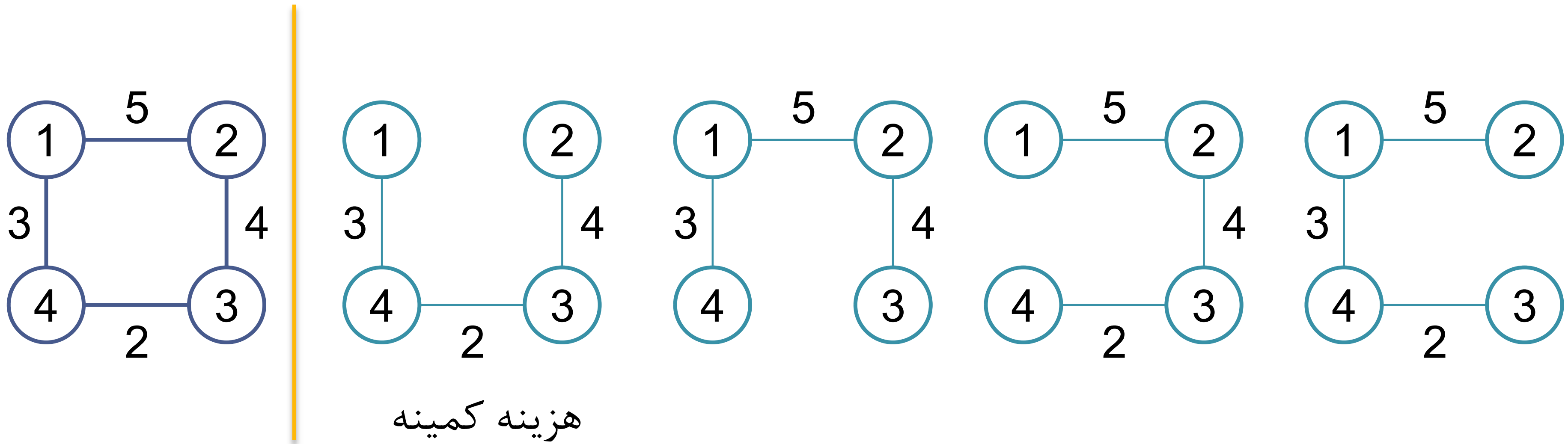
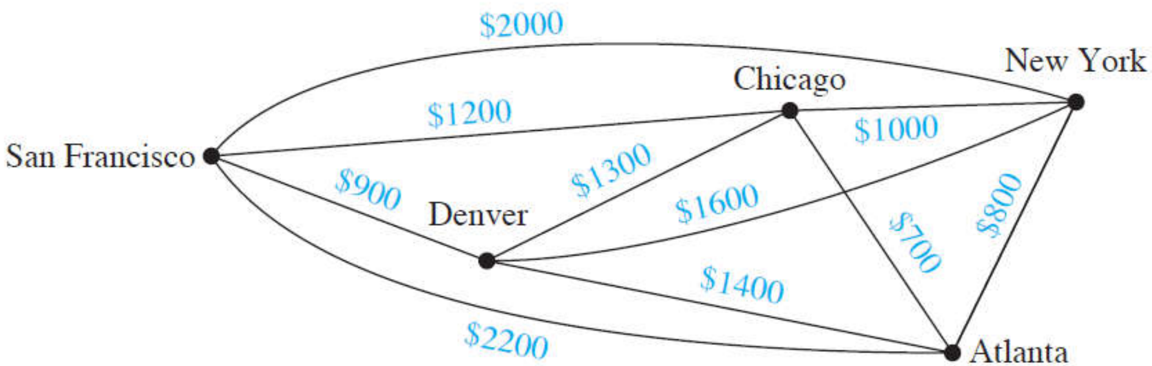
یادآوری

- **درخت پوشا (فراگیر):**
- اگر G یک گراف ساده باشد، درخت پوشای G، زیرگرافی از G است که درخت می‌باشد و همه‌ی رئوس گراف را نیز در بر دارد.
- گراف زیر را در نظر بگیرید. تمامی درخت‌های زیر، درخت پوشای گراف مربوطه می‌باشند.



کاربرد درخت فراگیر کمینه

- یک مثال:
- شرکتی می‌خواهد یک شبکه کامپیوتری بین ۵ مرکز کامپیوتری خود برقرار کند، هر جفت این مرکزها با خط تلفنی خاص به هم وصل هستند که برای کرایه این خطوط تلفن، هزینه‌ای مشخص وجود دارد. کدام اتصالات باید وجود داشته باشد به طوری که بین هر دو مرکز کامپیوتری مسیری وجود داشته باشد و هزینه کلی برقراری شبکه نیز مینیمم گردد؟



درخت فراگیر کمینه (Minimum Spanning Trees)

- نام دیگر: درخت پوشای مینیمم
- درخت فراگیر کمینه‌ی یک گراف وزن‌دار همبند، درختی پوشا است که مجموع وزن‌های آن مینیمم می‌باشد.

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

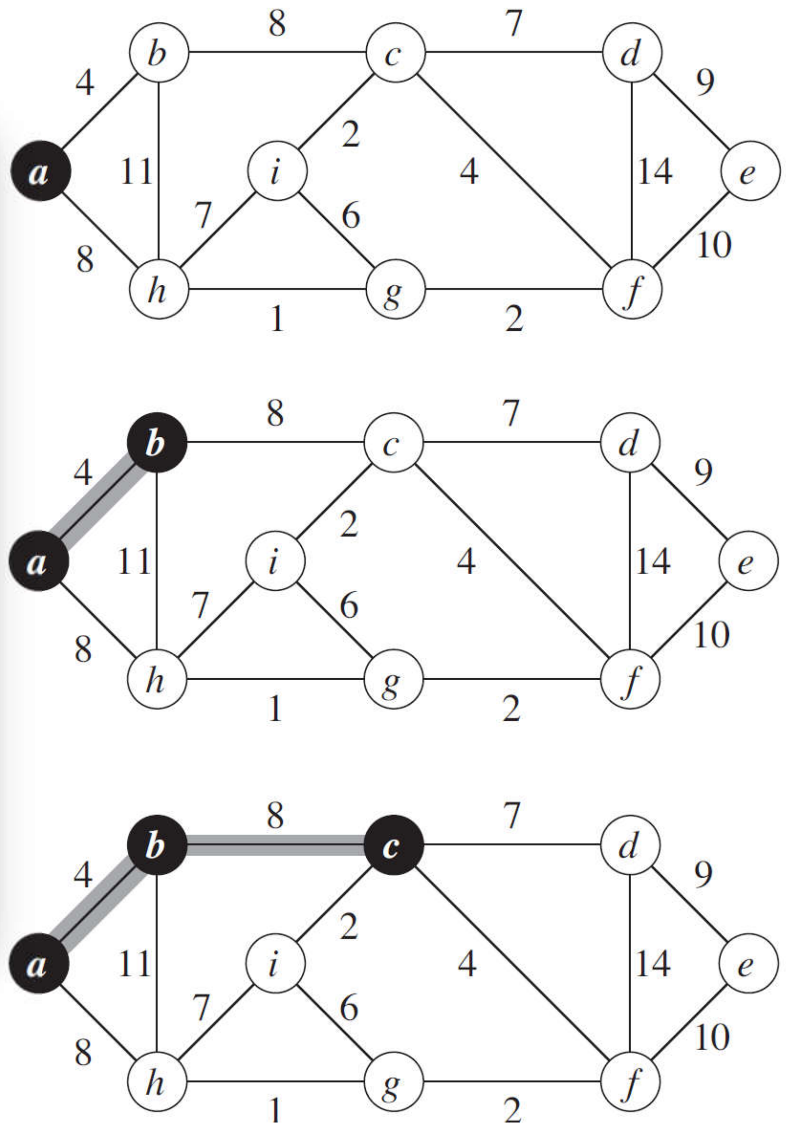
- الگوریتم‌های موجود برای بدست آوردن درخت پوشای مینیمم:
  1. الگوریتم پریم (Prim)
  2. الگوریتم کروسکال (Kruskul)
  3. الگوریتم سولین (Sollin)

الگوریتم پریم

- این الگوریتم در ابتدا توسط ریاضیدان چک، Jarník، در سال ۱۹۳۰ مطرح شد. الگوریتم مربوطه زمانی که توسط رابرت پریم در سال ۱۹۵۷ دوباره کشف شد، مورد استقبال قرار گرفت.
- الگوریتم کلی:
  - یک نود به عنوان نود شروع در نظر بگیر.
  - از بین تمامی یال‌ها متصل به نود شروع، یال با وزن کمتر را به درخت اضافه کن.
  - حال از بین یال‌هایی که به تمامی رئوس فعلی در درخت وصل هستند، یال با وزن کمتر را انتخاب کن به طوری که دور در درخت ایجاد نشود.
  - آن یال را به درخت اضافه کن.
  - همین فرآیند را ادامه بده تا n-1 یال دیده شود.

الگوریتم پریم (ادامه)

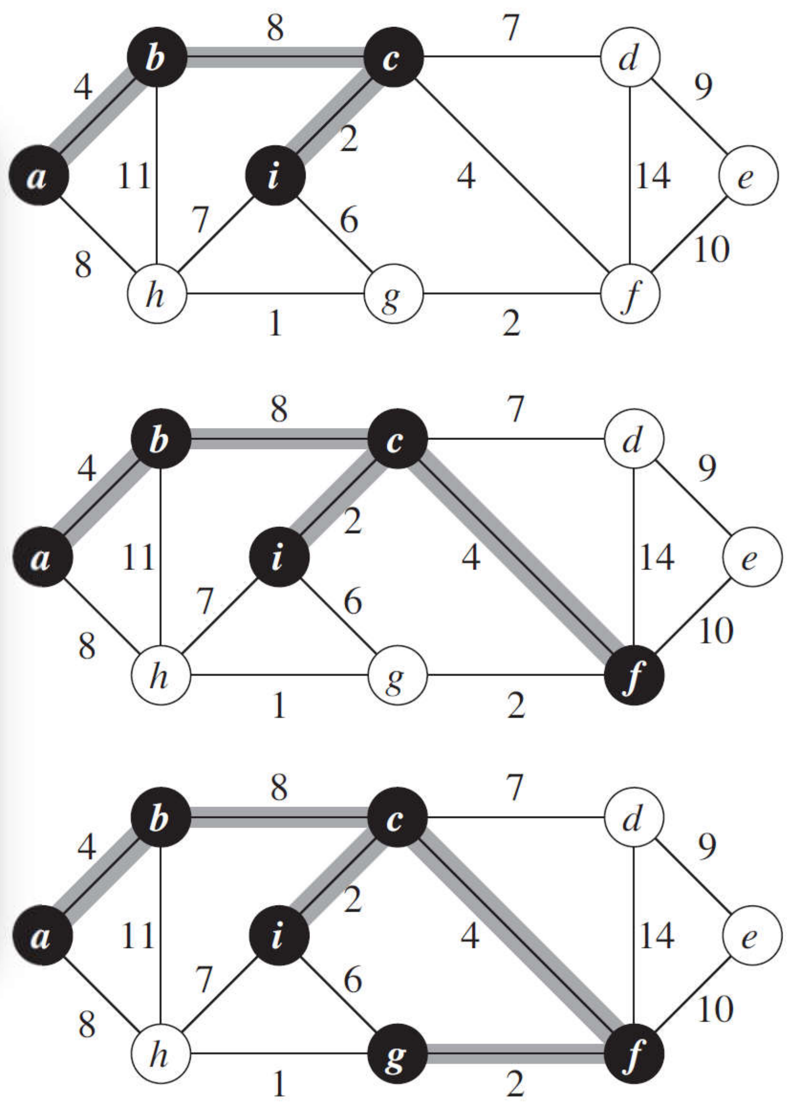
```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```



شکل ۲۳-۵ کتاب CLRS

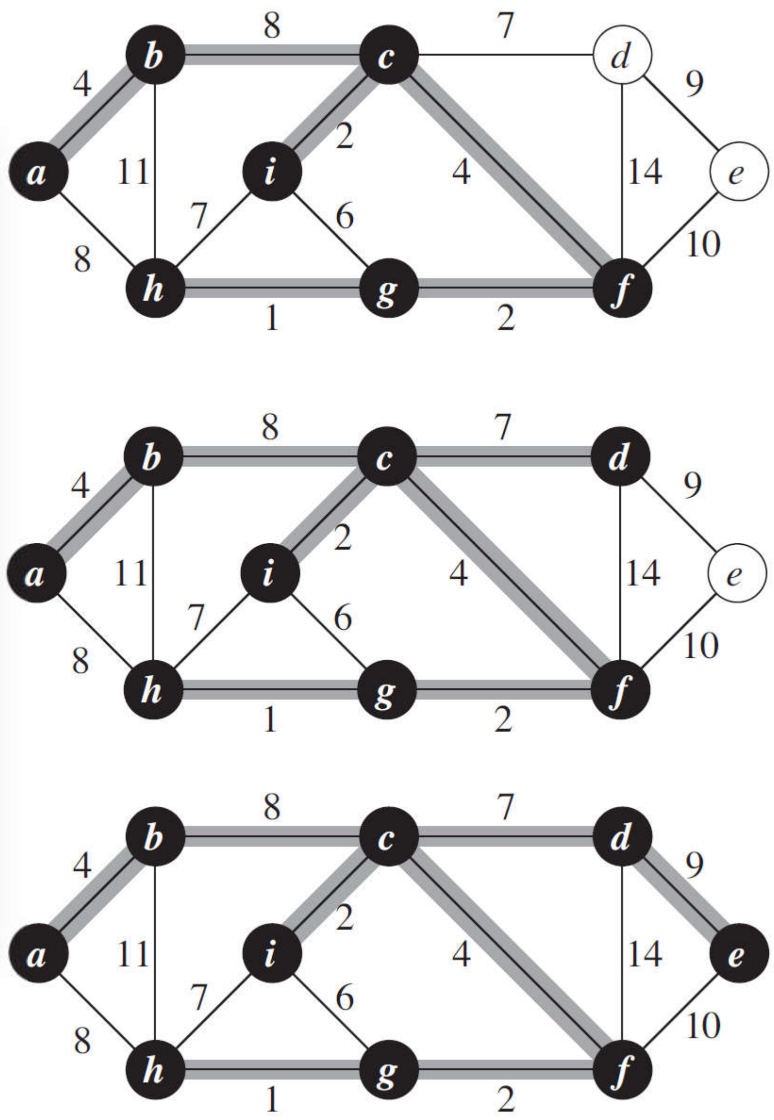
الگوریتم پریم (ادامه)

```
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)
```



الگوریتم پریم (ادامه)

```
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)
```



تحلیل الگوریتم پریم

- نوع صف؟

```
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)
```

- زمان EXTRACT-MIN(Q)؟

- زمانی که مقدار کلید کاهش پیدا می‌کند چه اتفاقی در صف می‌افتد؟

تحلیل الگوریتم پریم

- نوع صف؟

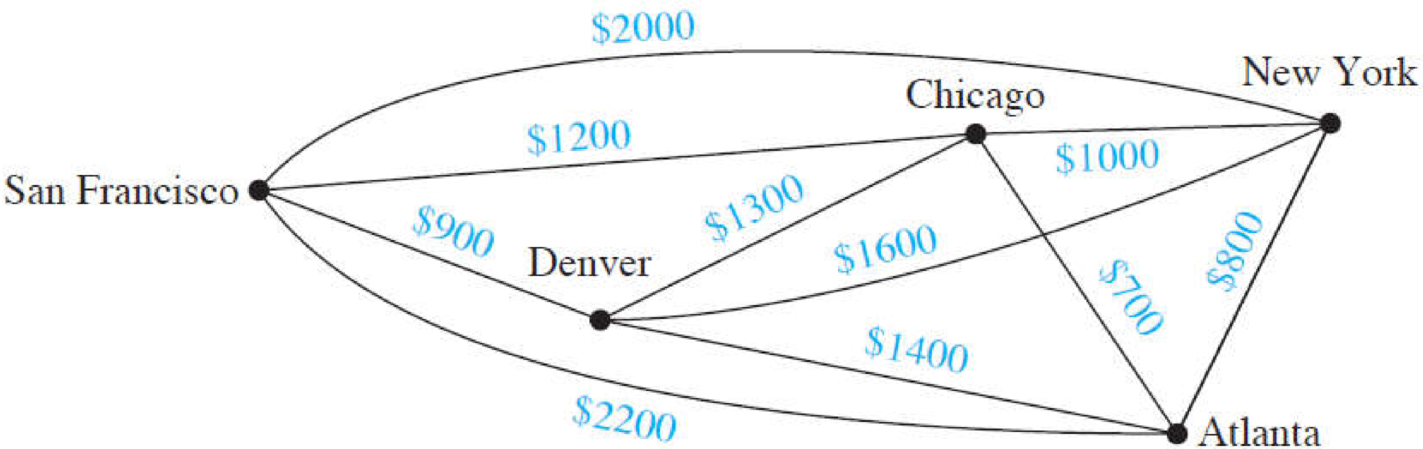
```
MST-PRIM(G, w, r)
for each u ∈ G.V
    u.key = ∞
    u.π = NIL
r.key = 0
Q = G.V
while Q ≠ Ø
    u = EXTRACT-MIN(Q)
    for each v ∈ G.Adj[u]
        if v ∈ Q and w(u, v) < v.key
            v.π = u
            v.key = w(u, v)
```

- زمان EXTRACT-MIN(Q)؟
- زمانی که مقدار کلید کاهش پیدا می‌کند چه اتفاقی در صف می‌افتد؟
- نوع الگوریتم؟

$$T(n) = |V| \times T(\text{Extract-Min}) + \Theta(E) \times T(\text{Decrease-Key})$$

تمرین ۱

- درخت پوشای مینیمم گراف زیر را با استفاده از الگوریتم پریم بدست آوردید. (یال‌ها را به ترتیب انتخاب لیست کنید).
- نود شروع: سان فرانسیسکو



الگوریتم کروسکال

- ارائه شده توسط ژوزف کروسکال در سال ۱۹۵۶.
- الگوریتم کلی:
  - یال‌ها را به ترتیب صعودی وزنشان مرتب کن.
  - سپس به ترتیب یال‌ها را بررسی کن.
  - اگر دور تشکیل نشود، آن یال را انتخاب در غیر اینصورت از آن یال صرفنظر کن.
  - این کار را تا زمانی ادامه بده تا n-1 یال در درخت پوشا بدست آید.

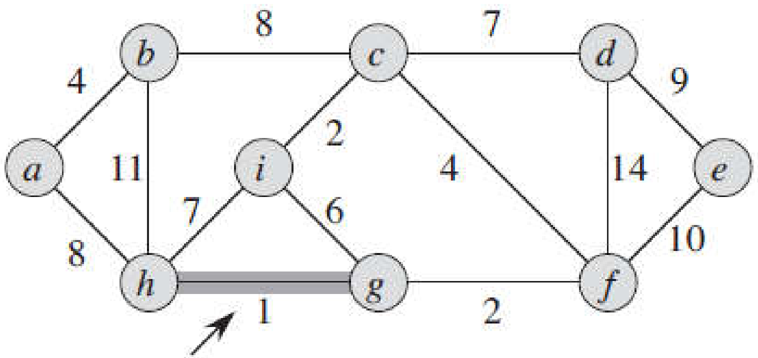
الگوریتم کروسکال (ادامه)

```
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
```

الگوریتم کروسکال (ادامه)

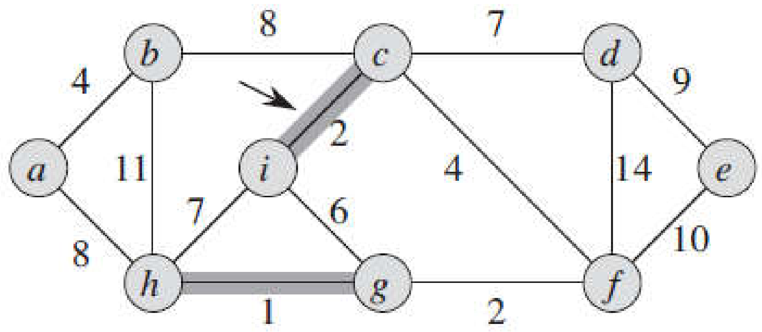
$$\{a\}\{b\}\{c\}\{d\}\{e\}\{f\}\{g\}\{h\}\{i\}$$

(a)



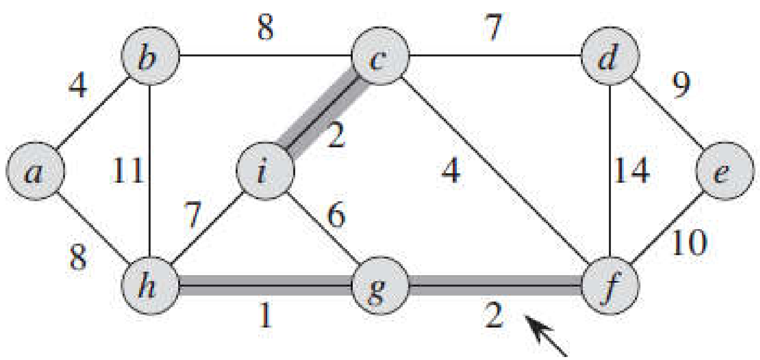
$$\{a\}\{b\}\{c\}\{d\}\{e\}\{f\}\{g,h\}\{i\}$$

(b)



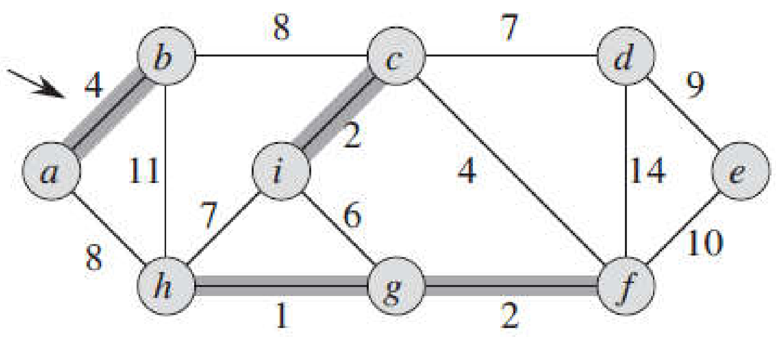
$$\{a\}\{b\}\{c,i\}\{d\}\{e\}\{f\}\{g,h\}$$

(c)



$$\{a\}\{b\}\{c,i\}\{d\}\{e\}\{f,g,h\}$$

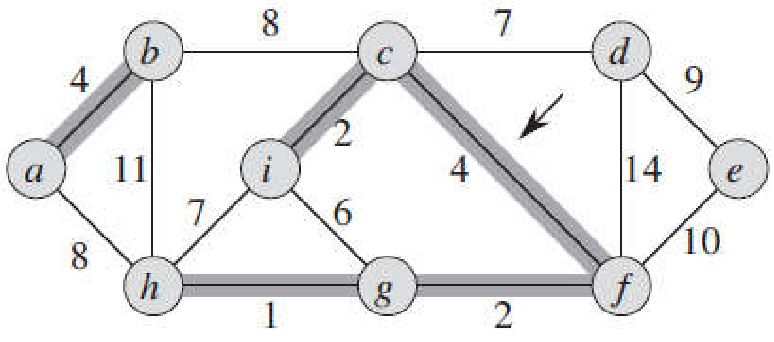
(d)



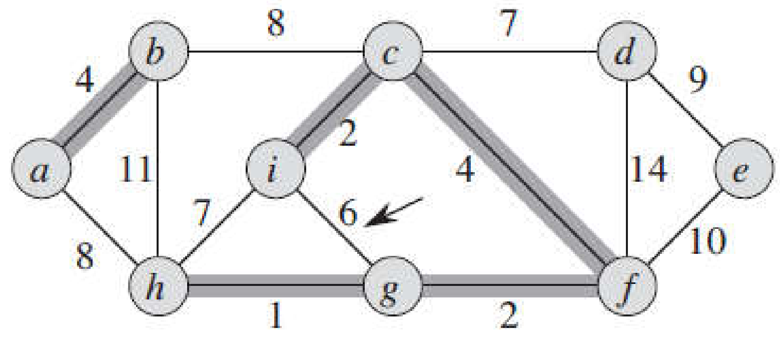
$$\{a,b\}\{c,i\}\{d\}\{e\}\{f,g,h\}$$

شکل ۲۳-۴ کتاب CLRS

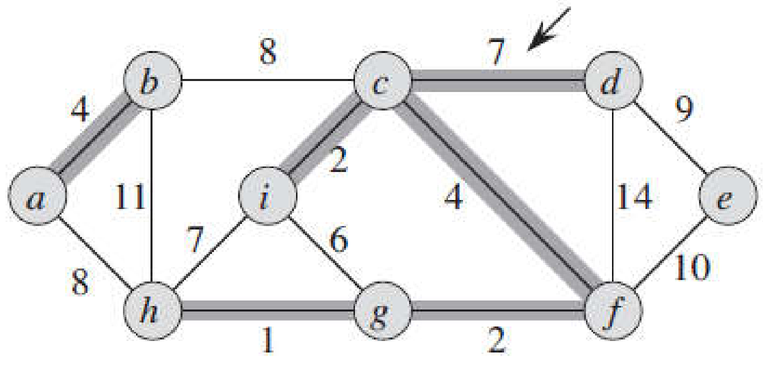
الگوریتم کروسکال (ادامه)

(e) 

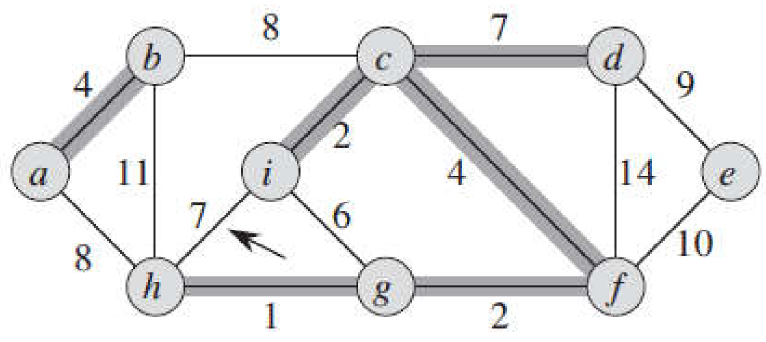
$\{a,b\}\{c,i,f,g,h\}\{d\}\{e\}$

(f) 

$\{a,b\}\{c,i,f,g,h\}\{d\}\{e\}$

(g) 

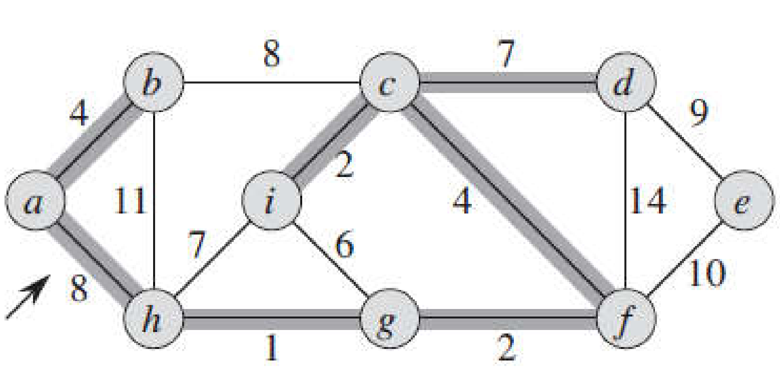
$\{a,b\}\{c,d,i,f,g,h\}\{e\}$

(h) 

$\{a,b\}\{c,d,i,f,g,h\}\{e\}$

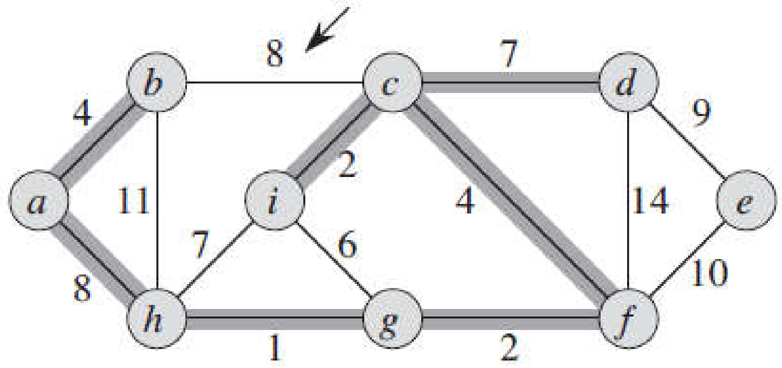
الگوریتم کروسکال (ادامه)

(i)



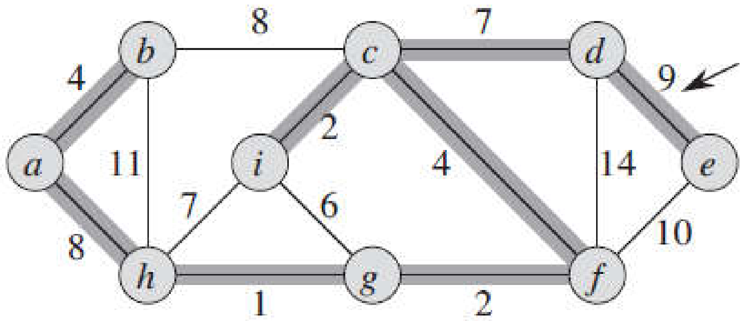
$\{a,b,c,d,i,f,g,h\}\{e\}$

(j)



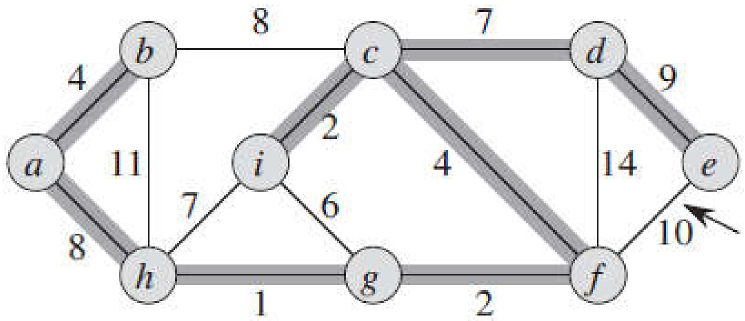
$\{a,b,c,d,i,f,g,h\}\{e\}$

(k)



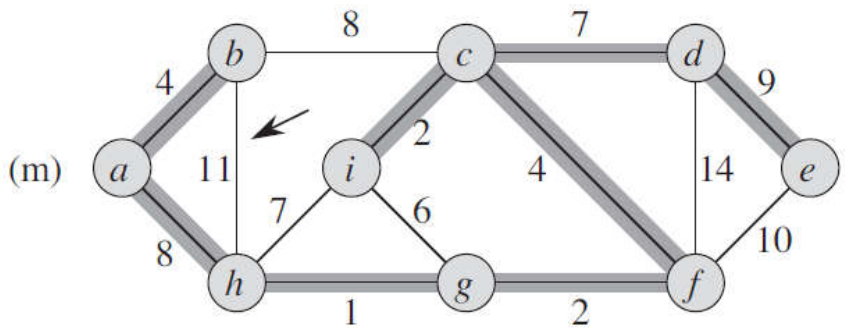
$\{a,b,c,d,e,i,f,g,h\}$

(l)

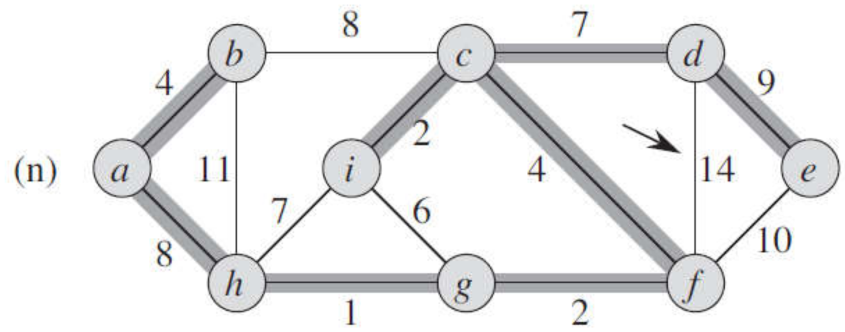


$\{a,b,c,d,e,i,f,g,h\}$

الگوریتم کروسکال (ادامه)



$\{a,b,c,d,e,i,f,g,h\}$



$\{a,b,c,d,e,i,f,g,h\}$

تحلیل الگوریتم کروسکال

```
MST-KRUSKAL(G, w)
A = ∅
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
```

- تعداد فراخوانی‌های MAKE-SET(v)؟
- مرتب‌سازی یال‌ها؟
- تعداد فراخوانی‌های FIND-SET(u)؟
- تعداد فراخوانی‌های UNION(u, v)؟

تحلیل الگوریتم کروسکال (ادامه)

```
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
```

- m عمل بر روی n مجموعه در زمان $\Theta\left(m \cdot \alpha(m,n)\right)$ انجام می‌شود، که در آن α عکس تابع اکرمن است که بسیار کند رشد می‌کند.
- بنابراین داریم: $\Theta\left(E \times \alpha(E,V)\right)$
- مرتبه کل: $\Theta\left(V + E\log V + E \times \alpha(E,V)\right) = \Theta\left(E\log V + E \times \alpha(E,V)\right)$

یادآوری تابع اکرمن (Ackermann Function)

- **تابع بازگشتی آکرمن:**

$$A(i,j)=\begin{cases}2^j & i=1 \text{ and } j\ge 1\\ A(i-1,2) & i\ge 2 \text{ and } j=1\\ A(i-1,A(i,j-1)) & i\ge 2 \text{ and } j\ge 2\end{cases}$$

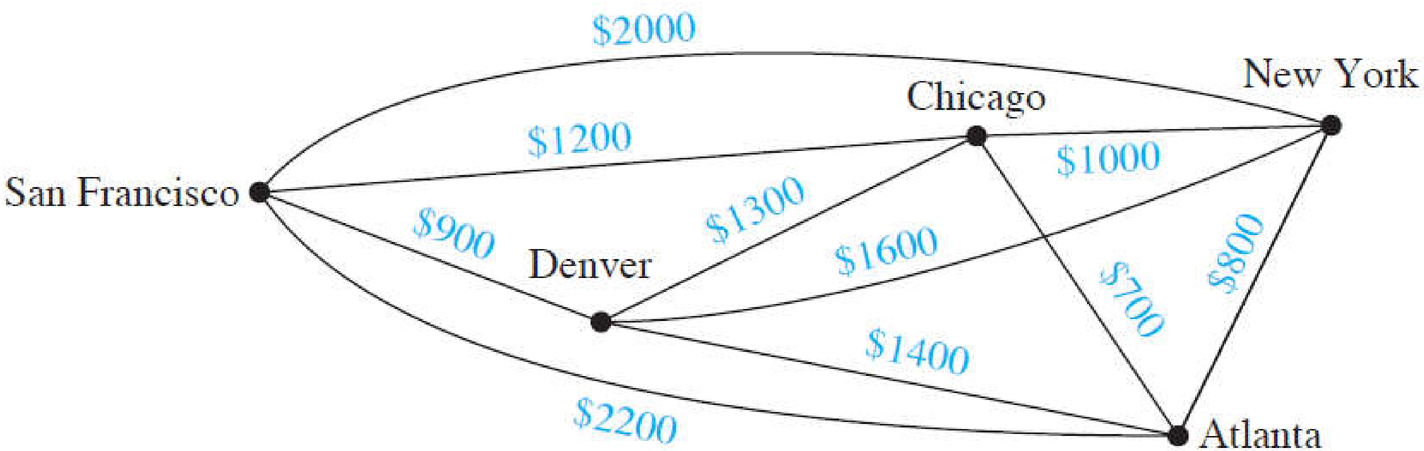
- این تابع رشد بسیار زیادی دارد به طوری که برای $A(4,2)$ عددی ۱۹۷۲۹ رقمی را پاسخ می‌دهد.
- **معکوس تابع آکرمن:** $\alpha(m,n)=\min\{i\ge 1: A(i,\lfloor m/n\rfloor)>\log n)\}$
- مقدار $\lfloor m/n\rfloor$ دست‌کم ۱ است. به ازای $\lfloor m/n\rfloor\ge 1$ داریم: $A(i,\lfloor m/n\rfloor)\ge A(i,1)$
- بنابراین:

$$A(4,\lfloor m/n\rfloor)\ge A(4,1)=2^{2^{\cdot^{\cdot^{2}}}}\Big\}16$$

- که بسیار بیشتر از تخمین تعداد کل اتم‌ها در جهان (حدود $۱۰^{۸۰}$ است). فقط برای مقادیر غیر عملی بزرگ n، ممکن است $A(4,1)\le\log n$
- بنابراین برای همه‌ی مقادیر عملی: $\alpha(m,n)\le 4$

تمرین ۲

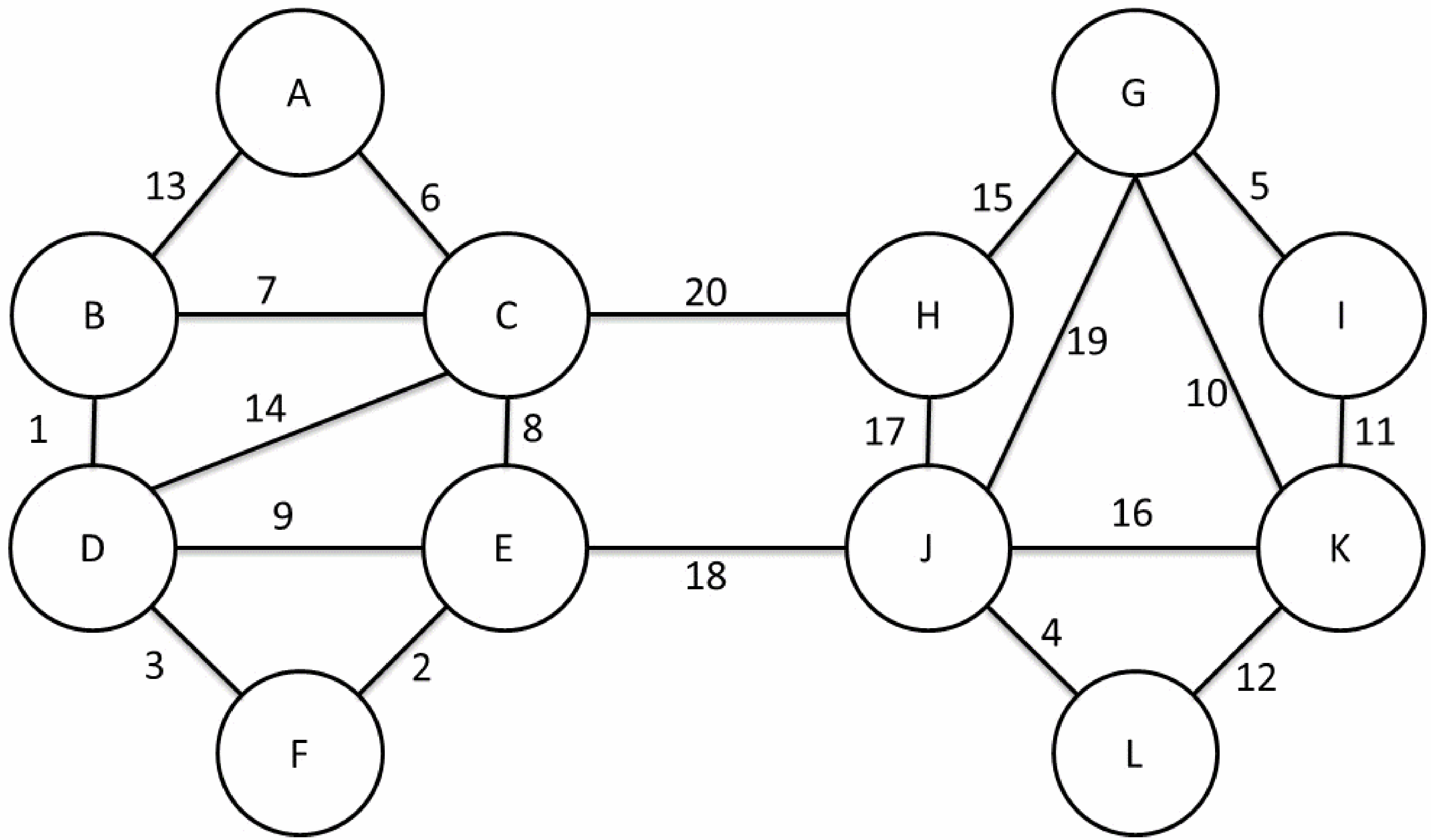
- درخت پوشای مینیمم گراف زیر را با استفاده از الگوریتم کروسکال بدست آوردید. (به ترتیب یال‌هایی که قبول و رد می‌شوند را لیست کنید).



الگوریتم سولین

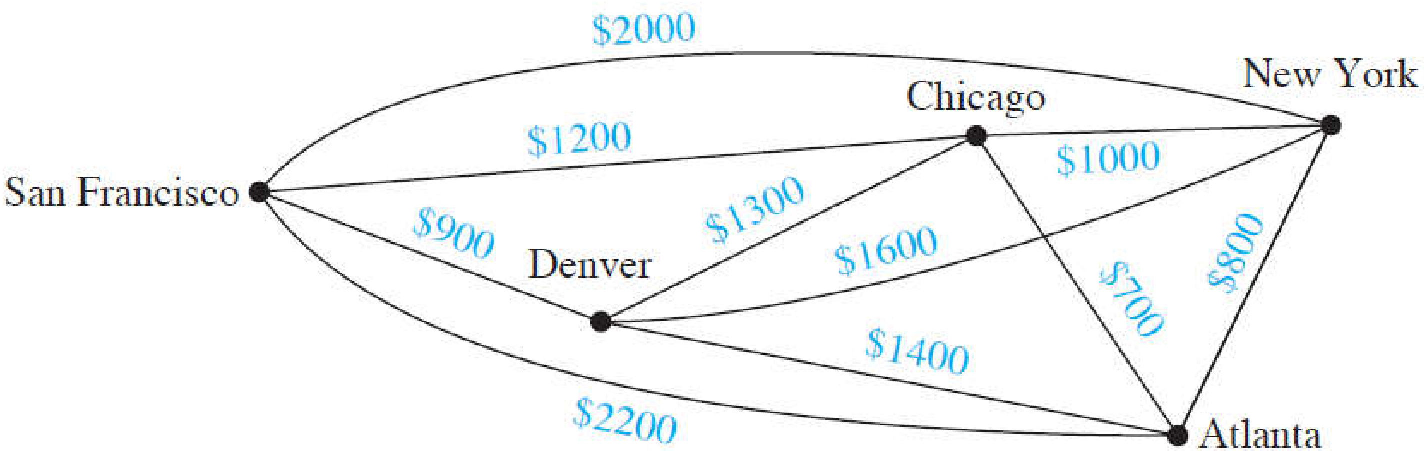
- نام دیگر: الگوریتم بروفکا (Borůvka).
- ارائه شده توسط اوتاکار بروفکا در سال ۱۹۲۶.
- الگوریتم کلی:
  - ابتدا با تک تک نودها شروع کرده (جنگلی از نودها) سپس از یال‌های متصل به هر نود، یال با کمترین هزینه را انتخاب کن.
  - از اجزای به دست آمده (درخت‌ها در جنگل) دوباره یال با کمترین هزینه را انتخاب کن.
  - این کار را ادامه داده تا زمانی که فقط یک جزء باقی بماند.
- برای گراف‌هایی استفاده می‌شود که یال‌های آن دارای وزن متفاوتی است.

مثال الگوریتم سولین



تمرین ۳

- الگوریتم سولین را اعمال کنید و به ترتیب اجزاء را در هر مرحله مشخص سازید.

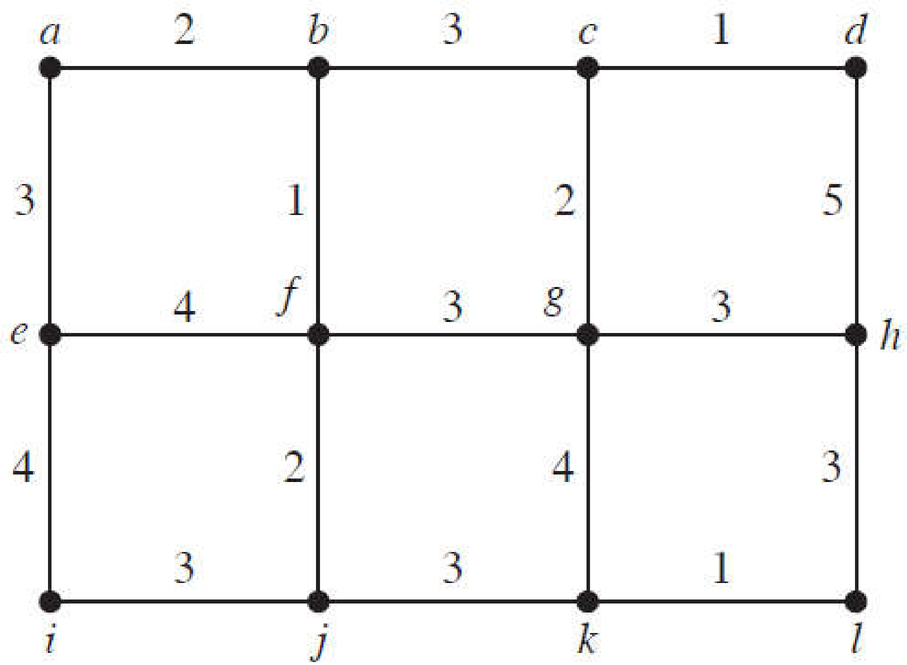


تحلیل الگوریتم سولین

- بدترین حالت برای الگوریتم سولین زمانی خواهد بود که در هر مرحله اجزاء نصف گردند.
- در هر مرحله نیز از یال‌های باقیمانده متصل کمترین باید انتخاب گردد.
- مرتبه؟

تمرین بیشتر

1. با استفاده از الگوریتم پریم، درخت پوشای مینیمم گراف زیر را بدست آورید.
2. الگوریتم کروسکال را بر روی گراف زیر اعمال کنید.
3. الگوریتم سولین را اعمال کنید. چه نتیجه‌ای می‌گیرید؟



The END