جستجوی گراف‌ها

- دلایل استفاده از جستجو در گراف‌ها
  - به دنبال یک نود (رأس) یا مسیر با ویژگی خاص در گراف هستیم.
  - می‌خواهیم گراف را پیمایش کنیم یعنی بررسی یال‌ها و رأس‌های گراف.
- مثال:
  - جستجوی سطح اول (پیمایش سطحی – breadth first search)
  - جستجوی عمق اول (پیمایش عمقی – depth first search)
- فرض:
  - از لیست مجاورت استفاده می‌کنیم.

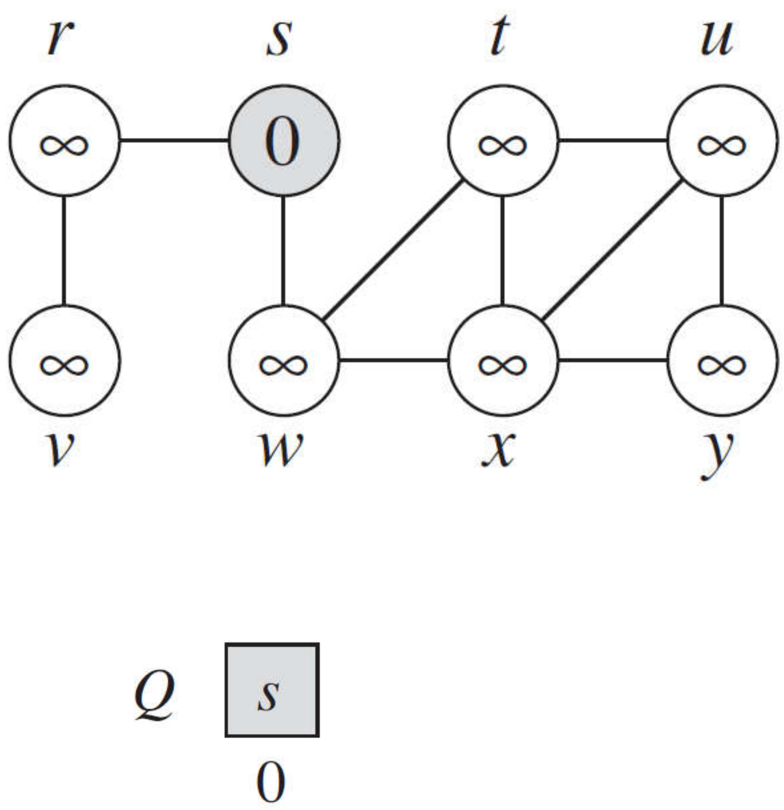
جستجوی سطح اول (BFS)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- جستجوی لایه به لایه.
- همه‌ی رأس‌ها با فاصله‌ی k از s (نود شروع) را زودتر از رأس‌ها با فاصله‌ی $k+1$ را کشف می‌کند.
- رنگ کردن هر سطح با سه رنگ سفید، خاکستری و سیاه.

مثال از جستجوی سطح اول

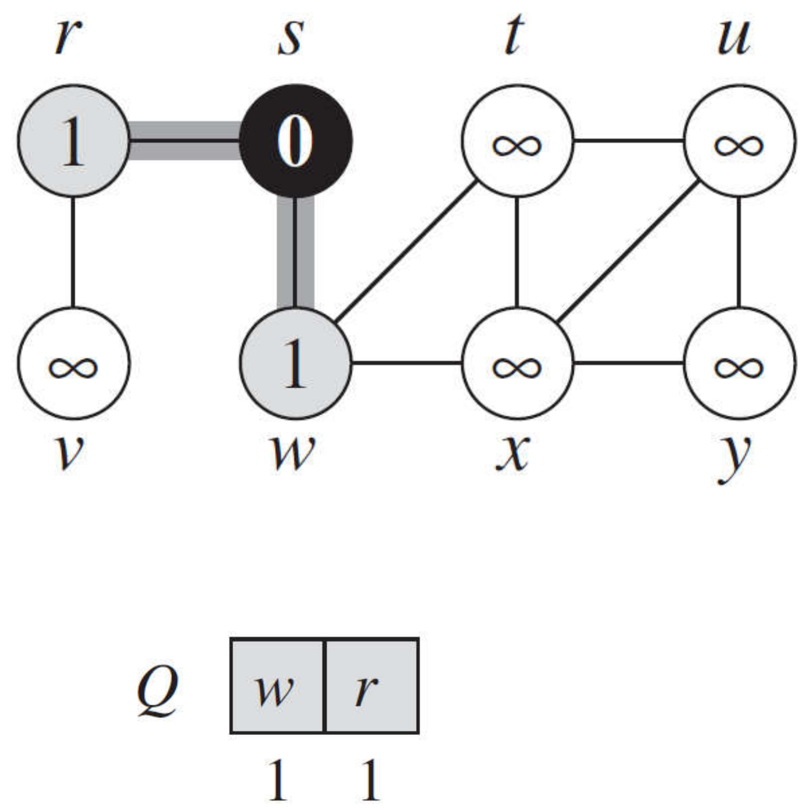
```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



شکل ۲۲-۳ کتاب CLRS

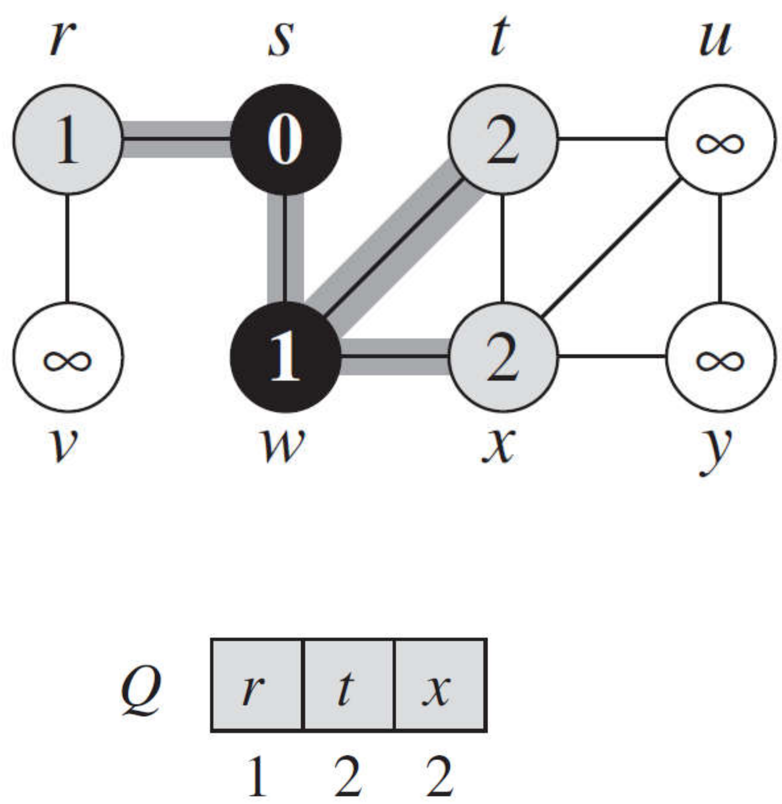
مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



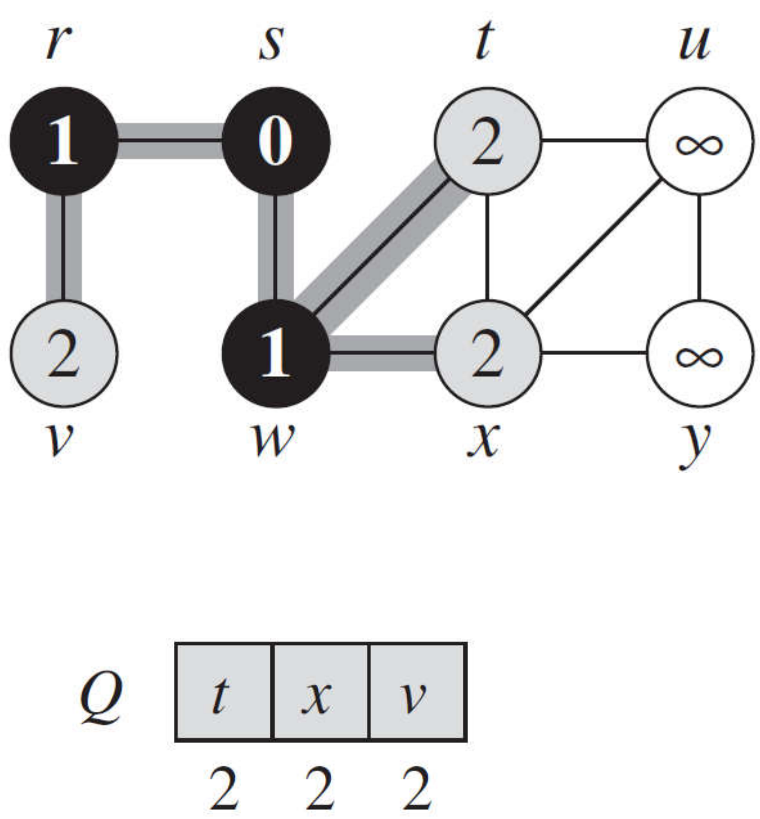
مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



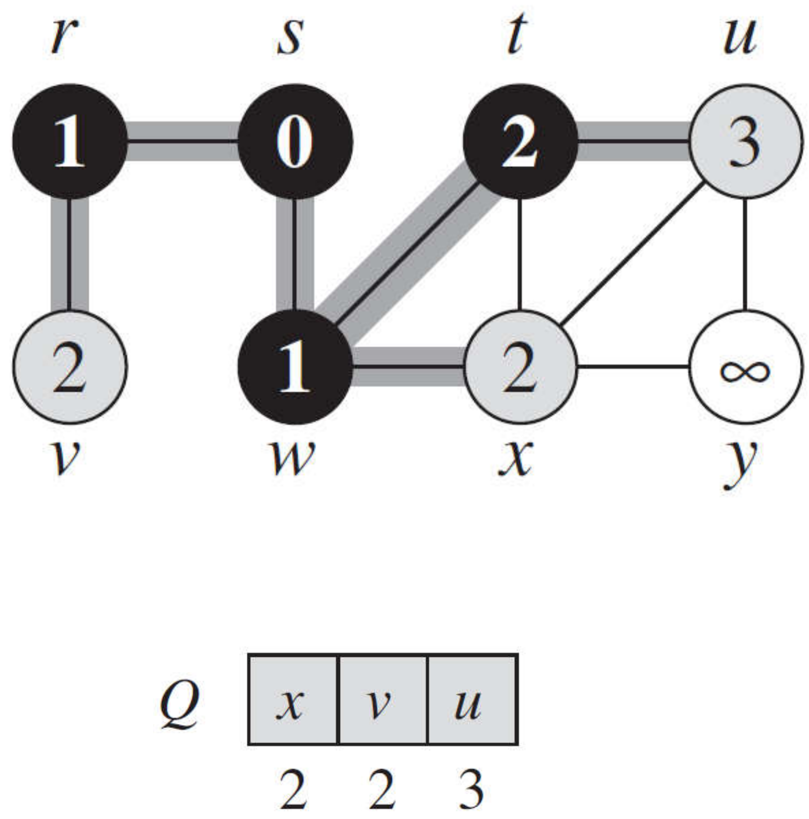
مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



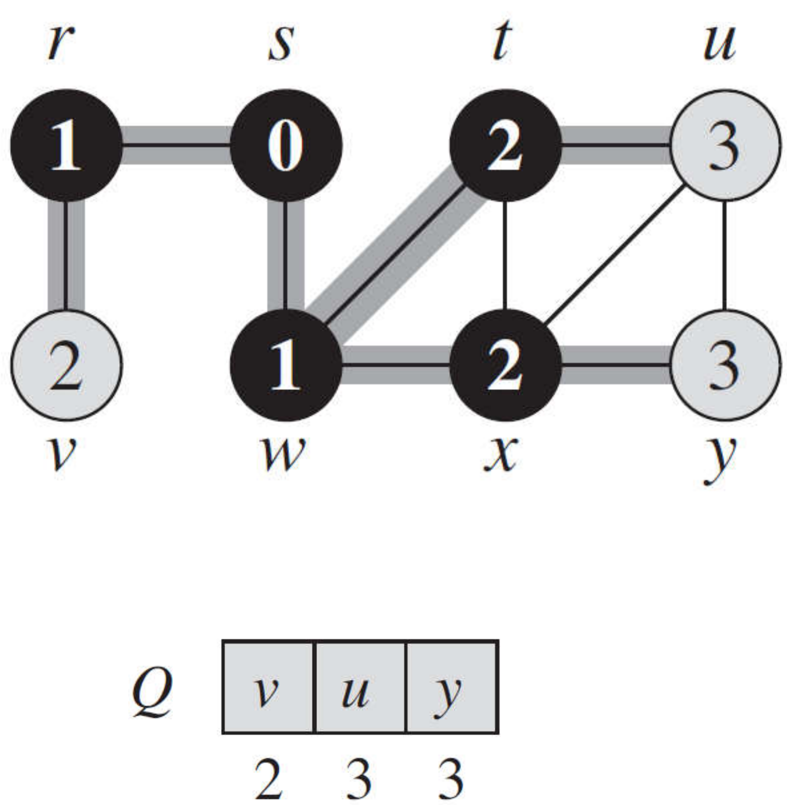
مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



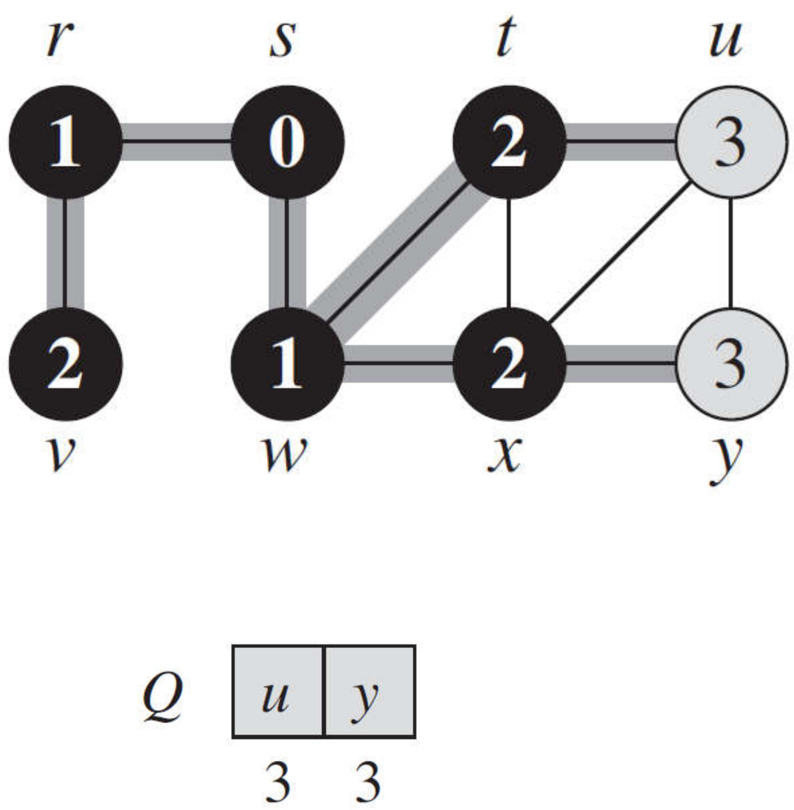
مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



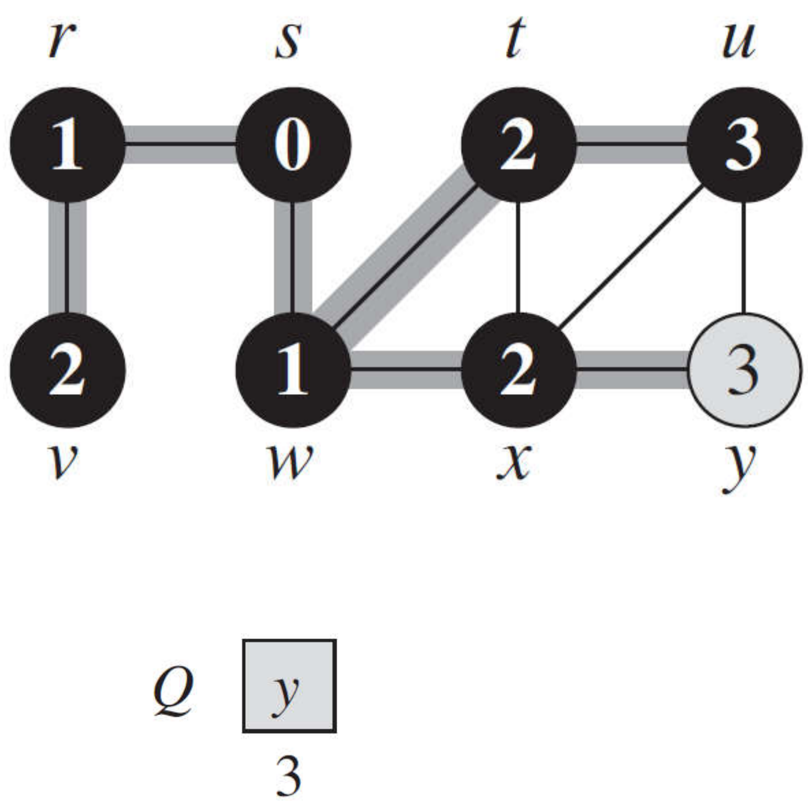
مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



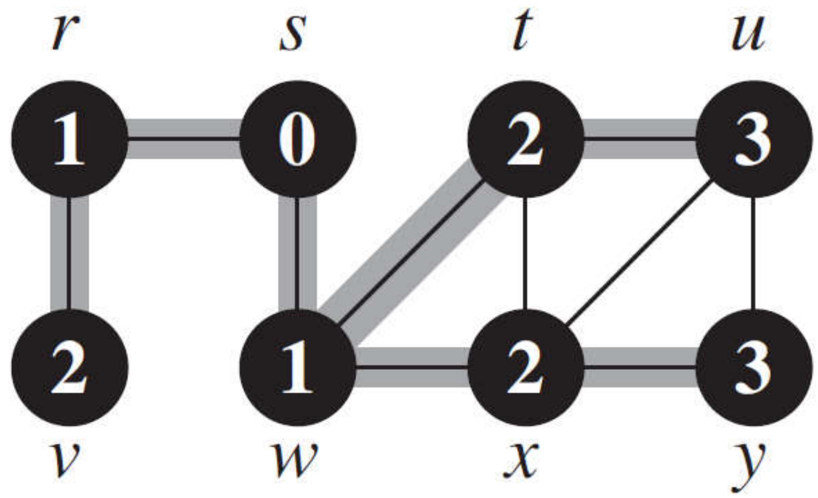
مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



مثال از جستجوی سطح اول (ادامه)

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```



$Q \quad \emptyset$

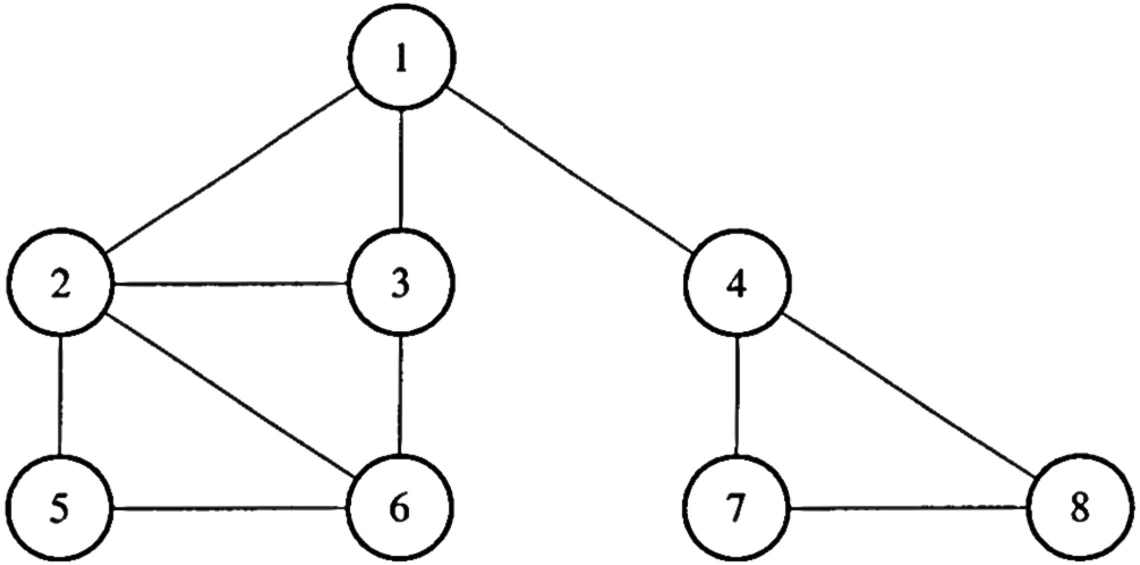
تحلیل جستجوی سطح اول

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- پس از جستجو، اگر V به عنوان نود کشف شده تعیین گردد، نشان دادیم که بین نود S و نود V مسیری وجود دارد.
- تعداد درج‌ها به صف؟
- تعداد بررسی‌های نودهای مجاور؟
- کل زمان اجرای BFS؟
- آیا برای گراف جهتدار الگوریتم برقرار است؟

تمرین ۱

- ترتیب مشاهده گره‌ها با جستجوی سطح اول بر گراف زیر به چه صورت می‌باشد. نود شروع، نود با شماره ۱ می‌باشد.



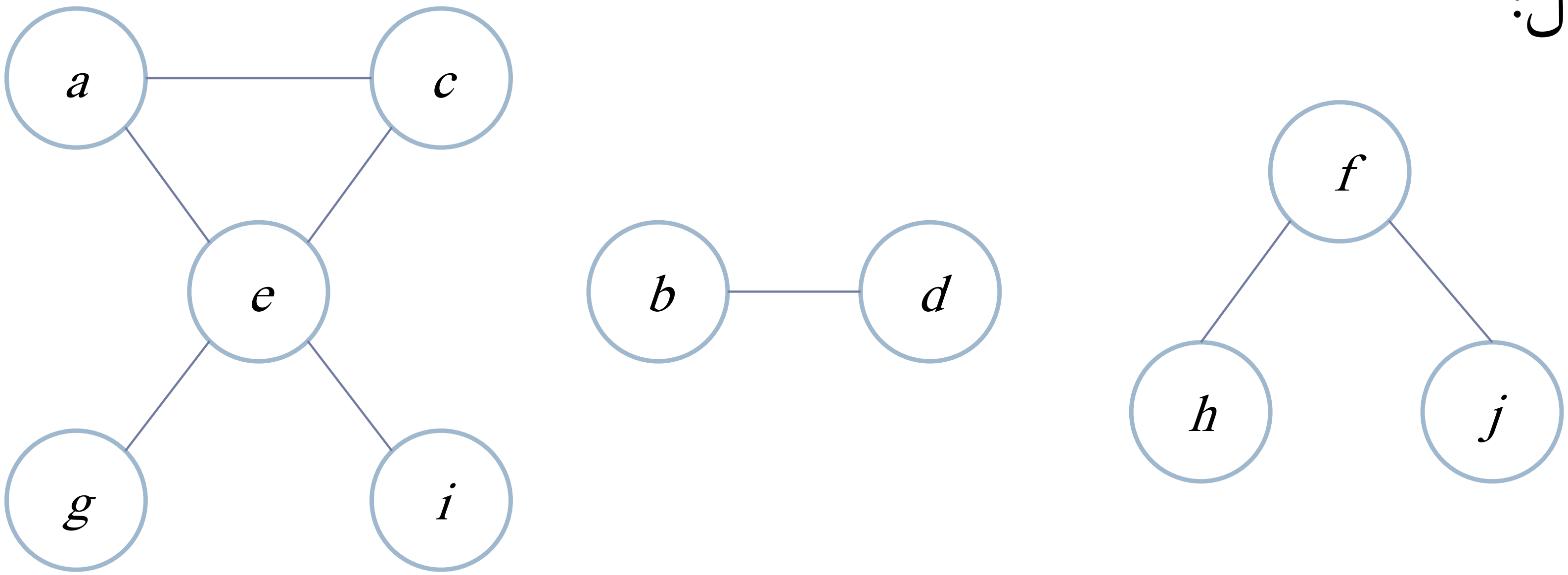
کاربرد جستجوی سطح اول

1. کوتاهترین مسیرها: جستجوی سطح اول کوتاهترین مسیرها را از نظر تعداد یال (یا زمانی که وزن همه‌ی یال‌ها برابر باشد) بدست می‌آورد.
   - مکان‌یابی GPS.
   - شبکه‌های کامپیوتری و سیستم‌های p2p.

2. بدست آوردن تمامی مؤلفه‌های همبند در گراف غیرجهتدار:
- ابتدا نودی که کشف نشده است را انتخاب کرده و جستجو سطح اول را اجرا می‌کنیم ← نتیجه: بدست آوردن مؤلفه اول
- نودی دیگر که کشف نشده است را انتخاب کرده و دوباره جستجو را اجرا می‌کنیم ← نتیجه: بدست آوردن مؤلفه دوم
- تکرار الگوریتم تا تمامی مؤلفه‌ها بدست آید.

بدست آوردن مؤلفه‌های همبند با جستجوی سطح اول

- مثال:



- مرتبه زمانی؟

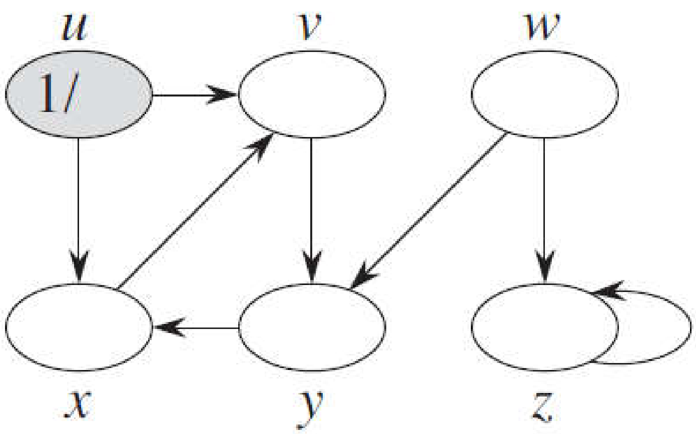
جستجوی عمق اول (DFS)

- تا زمانی که امکان دارد در عمق پیش برو، اگر امکانش وجود نداشت، برگرد و شاخه‌ای دیگر را در پیش بگیر.
- استفاده از یک متغیر سراسری *time* که زمان شروع و پایان ملاقات نود را تعیین می‌کند.

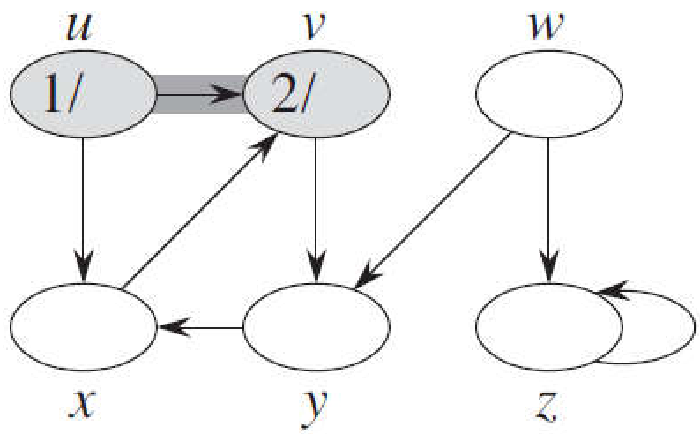
```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

مثال از جستجوی عمق اول



(a)



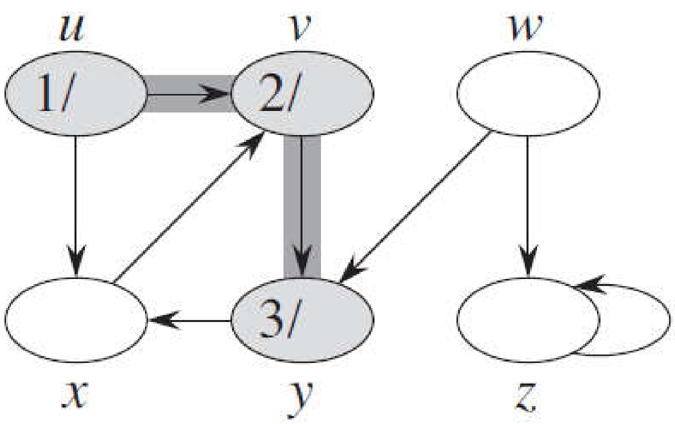
(b)

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

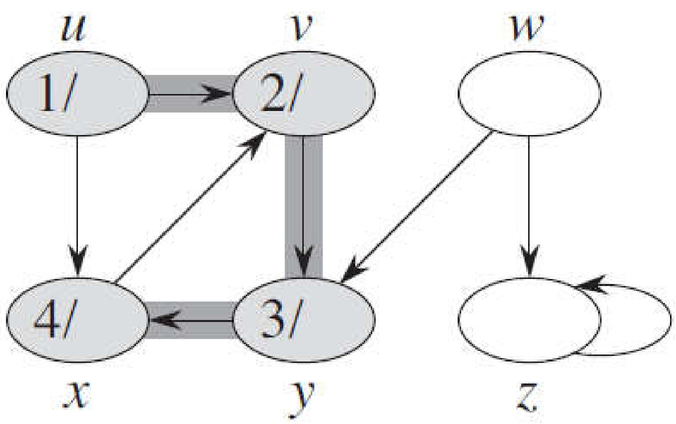
```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

شکل ۲۲-۴ کتاب CLRS

مثال از جستجوی عمق اول



(c)



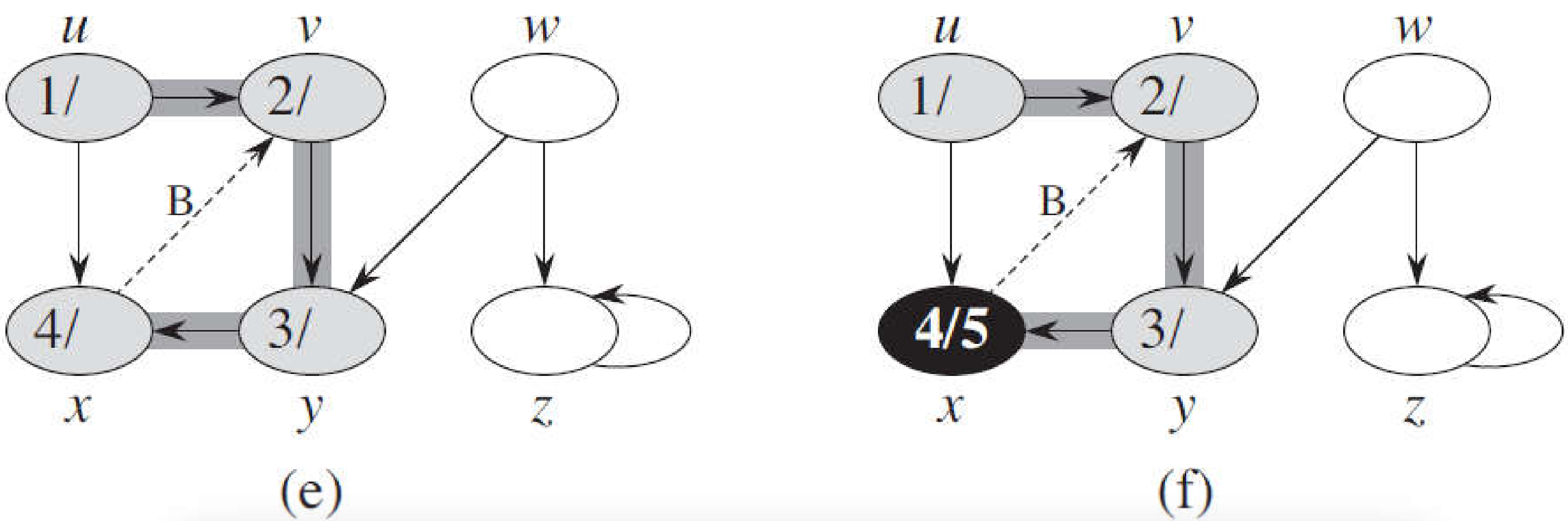
(d)

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

شکل ۲۲-۴ کتاب CLRS

مثال از جستجوی عمق اول (ادامه)

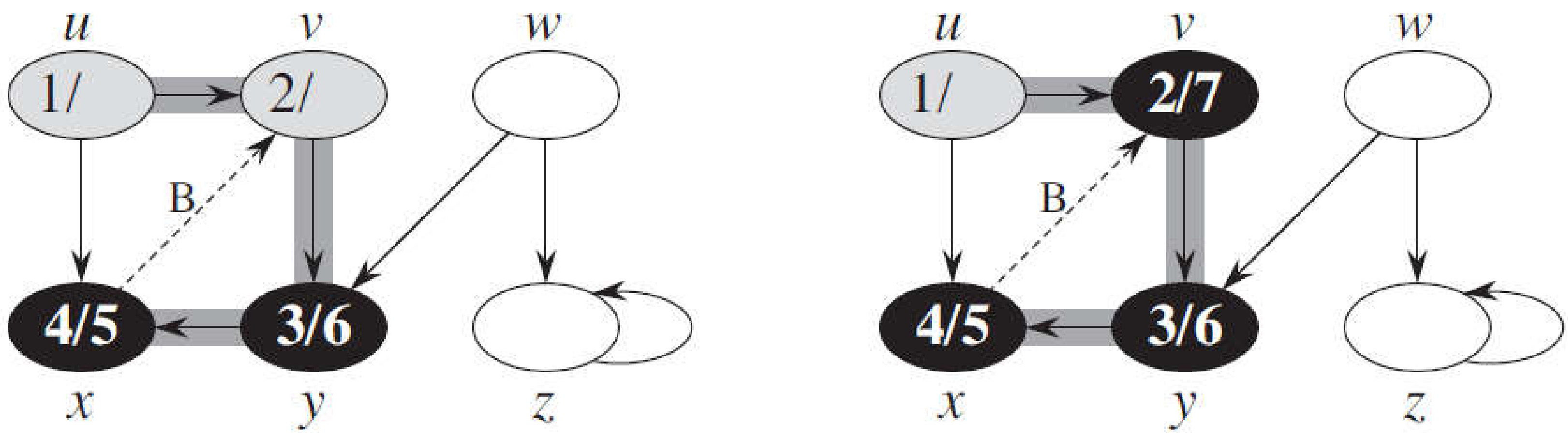



```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

شکل ۲۲-۴ کتاب CLRS

مثال از جستجوی عمق اول (ادامه)



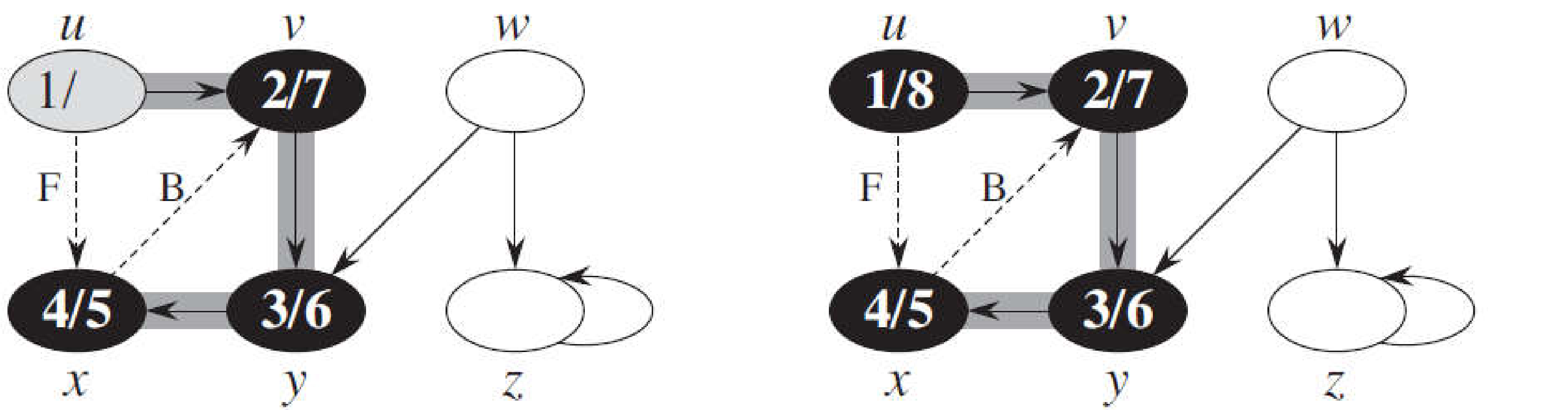
(g) (h)

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

شکل ۲۲-۴ کتاب CLRS

مثال از جستجوی عمق اول (ادامه)



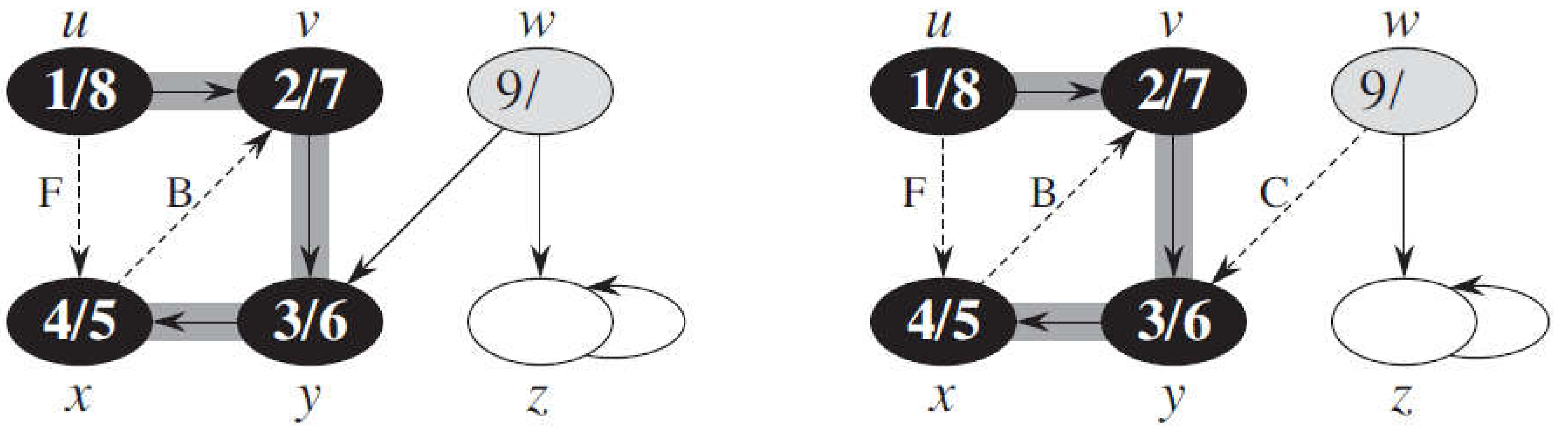
(i) (j)

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

شکل ۲۲-۴ کتاب CLRS

مثال از جستجوی عمق اول (ادامه)

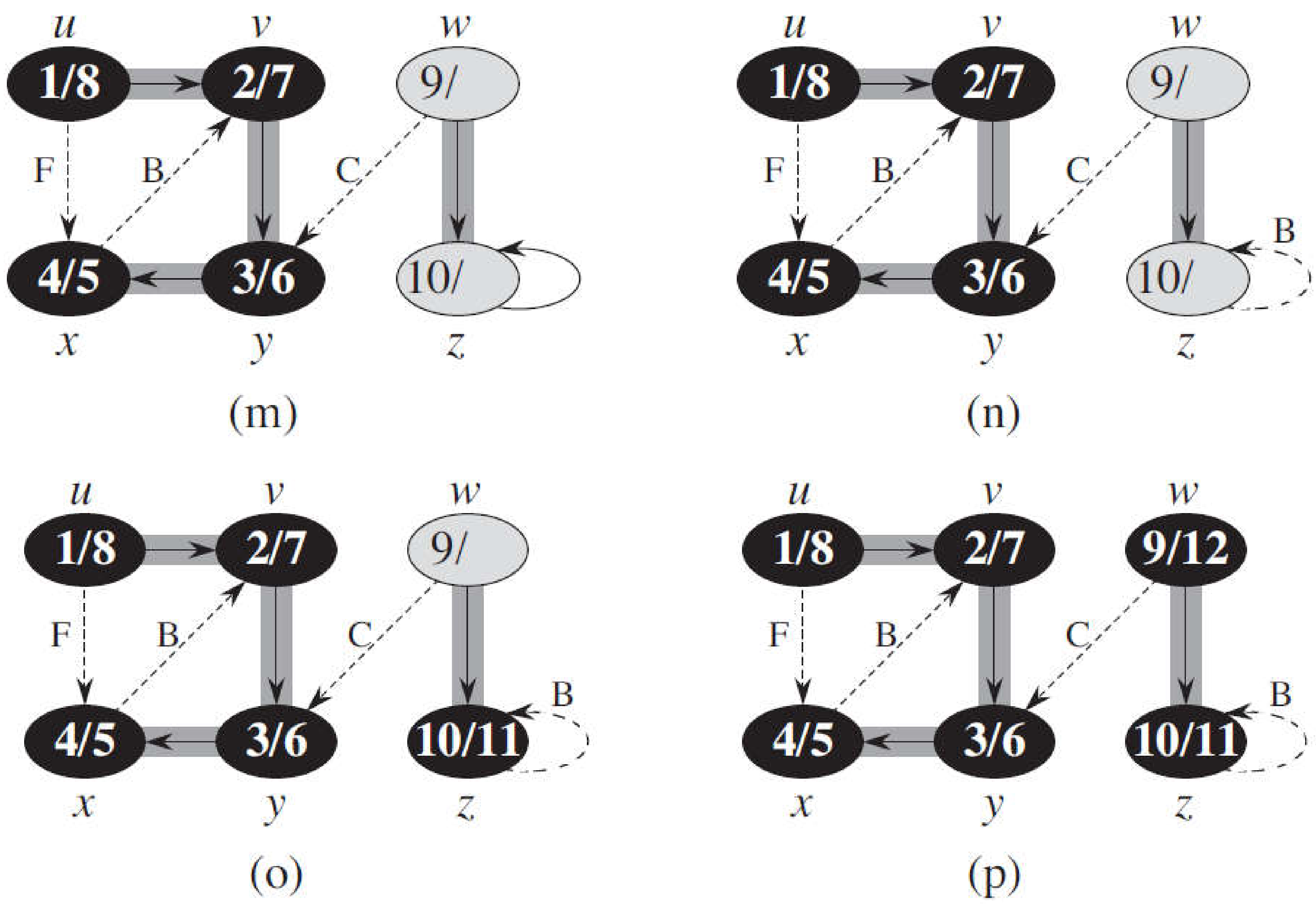


(k) (l)

```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

مثال از جستجوی عمق اول (ادامه)



شکل ۲۲-۴ کتاب CLRS

تحلیل جستجوی عمق اول

- مرتبه زمانی؟

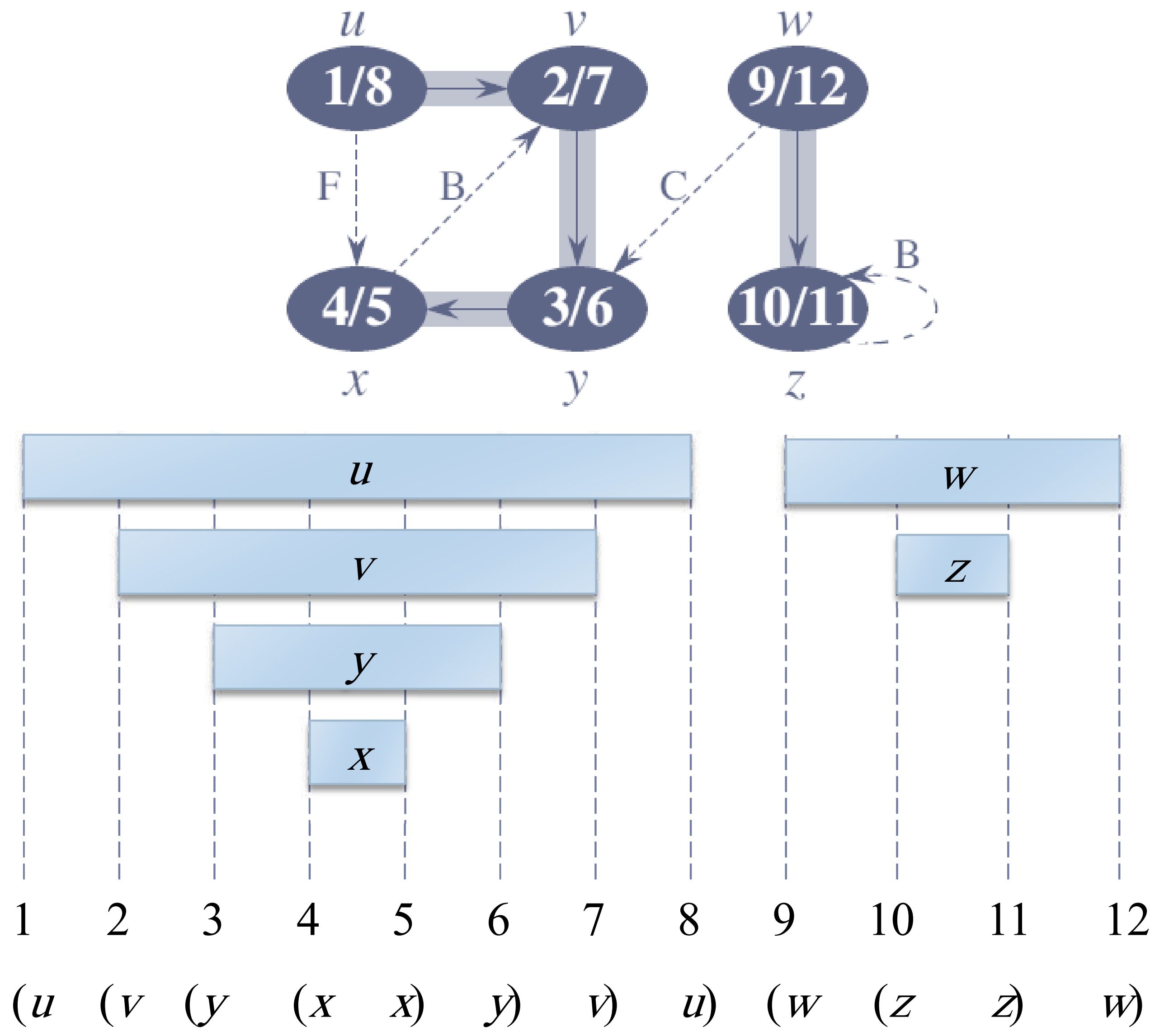
```
DFS(G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK
time = time + 1
u.f = time
```

ساختار پرانتزگذاری در جستجوی عمق اول

- زمان‌های شروع و پایان ملاقات در جستجوی عمق اول دارای ساختار پرانتزگذاری هستند.
- شروع ملاقات u را با ‘$(u$’
- پایان ملاقات u را با ‘$u)$’

ساختار پرانتزگذاری در جستجوی عمق اول (ادامه)



ساختار پرانتزگذاری در جستجوی عمق اول (ادامه)

- نتیجه:
- در پیمایش DFS برای هر دو رأس u و v دقیقاً فقط یکی از شرایط زیر برقرار است:

1. بازه $[u.d, u.f]$ و بازه $[v.d, v.f]$ هیچ اشتراکی با هم ندارند. u و v جد یا نواده هم نیستند.
2. $v.d< u.d< u.f< v.f$ و u نواده v است ← پرانتزگذاری به صورت ([ ]).
3. $u.d< v.d< v.f< u.f$ و v نواده u است ← پرانتزگذاری به صورت [( )].

- بنابراین هیچگاه حالت $u.d < v.d < u.f < v.f$ بوجود نمی‌آید یعنی نمی‌توانیم فرم [( ]) را داشته باشیم.

دسته‌بندی یال‌ها در DFS

- یال درختی (tree edge): (از رأس خاکستری به رأس سفید)
  - یالی است که DFS آن را ملاقات می‌کند.
- یال پس‌سو (back edge): (از رأس خاکستری به رأس خاکستری)
  - یال (u, v) که DFS آن را ملاقات نمی‌کند و v والد u است.
- یال پیش‌سو (forward edge): (از رأس خاکستری به رأس سیاه)
  - یال (u, v) که DFS آن را ملاقات نمی‌کند و u والد v است.
- یال چپ‌سو/عبوری (cross edge): (از رأس خاکستری به رأس سیاه)
  - هیچکدام از حالت‌های فوق نیست، بین گره‌های دو زیر درخت اتفاق می‌افتد.

دسته‌بندی یال‌ها در DFS (ادامه)

- رابطه‌ی بین زمان شروع و پایان ملاقات گره‌ها و انواع یال‌ها:
- برای یال (u, v) داریم:
- یالی را یال درختی یا پیش‌سو گویند اگر و تنها اگر داشته باشیم:

$$u.d < v.d < v.f < u.f$$

- یالی را پس‌سو گویند اگر و تنها اگر داشته باشیم:

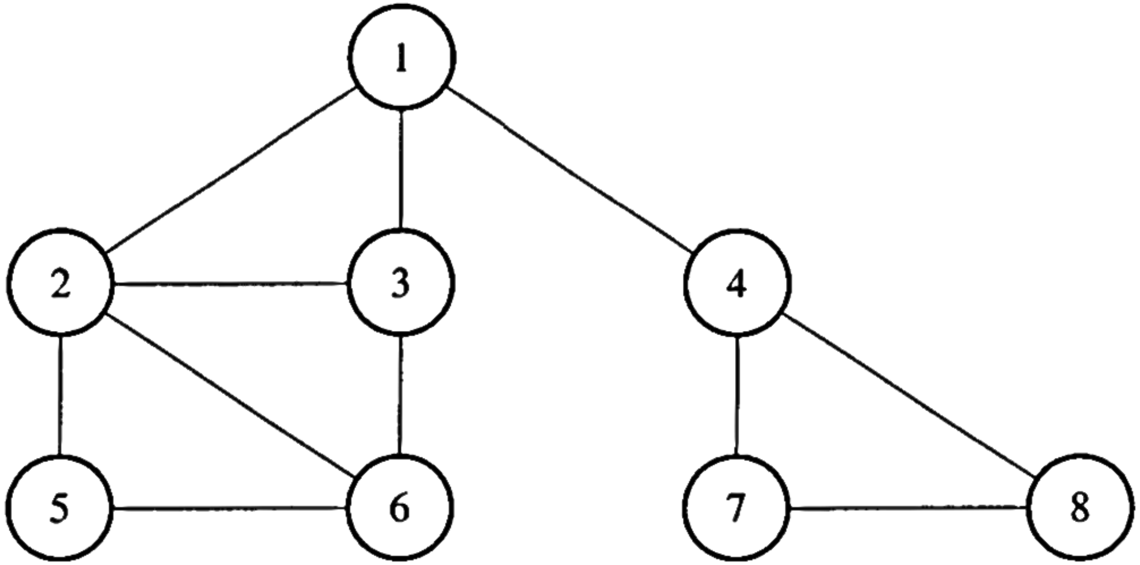
$$v.d < u.d < u.f < v.f$$

- یالی را چپ‌سو گویند اگر و تنها اگر داشته باشیم:

$$v.d < v.f < u.d < u.f$$

تمرین ۲

- جستجوی عمق اول را اجرا کرده و انواع یال‌ها را مشخص سازید. چه نتیجه‌ای می‌توان گرفت؟



کاربرد جستجوی اول عمق

- **لم ۱:** شرط لازم و کافی برای این‌که یک گراف بدون جهت، دور داشته باشد آن است که DFS، یال پس‌سو داشته باشد.
- باید ثابت کنیم:

1. اگر DFS یال پس‌سو داشته باشد آنگاه گراف دور دارد.
2. اگر گراف دور داشته باشد، آنگاه در جستجوی عمق‌اول آن، یال پس‌سو وجود دارد.

- اثبات بخش یک:
  - فرض کنید یک یال پس‌سو (u,v) وجود داشته باشد، آنگاه رأس v یک جد رأس u در جنگل اول عمق است، بنابراین یک مسیر از v به u در G وجود دارد و یال پس‌سو باعث دور می‌گردد.

کاربرد جستجوی اول عمق (ادامه)

- **لم ۱:** شرط لازم و کافی برای این‌که یک گراف بدون جهت، دور داشته باشد آن است که DFS، یال پس‌سو داشته باشد.
- باید ثابت کنیم:

1. اگر DFS یال پس‌سو داشته باشد آنگاه گراف دور دارد.
2. اگر گراف دور داشته باشد، آنگاه در جستجوی عمق‌اول آن، یال پس‌سو وجود دارد. (اگر در جستجوی عمق‌اول یال پس‌سو وجود نداشته باشد، آنگاه گراف دور ندارد)

- اثبات بخش دو:
  - اگر گراف بدون جهت یال پس‌سو نداشته باشد، در نتیجه تمامی یال‌ها، یال درختی است و درخت نیز دور ندارد.

کاربرد جستجوی اول عمق (ادامه)

- **لم ۲:** گراف جهت‌دار G، بدون دور است، اگر و فقط اگر از جستجوی اول عمق G هیچ یال پس‌سو حاصل نگردد.
- اثبات: باید ثابت کنیم

1. اگر DFS یال پس‌سو داشته باشد آنگاه گراف دور دارد.
2. اگر گراف دور داشته باشد، آنگاه در جستجوی عمق‌اول آن، یال پس‌سو وجود دارد.

- اثبات بخش یک:

  - فرض کنید یک یال پس‌سو (u,v) وجود داشته باشد، آنگاه رأس v یک جد رأس u در جنگل اول عمق است، بنابراین یک مسیر از v به u در G وجود دارد و یال پس‌سو باعث دور می‌گردد.

کاربرد جستجوی اول عمق (ادامه)

- **لم ۲:** گراف جهت‌دار G، بدون دور است، اگر و فقط اگر از جستجوی اول عمق G هیچ یال پس‌سو حاصل نگردد.
- اثبات: باید ثابت کنیم
  1. اگر DFS یال پس‌سو داشته باشد آنگاه گراف دور دارد.
  2. اگر گراف دور داشته باشد، آنگاه در جستجوی عمق‌اول آن، یال پس‌سو وجود دارد.
- اثبات بخش دو:
  - فرض کنید G، شامل دور C باشد و فرض کنید v اولین رأسی باشد که در دور C ملاقات می‌شود و u رأس پدر او در دور باشد، در زمان ملاقات v بقیه‌ی رأس‌های C سفید هستند، ملاقات v تمام نمی‌شود تا همه‌ی رأس‌های سفید قابل دسترسی از v سیاه شوند، در نتیجه (u, v) یال پس‌سو است.

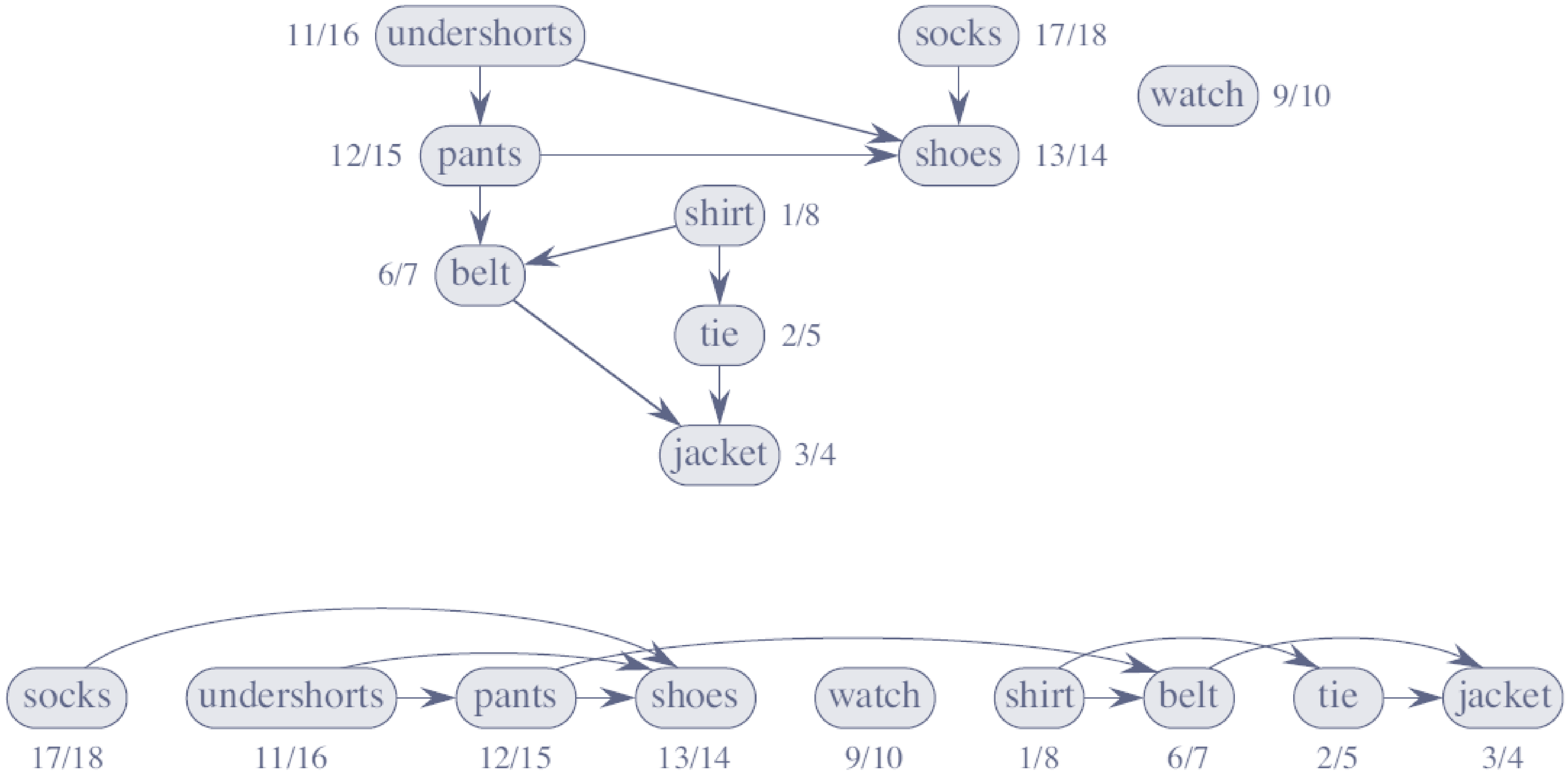
کاربرد جستجوی اول عمق

- **مرتب‌سازی توپولوژیکی:**
  - ورودی: یک گراف جهت‌دار بدون دور (DAG*)
  - خروجی: ترتیبی از رأس‌ها به طوری که برای هر یال (u, v) در گراف، رأس u قبل از v بیاید.
  - اگر گراف دور داشته باشد، ترتیب توپولوژیکی امکان‌پذیر نیست.

* Directed Acyclic Graph

مثال ۱ مرتب‌سازی توپولوژیکی

- ترتیبی که پروفسور بامس‌تِد باید لباس‌های خود را بپوشد:



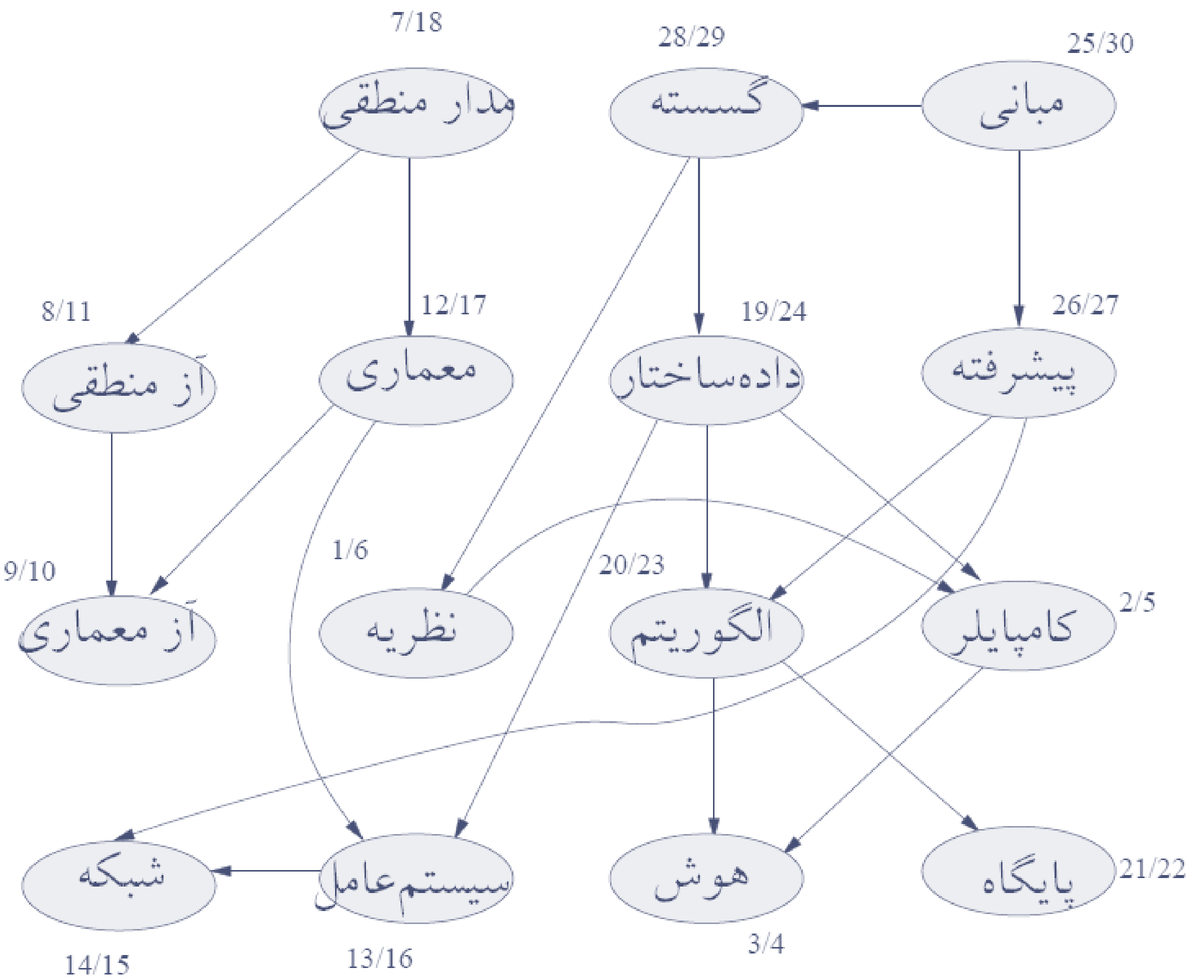
شکل ۲۲-۷ کتاب CLRS

الگوریتم مرتب‌سازی توپولوژیکی و تحلیل آن

```
TOPOLOGICAL-SORT(G)
call DFS(G) to compute finishing times v.f for each vertex v
as each vertex is finished, insert it onto the front of a linked list
return the linked list of vertices
```

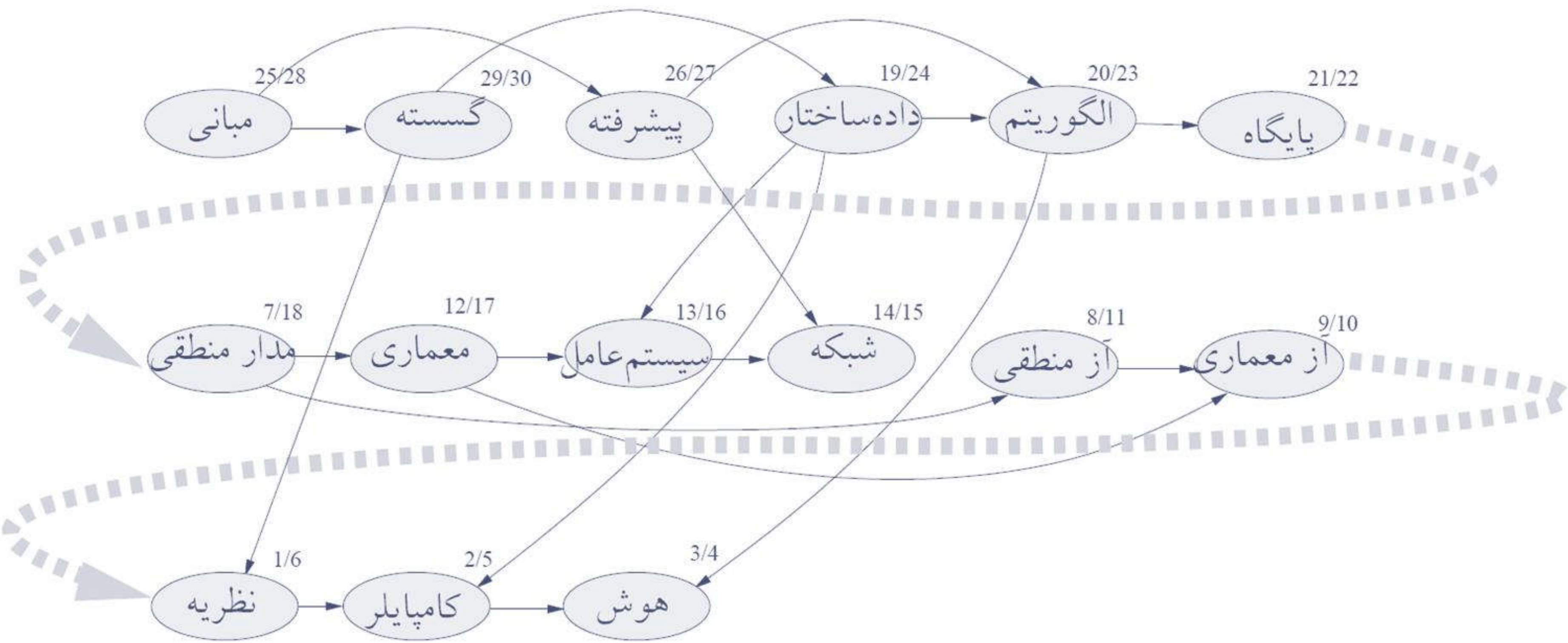
- مرتبه DFS؟
- مرتبه درج در لیست؟
- مرتبه کل؟

مثال ۲ مرتب‌سازی توپولوژیکی



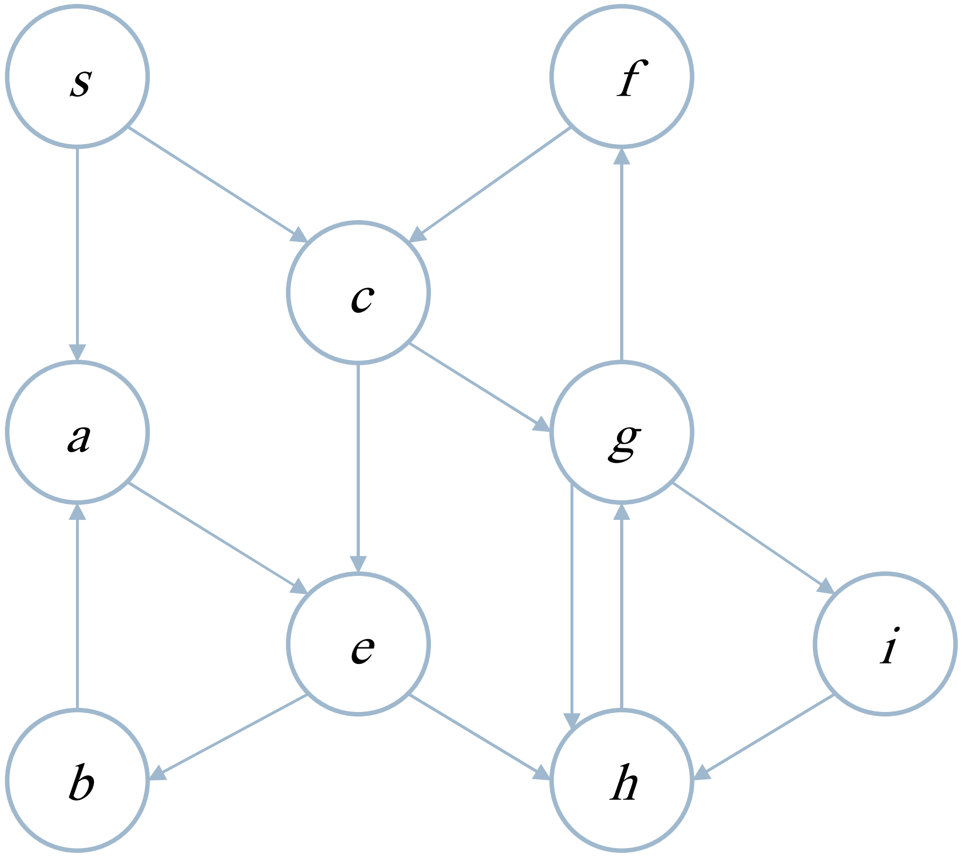
مثال ۲ مرتب‌سازی توپولوژیکی

- ترتیب توپولوژیکی



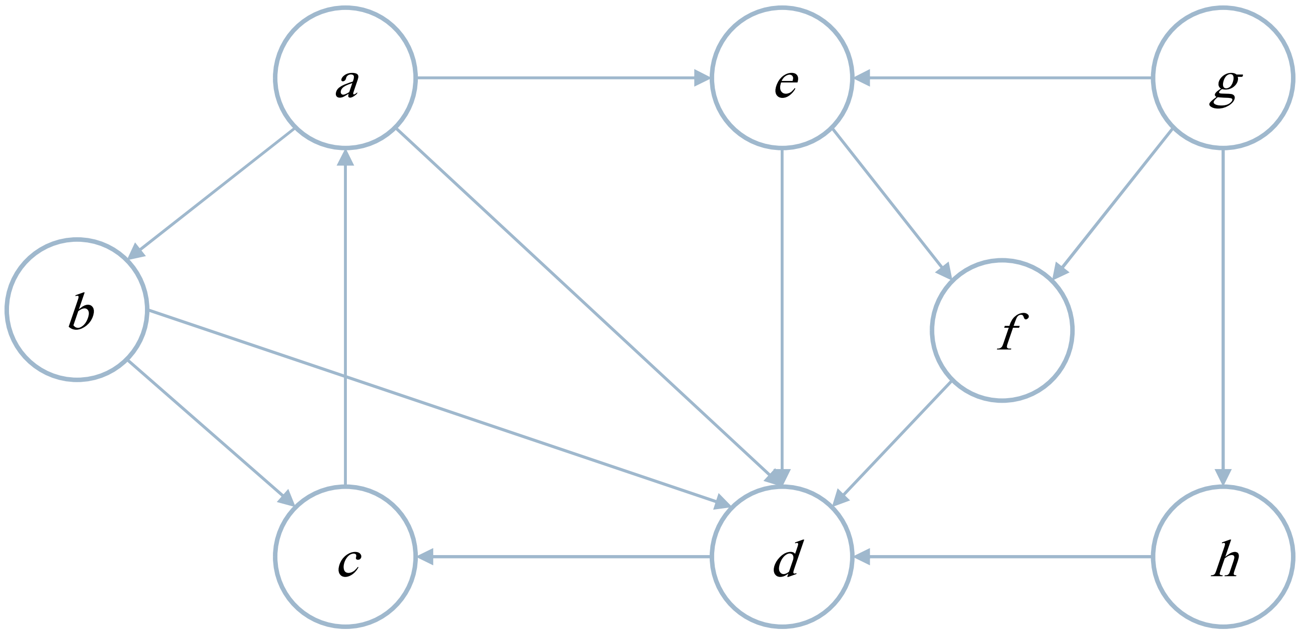
تمرین بیشتر

1. جستجوی BFS را انجام دهید. نود شروع، رأس s است.



تمرین بیشتر

2. جستجوی DFS را انجام داده و یال‌های پس‌رو، پیش‌رو و چپ‌رو را تعیین کنید (نود شروع: a).



The END