MST
KRUSKAL / PRIM

طراحی الگوریتم‌ها – جلسه نوزدهم و بیستم

Introduction to Algorithm

استاد: جوانمردی

فصل ۲۳ کتاب – درخت پوشای کمینه

- ۲۳.۱: رشد درخت پوشای کمینه
- ۲۳.۲: الگوریتم‌های Kruskal و Prim

23 Minimum Spanning Trees *624*

23.1 Growing a minimum spanning tree *625*

23.2 The algorithms of Kruskal and Prim *631*

تعریف درخت پوشا کمینه یا MST

- درخت پوشا یا Spanning tree

با در اختیار داشتن یک گراف متصل با n گره، یک درخت پوشا عبارت است از درختی با n-1 یال که همه n گره را به هم متصل کند

آن n گره و n-1 یال یک زیرگراف متصل را تشکیل می‌دهد

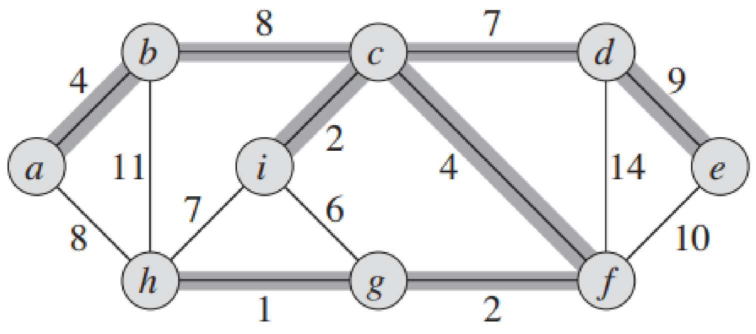
- درخت پوشا یکتا نیست و ممکن است چندین درخت پوشا بتوان تعریف کرد

- درخت پوشای کمینه:

$G = (V, E)$

$(u, v) \in E \longrightarrow w(u, v)$

$$w(T) = \sum_{(u,v) \in T} w(u, v)$$



ایجاد یک درخت پوشای کمینه

- روش حریصانه برای ایجاد یک درخت پوشای کمینه برای گراف وزن‌دار $G = (V, E)$ با $w : E \to \mathbb{R}$

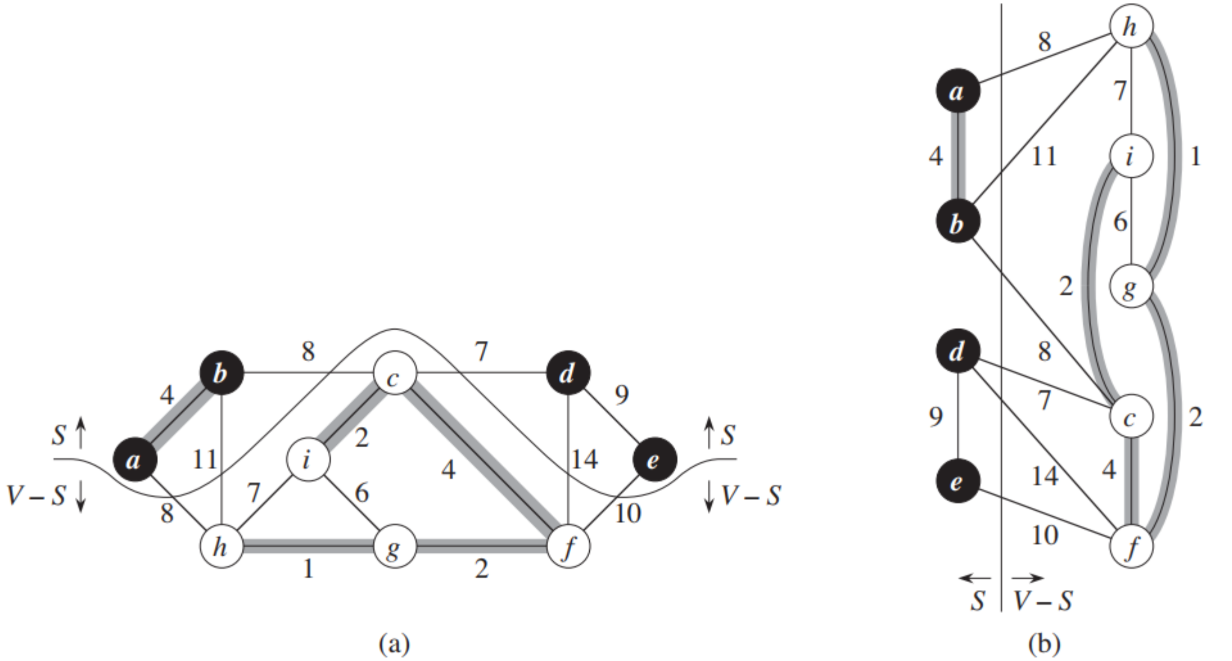
Prior to each iteration, A is a subset of some minimum spanning tree.

در هر مرحله یک یال (u,v) را بگونه ای انتخاب میکنیم که $A \cup \{(u, v)\}$ کماکان یک MST باشد

```
GENERIC-MST(G, w)
1  A = Ø
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
```

ایجاد یک درخت پوشای کمینه

- برخی تعاریف مورد نیاز برای ادامه: برش گراف $G = (V, E)$ با نماد $cut\ (S, V - S)$



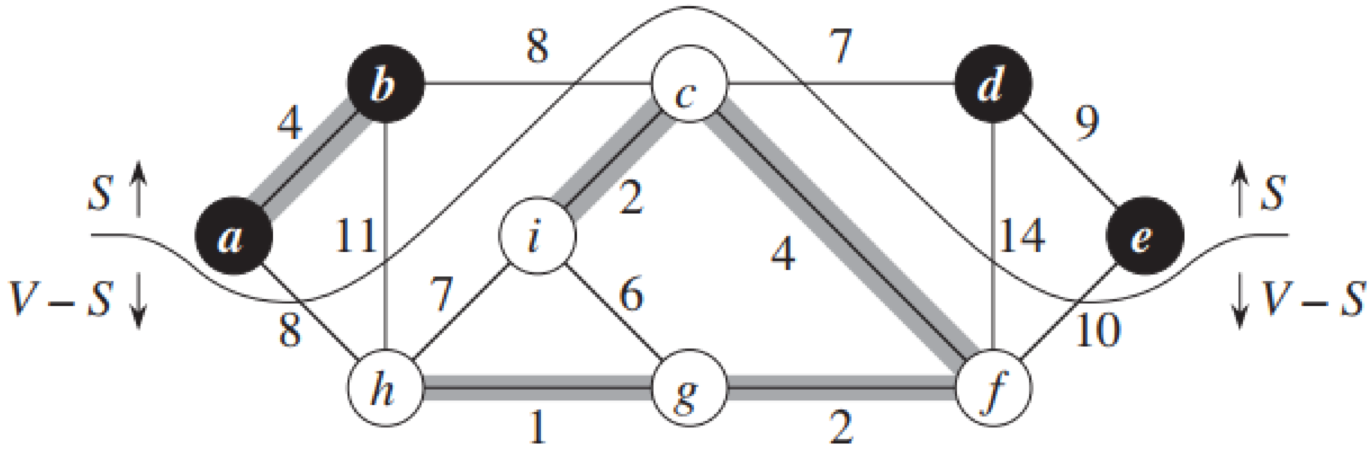
an edge $(u, v) \in E$ ***crosses*** the cut $(S, V - S)$ if one of its endpoints is in S and the other is in $V - S$

a cut ***respects*** a set A of edges if no edge in A crosses the cut

An edge is a ***light edge*** crossing a cut if its weight is the minimum of any edge crossing the cut

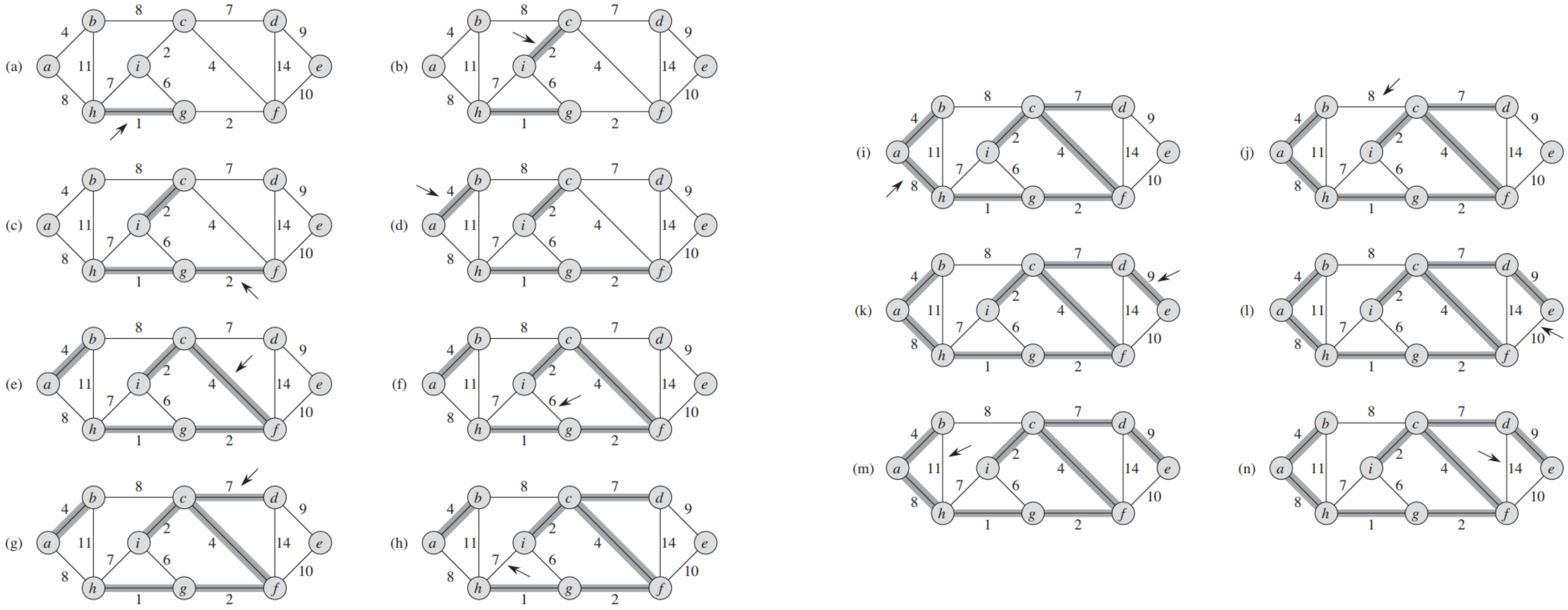
ایجاد یک درخت پوشای کمینه

- قضیه: اگر $G(V,E)$ یک گراف بدون‌جهت وزن‌دار با وزن‌های w روی E باشد. فرض کنید A یک زیرمجموعه‌ای از یال‌های E باشد که مشمول در یکی از MSTهای G باشد. حال فرض کنید برش $(S,V-S)$ یک برش از G باشد که به A احترام می‌گذارد، و (u,v) یک یال سبک گذرکننده از $(S,V-S)$ باشد. در این صورت یال (u,v) یک یال امن برای A خواهد بود.



الگوریتم Kruskal برای ایجاد MST

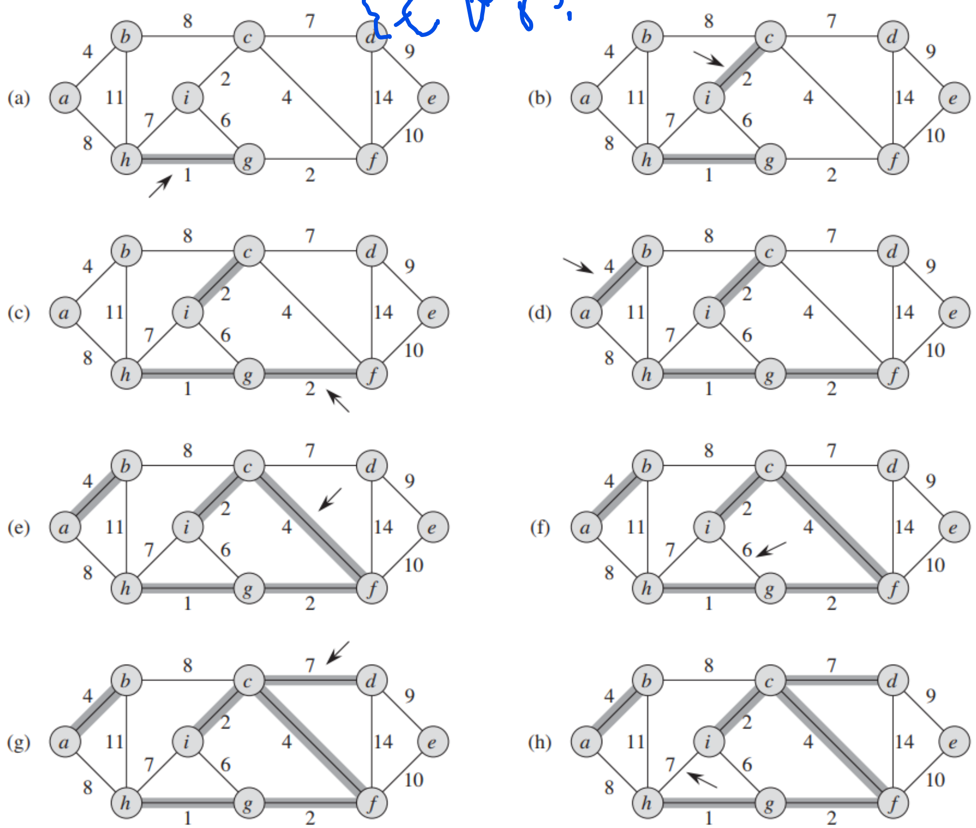
- یال‌های گراف را بر اساس وزن آنها بصورت صعودی مرتب می‌کنیم
- یال‌ها با وزن کمتر را به MST اضافه میکنیم به شرط اینکه حلقه ایجاد نشود
- هرگاه v-1 یال اضافه کردیم الگوریتم به پایان می‌رسد

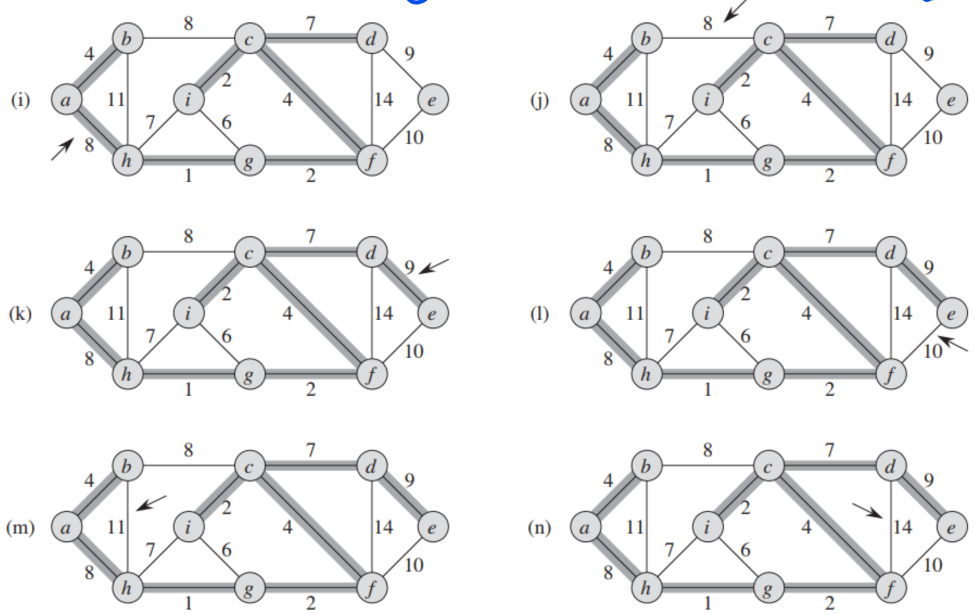


الگوریتم Kruskal برای ایجاد MST

- یال‌های گراف را بر اساس وزن آنها بصورت صعودی مرتب می‌کنیم $\Theta(|E|\lg|E|)$
- یال‌ها با وزن کمتر را به MST اضافه میکنیم به شرط اینکه حلقه ایجاد نشود $\Theta(|E|+|V|)=\Theta(V)$ $\rightarrow \Theta(|V|\cdot|E|)$
- هرگاه v-1 یال اضافه کردیم الگوریتم به پایان می‌رسد

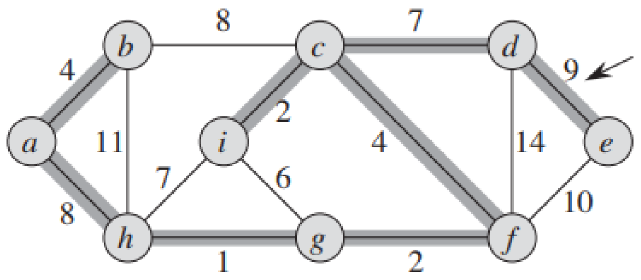
$\rightarrow \Theta(|E|\lg|E|)+\Theta(|V||E|)$



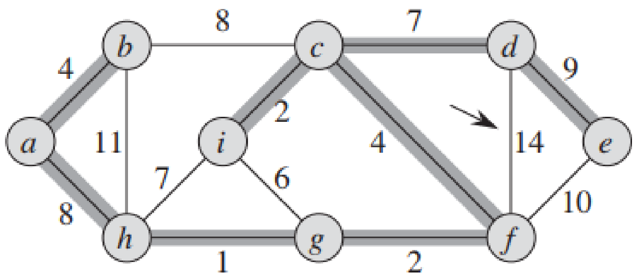


الگوریتم Kruskal برای ایجاد MST

(k)



(n)

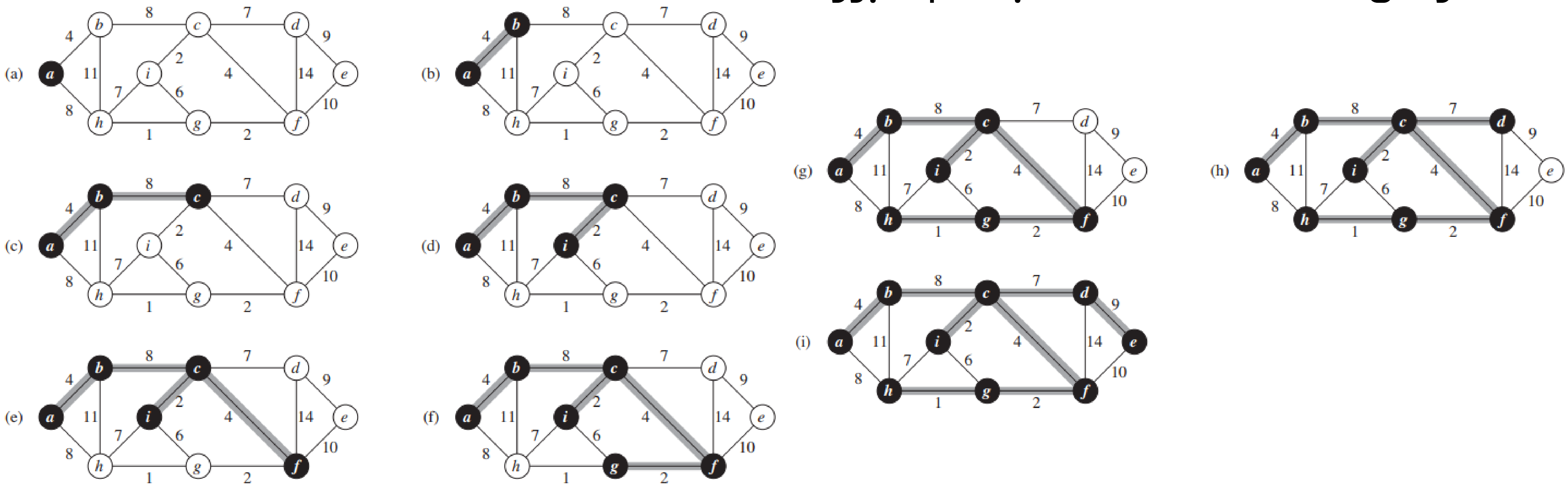


MST-KRUSKAL(G, w)

$A = \emptyset$
for each vertex $v \in G.V$
  MAKE-SET(v)
sort the edges of $G.E$ into nondecreasing order by weight w
for each edge $(u, v) \in G.E$, taken in nondecreasing order by weight
  if FIND-SET$(u) \neq$ FIND-SET(v)
    $A = A \cup \{(u, v)\}$
    UNION(u, v)
return A

الگوریتم Prim برای ایجاد MST

1. از بین گره‌های گراف یک گره v را به دلخواه انتخاب کرده قرار میدهیم $S = \{v\}$
2. از بین یال‌های گذرکننده از $(S, V - S)$ یال با وزن کمینه را انتخاب و به MST اضافه می‌کنیم
3. راس طرف V-S را به S اضافه میکنیم
4. تا زمانی که S=V نشده است به گام ۲ برو

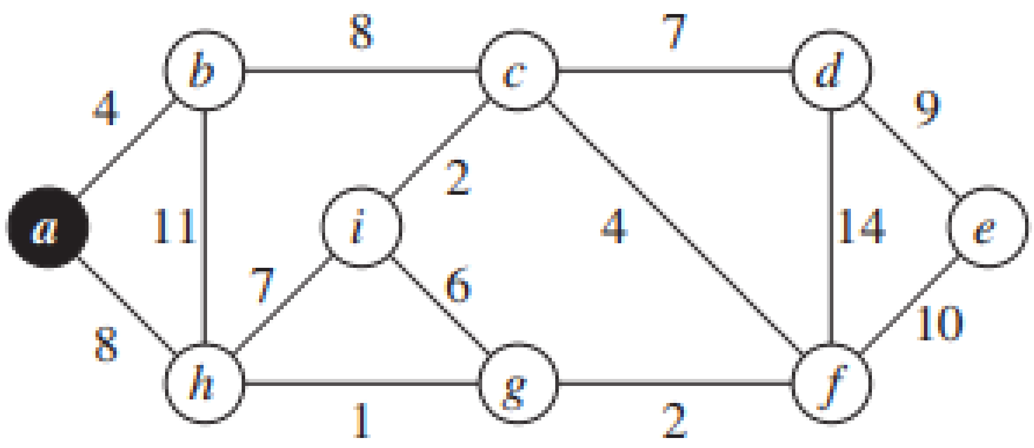


الگوریتم Prim برای ایجاد MST

1. از بین گره‌های گراف یک گره v را به دلخواه انتخاب کرده قرار میدهیم $S = \{v\}$
2. از بین یال‌های گذرکننده از $(S, V - S)$ یال با وزن کمینه را انتخاب و به MST اضافه می‌کنیم

```
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)
```

(a)



تحلیل زمان اجرای Prim با binary min heap

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)
```

binary min-heap

BUILD-MIN-HEAP $O(V)$

$|V|$ times

EXTRACT-MIN $O(\lg V)$ $O(V \lg V)$

$O(E)$

DECREASE-KEY $O(\lg V)$ $O(E \lg V)$

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