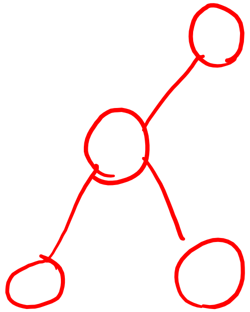
- All vertices that are *not* in the tree reside in a min-priority queue Q based on a *key* attribute
- For each vertex , the attribute $v.key$ is the minimum weight of any edge **connecting to a vertex in the tree**;
- $v.key = \infty$ if there is no such edge
- $V.\pi$ names the parent of v in the tree.



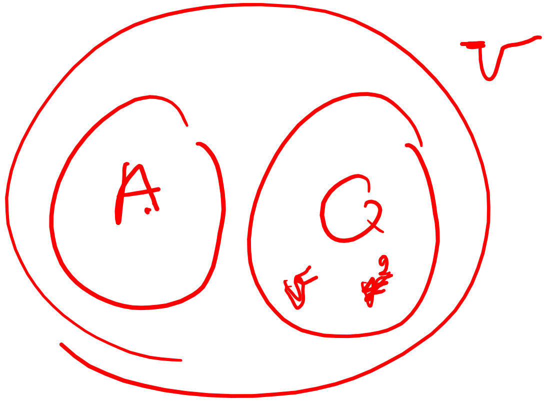

```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```


سمانه حسینی سمنانی
هیات علمی دانشکده برق و کامپیوتر- دانشگاه صنعتی اصفهان

Prim’s algorithm



$A = V - Q$



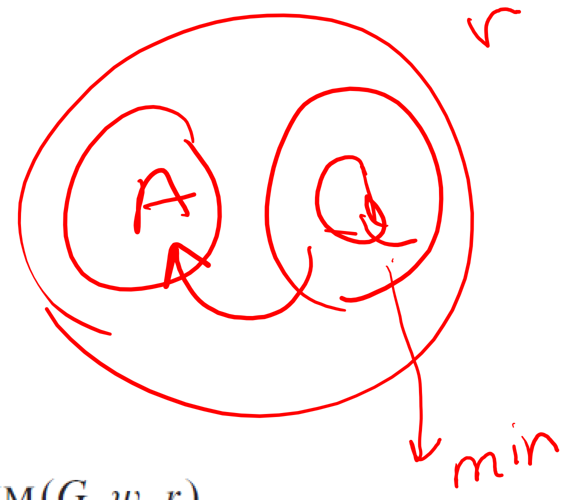
Prior to each iteration of the **while** loop of lines 6–11,

1. $A = \{(v, v.\pi) : v \in V - \{r\} - Q\}$.
2. The vertices already placed into the minimum spanning tree are those in $V - Q$.
3. For all vertices $v \in Q$, if $v.\pi \neq \text{NIL}$, then $v.key < \infty$ and $v.key$ is the weight of a light edge $(v, v.\pi)$ connecting v to some vertex already placed into the minimum spanning tree.

MST-PRIM(G, w, r)

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

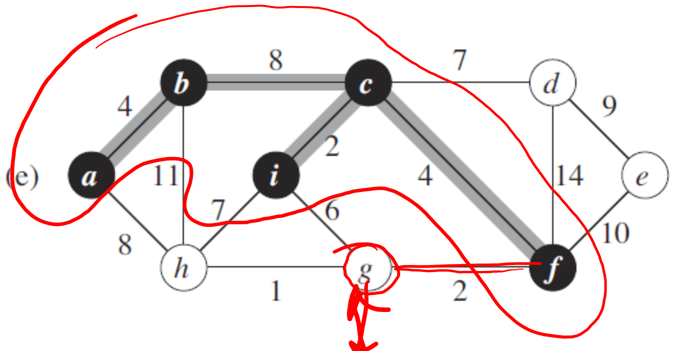
Prim's algorithm

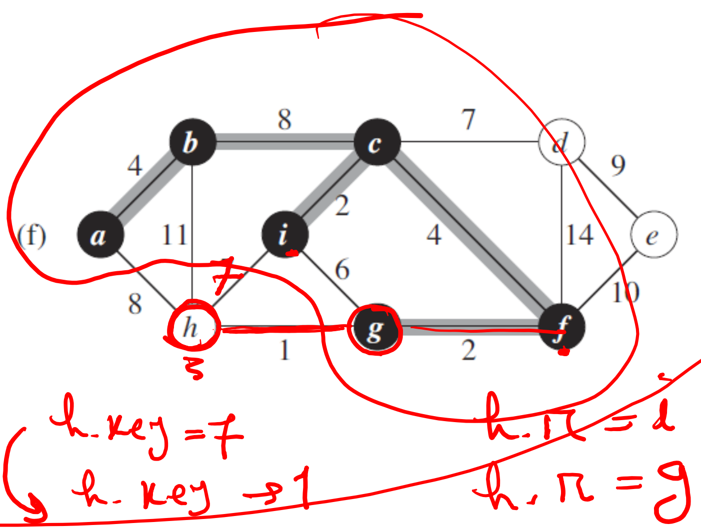


- Line 7 identifies a vertex $u \in Q$ incident on a light edge that crosses the cut $(V - Q, Q)$
- The **for** loop of lines 8–11 updates the *key* and attributes

  of every vertex adjacent to u but not in the tree,

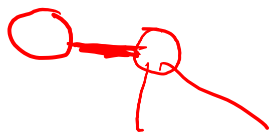
```
MST-PRIM(G, w, r)
1  for each u ∈ G.V
2      u.key = ∞
3      u.π = NIL
4  r.key = 0
5  Q = G.V
6  while Q ≠ ∅
7      u = EXTRACT-MIN(Q)
8      for each v ∈ G.Adj[u]
9          if v ∈ Q and w(u, v) < v.key
10             v.π = u
11             v.key = w(u, v)
```





The running time of Prim’s algorithm

- If we implement Q as a binary min-heap, we can use the BUILD-MIN-HEAP procedure to perform lines 1–5 in O(V) time.
- The body of the **while** loop executes |V | times,
- and since each EXTRACT-MIN operation takes $O(logV)$ time,
- the total time for all calls to EXTRACT-MIN is $O(V\ logV)$.
- The for loop in lines 8–11 executes O(E) times altogether,
- since the sum of the lengths of all adjacency lists is $2|E|$.



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

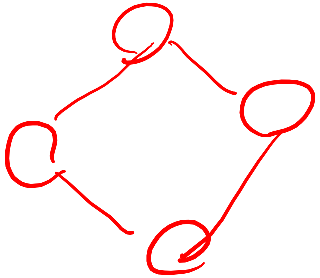
O(V)

|V|

V log V

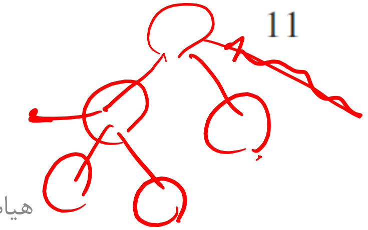
The running time of Prim's algorithm

- Within the **for** loop, we can implement the test for membership in Q in line 9 in constant time by keeping a bit for each vertex that tells whether or not it is in Q, and updating the bit when the vertex is removed from Q.
- The assignment in line 11 involves an implicit DECREASE-KEY in min-heap
- which a binary min-heap supports in O(lg V) time
- total time for Prim's algorithm:



$$O(V \lg V + E \lg V) = O(E \lg V)$$

the same as for our implementation of Kruskal's algorithm



MST-PRIM(G, w, r)

```
1   for each u ∈ G.V
2       u.key = ∞
3       u.π = NIL
4   r.key = 0
5   Q = G.V
6   while Q ≠ ∅                      v. log v
7       u = EXTRACT-MIN(Q)
8   E   for each v ∈ G.Adj[u]
9           if v ∈ Q and w(u, v) < v.key
10              v.π = u
11    log(v)    v.key = w(u, v)
```

Steps of Sollin's Algorithm

1. Write all vertices of a connected graph.

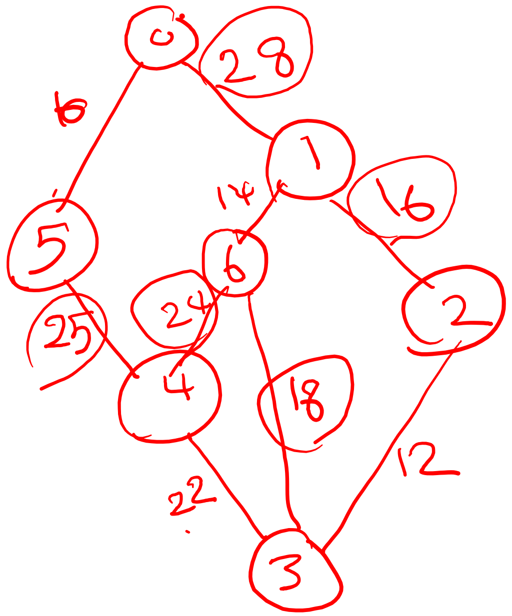
2. Highlight the cheapest outgoing edge of all vertices.

3. Choose only one cheapest edge for each vertex.

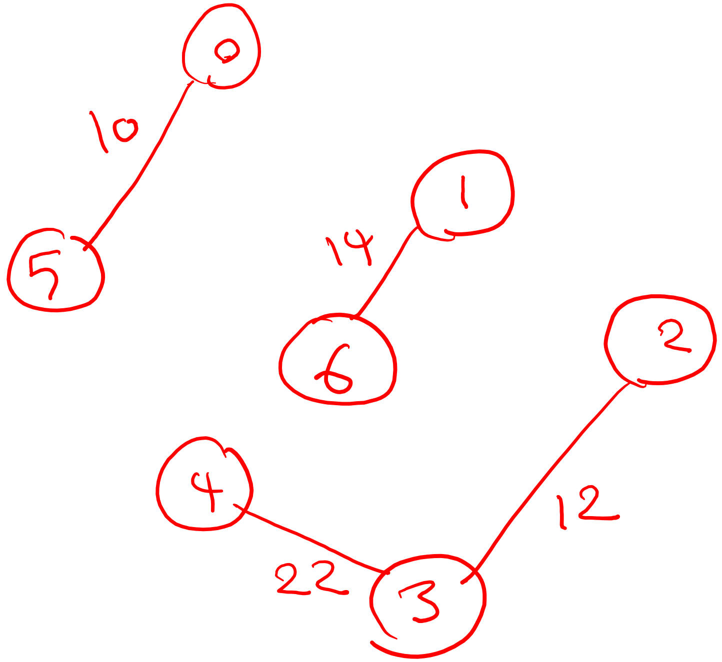
4. Repeat the algorithm for each sub graph (each differently colored set). This time, for each node, choose the cheapest edge **outside of the sub-graph.**

5. If an edge is already selected then skip it.

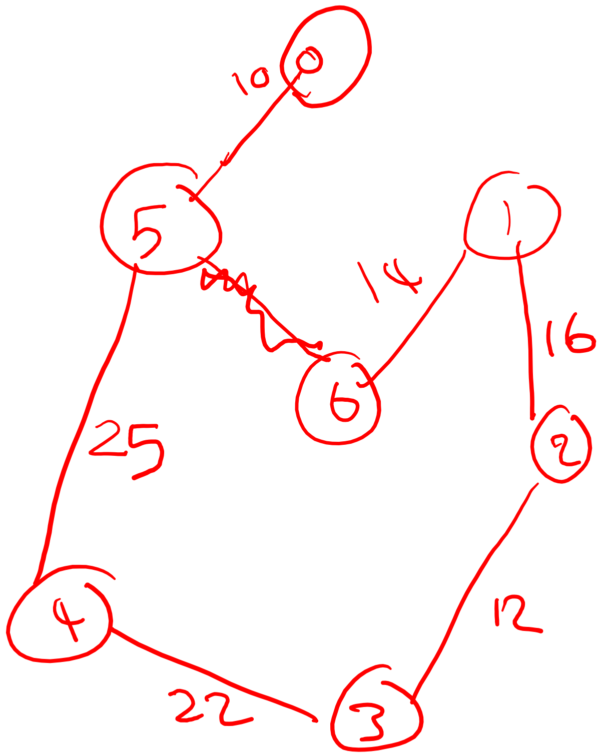
Example



0
28
1
16
14
5
6
2
24
25
4
18
12
22
3



0
10
5
1
14
6
2
4
12
22
3



10
0
5
1
14
6
16
25
2
4
12
22
3

Thank you