

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



طراحی و تحلیل الگوریتم‌ها

مبحث دهم

مطالعه‌ی موضوعی الگوریتم‌ها

الگوریتم‌های گراف

Graph Algorithms

کاظم فولادی قلعه

دانشکده مهندسی، دانشکدگان فارابی

دانشگاه تهران

<http://courses.fouladi.ir/algorithm>

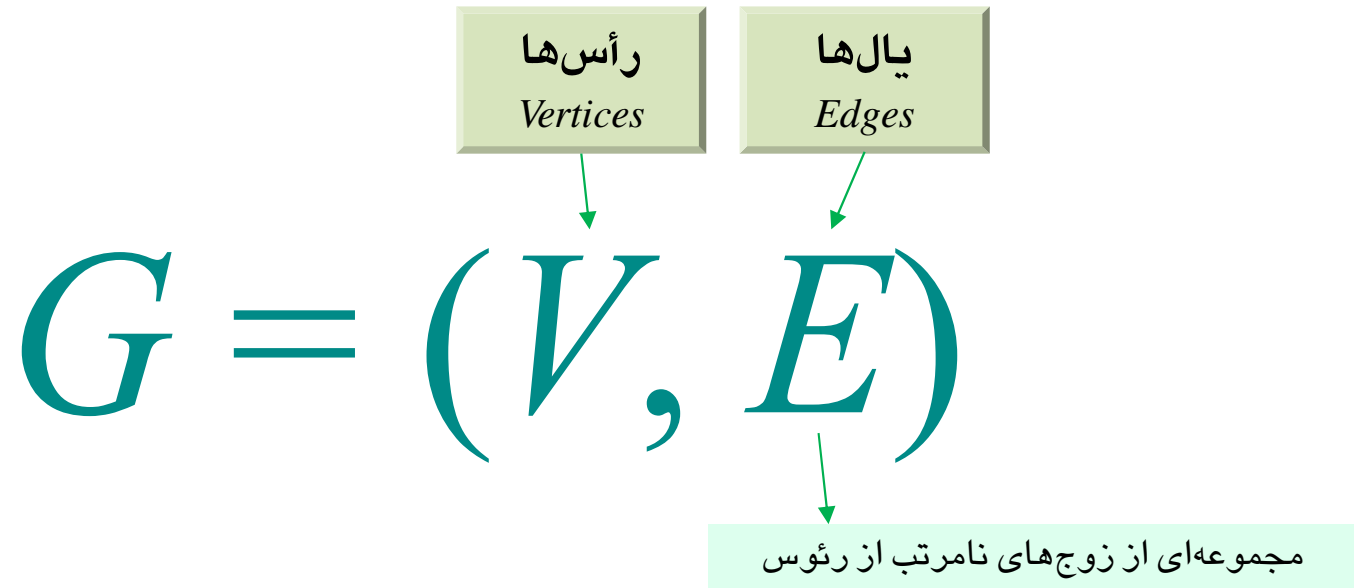
گراف

کاربردها

Example	Nodes	Edges
Transportation network: airline routes	airports	nonstop flights
Communication networks	computers, hubs, routers	physical wires
Information network: web	pages	hyperlinks
Information network: scientific papers	articles	references
Social networks	people	“u is v’s friend”, “u sends email to v”, “u’s MySpace page links to v”
Computer programs	functions (or modules) statement blocks	“u calls v” “v can follow u”

بازنمایی گراف

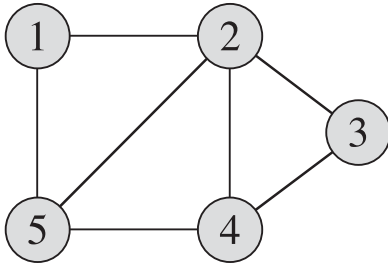
گراف بدون جهت

UNDIRECTED GRAPH

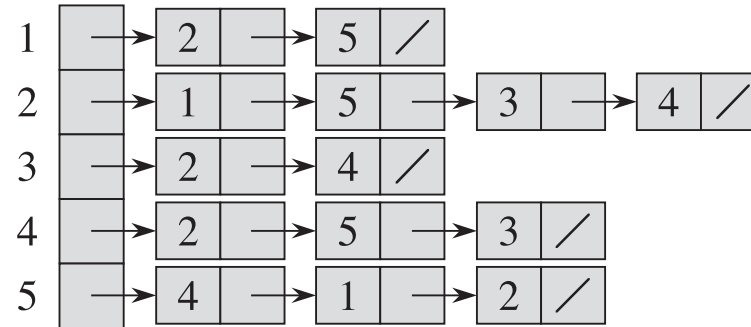
بازنمایی گراف

گراف بدون جهت: مثال

GRAPH REPRESENTATION



(a)



(b)

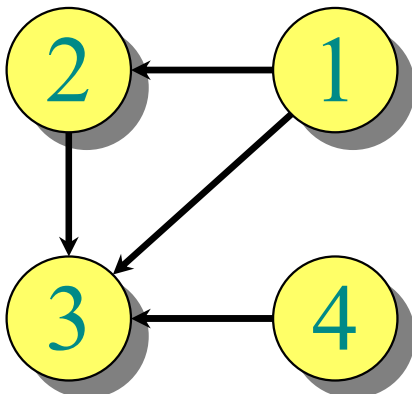
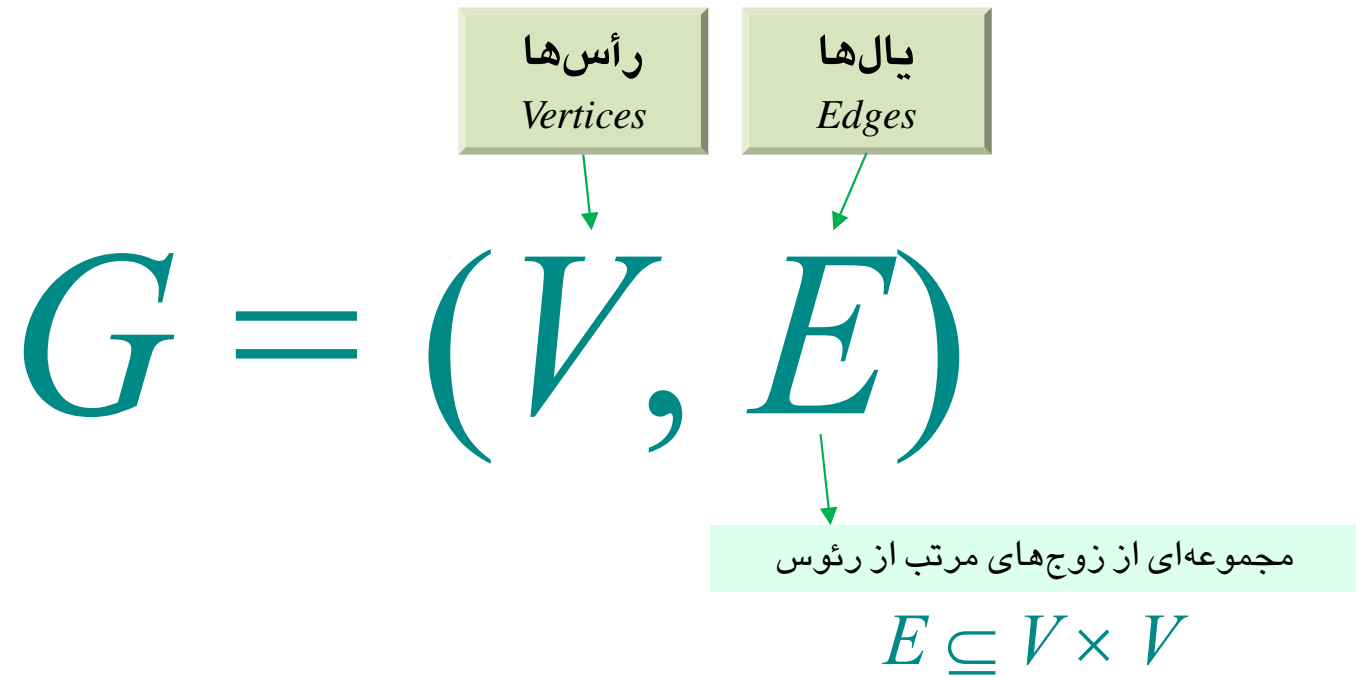
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

بازنمایی گراف

گراف جهت‌دار

DIRECTED GRAPH



بازنمایی گراف

گراف جهت‌دار

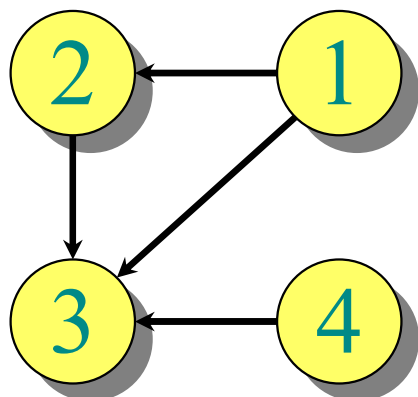
GRAPH REPRESENTATION

$$G = (V, E)$$

$$V = \{1, 2, \dots, n\} \quad E \subseteq V \times V$$

$$A[1 \dots n, 1 \dots n]$$

$$A[i, j] = \begin{cases} 1 & \text{if } (i, j) \in E, \\ 0 & \text{if } (i, j) \notin E. \end{cases}$$



ماتریس مجاورت
Adjacency Matrix

A	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0

بازنمایی گراف

گراف جهت‌دار

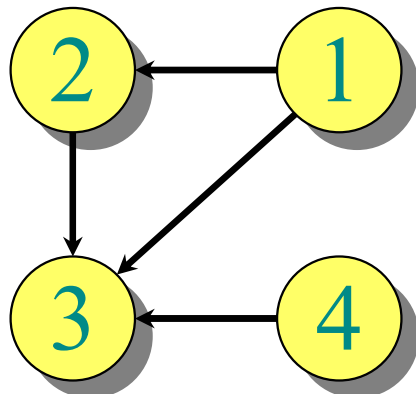
GRAPH REPRESENTATION

$$G = (V, E)$$

$$V = \{1, 2, \dots, n\}$$

$$E \subseteq V \times V$$

An **adjacency list** of a vertex $v \in V$ is the list $Adj[v]$ of vertices adjacent to v



لیست مجاورت
Adjacency List

For undirected graphs, $|Adj[v]| = degree(v)$.
For digraphs, $|Adj[v]| = out-degree(v)$.

$$Adj[1] = \{2, 3\}$$

$$Adj[2] = \{3\}$$

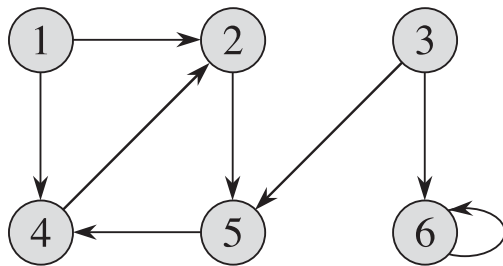
$$Adj[3] = \{\}$$

$$Adj[4] = \{3\}$$

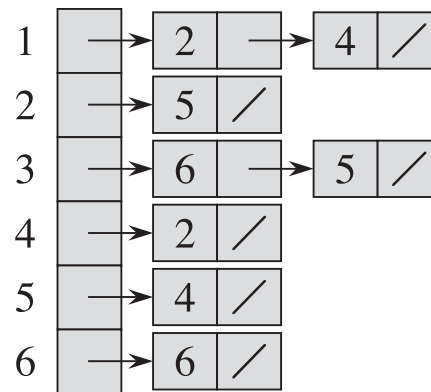
بازنمایی گراف

گراف جهت‌دار: مثال

GRAPH REPRESENTATION



(a)



(b)

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

گراف

رابطه‌ی رأس‌ها و یال‌ها

$$G = (V, E)$$

$$|E| = O(V^2)$$

در گراف همبند:

$$|E| \geq |V| - 1$$

$$\lg |E| = \Theta(\lg V)$$

گراف

لم «دست دادن»

HANDSHAKING LEMMA

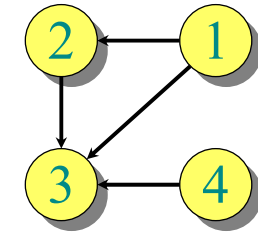
در گراف بدون جهت

$$\sum_{v \in V} \text{degree}(v) = 2|E|$$

بازنمایی گراف

گراف پر و گراف خلوت

GRAPH REPRESENTATION



فضای مصرفی

$$\Theta(V^2)$$

بازنمایی پر
Dense Representation

ماتریس مجاورت
Adjacency Matrix

A	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0

$$\Theta(V + E)$$

بازنمایی خلوت
Sparse Representation

لیست مجاورت
Adjacency List

$Adj[1] = \{2, 3\}$
 $Adj[2] = \{3\}$
 $Adj[3] = \{\}$
 $Adj[4] = \{3\}$

مسیر

PATH

یک دنباله از یال‌های مجاور متمایز در یک گراف

مسیر
Path

$$(u, w_1), (w_1, w_2), (w_2, w_3), \dots, (w_{k-1}, v)$$

$$u \rightsquigarrow v$$

نکته: در گراف جهت‌دار، جهت یال‌ها در مسیر مهم است.

مسیر ساده

SIMPLE PATH

یک مسیر بدون رأس تکراری

مسیر ساده
Simple Path

$$(u, w_1), (w_1, w_2), (w_2, w_3), \dots, (w_{k-1}, v)$$

$$u \rightsquigarrow v$$

$$Alldiffs(u, w_1, w_2, w_3, \dots, w_{k-1}, v)$$

همبندی

CONNECTIVITY

گراف بدون جهت	گرافی که بین هر دو رأس دلخواه آن مسیری وجود داشته باشد.	گراف همبند <i>Connected Graph</i>
گراف جهت‌دار	گرافی که گراف بدون جهت متناظر با آن (پس از حذف جهت‌ها) همبند باشد.	

گراف جهت‌دار	گرافی که بین هر دو رأس دلخواه آن مسیری جهت‌دار وجود داشته باشد.	گراف همبند قوی <i>Strongly Connected Graph</i>
--------------	---	---

گراف بدون جهت	گراف همبندی که پس از حذف یک یال دلخواه از آن همچنان همبند بماند.	گراف دوهمبند <i>Biconnected Graph</i>
---------------	--	--

در گراف همبند:

$$|E| \geq |V| - 1$$

دور (چرخه)

CYCLE

یک مسیر که رئوس ابتدا و انتهای آن یکسان است.

دور
Cycle

$$(u, w_1), (w_1, w_2), (w_2, w_3), \dots, (w_{k-1}, u)$$

$$u \rightsquigarrow u$$

نکته: در گراف جهت‌دار، جهت یال‌ها در مسیر مهم است.

دور ساده (چرخه‌ی ساده)

SIMPLE CYCLE

دور ساده
Simple Cycle

دوری که رأس تکراری ندارد (به جز رئوس ابتدا و انتهای آن).

$$(u, w_1), (w_1, w_2), (w_2, w_3), \dots, (w_{k-1}, u)$$

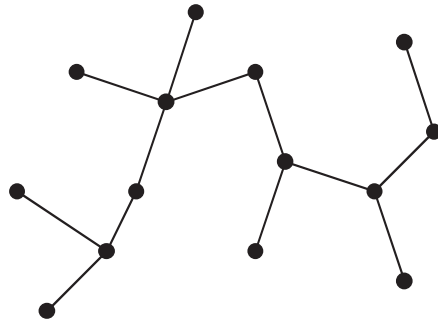
$$u \rightsquigarrow u$$

$$Alldiffs(w_1, w_2, w_3, \dots, w_{k-1})$$

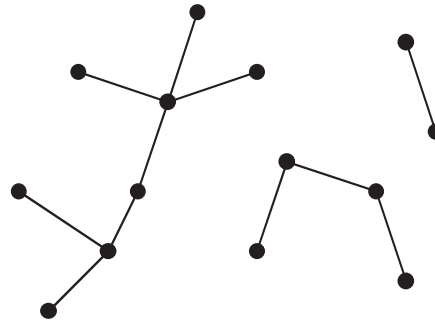
درخت

TREE

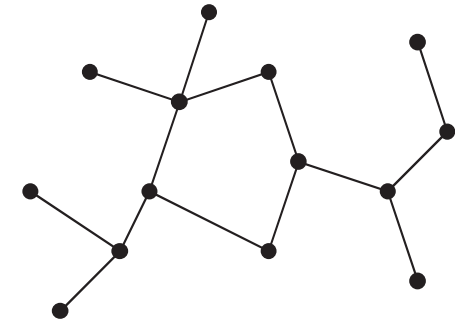
گراف همبند بدون دور

درخت
Tree

(a)

گراف همبند بدون دور
= درخت

(b)

گراف ناهمبند بدون دور
= جنگل

(c)

گراف همبند دارای دور
(درخت نیست)

درخت

خصوصیات درخت‌های آزاد

PROPERTIES OF FREE TREES

گراف همبند بدون دور

درخت
*Tree****Theorem B.2 (Properties of free trees)***

Let $G = (V, E)$ be an undirected graph. The following statements are equivalent.

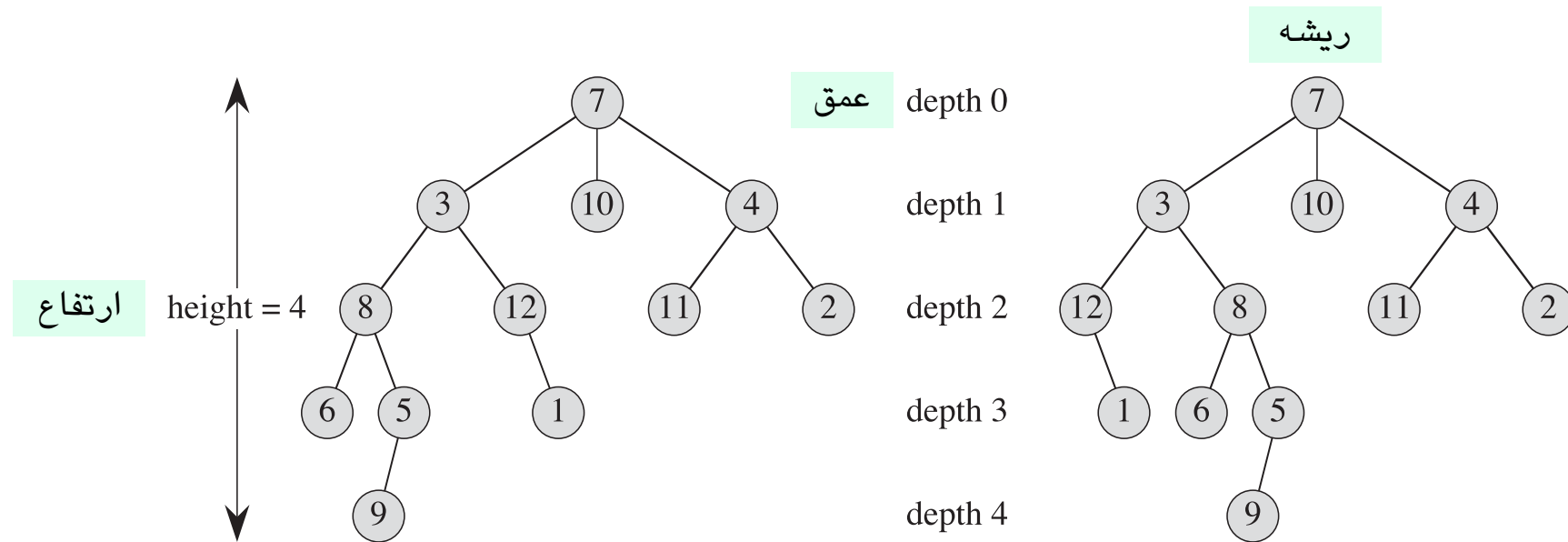
1. G is a free tree.
2. Any two vertices in G are connected by a unique simple path.
3. G is connected, but if any edge is removed from E , the resulting graph is dis-connected.
4. G is connected, and $|E| = |V| - 1$.
5. G is acyclic, and $|E| = |V| - 1$.
6. G is acyclic, but if any edge is added to E , the resulting graph contains a cycle.

درخت ریشه‌دار مرتب

ORDERED ROOTED TREE

یک درخت جهت‌دار که یک گرهی خاص به نام ریشه دارد (بدون یال ورودی) و ترتیب گره‌های فرزند هر گره اهمیت دارد.

درخت ریشه‌دار مرتب
Ordered Rooted Tree

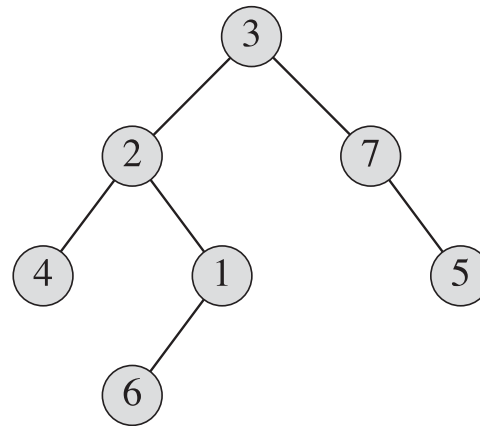
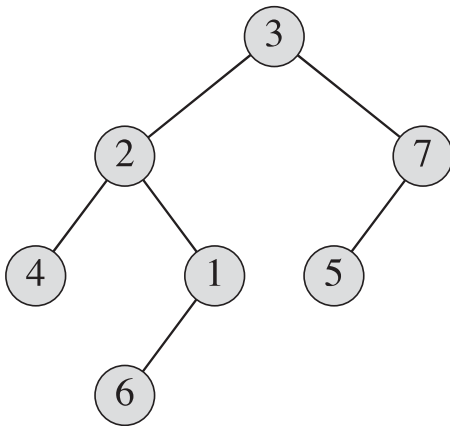


درخت دودویی

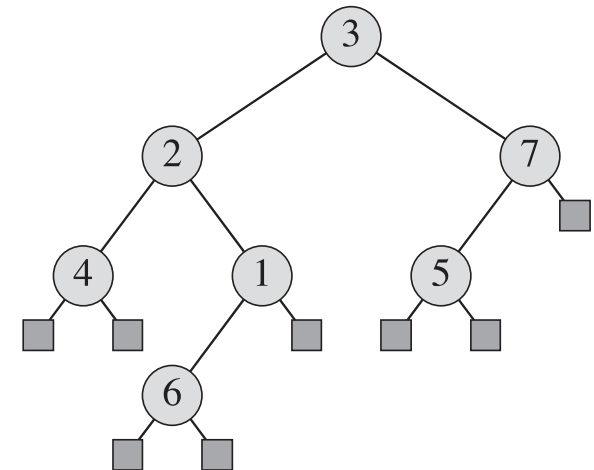
BINARY TREE

یک درخت ریشه‌دار مرتب که هر گرهی آن حداکثر دو فرزند دارد.

درخت دودویی
Binary Tree



برگ

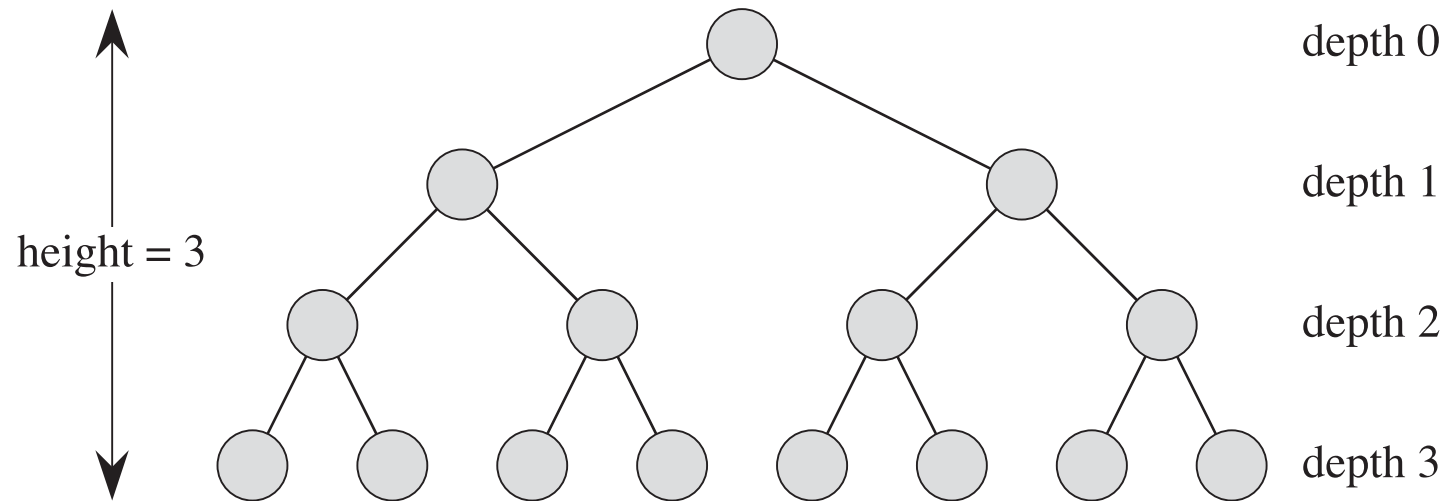


درخت دودویی کامل

COMPLETE BINARY TREE

یک درخت دودویی که همه‌ی گره‌های آن به جز سطح آخر، دو فرزند دارند.

درخت دودویی کامل
Complete Binary Tree



A complete binary tree of height 3 with 8 leaves and 7 internal nodes.

درخت k -تایی

مثال

 K -ARY TREE

یک درخت ریشه‌دار مرتب که هر گره‌ی آن حداکثر k فرزند دارد.

درخت k -تایی
 k -ary Tree

حداکثر تعداد گره‌های داخلی (internal nodes) در یک درخت k -تایی با ارتفاع h :

$$1 + k + k^2 + \dots + k^{h-1} = \sum_{i=0}^{h-1} k^i = \frac{k^h - 1}{k - 1}$$

درخت k -تایی

مثال

 K -ARY TREE

یک درخت ریشه‌دار مرتب که هر گره‌ی آن حداکثر k فرزند دارد.

درخت k -تایی
 k -ary Tree

حداکثر تعداد گره‌ها در یک درخت k -تایی با ارتفاع h :

$$1 + k + k^2 + \dots + k^h = \sum_{i=0}^h k^i = \frac{k^{h+1} - 1}{k - 1}$$

پیمایش گراف

GRAPH TRAVERSAL

دو رأس دلخواه $s, t \in V$ از گراف $G = (V, E)$ داده شده است.
آیا مسیری از s به t وجود دارد؟

راه حل: همه‌ی رئوس قابل دسترس از رأس s را پیمایش می‌کنیم.

تکنیک‌های پیمایش گراف

جستجوی عمق-اول

Depth-First Search (DFS)

فرزندان یک گره
به صورت بازگشتی پیش از همزادهای آن
پیمایش می‌شوند.

جستجوی عرض-اول

Breadth-First Search (BFS)

فرزندان یک گره به ترتیب
فاصله از گره‌ی شروع
پیمایش می‌شوند.

جستجوی عرض-اول

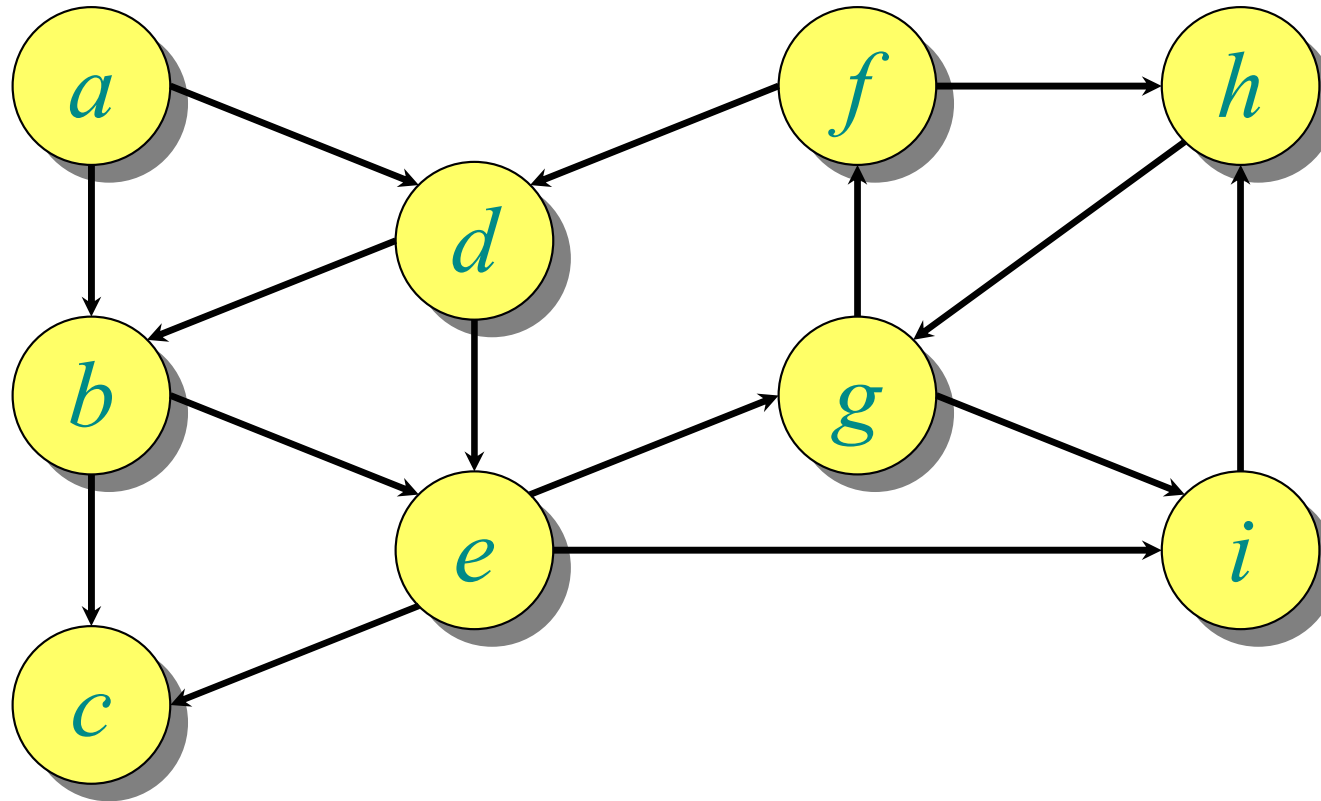
Breadth-first search

```
while  $Q \neq \emptyset$ 
do  $u \leftarrow \text{DEQUEUE}(Q)$ 
  for each  $v \in \text{Adj}[u]$ 
  do if  $d[v] = \infty$ 
    then  $d[v] \leftarrow d[u] + 1$ 
         $\text{ENQUEUE}(Q, v)$ 
```

Analysis: Time = $O(V + E)$

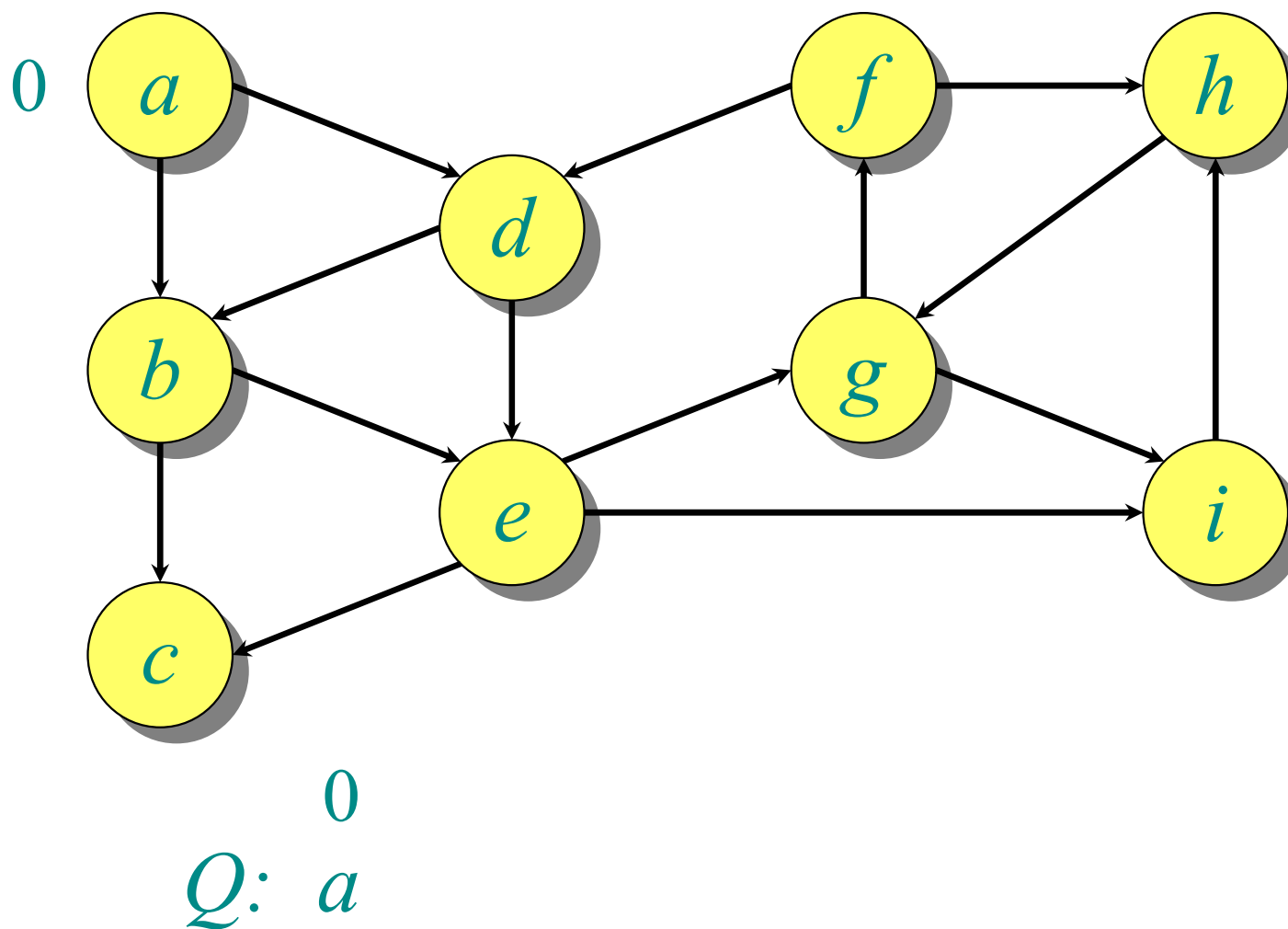
جستجوی عرض-اول

مثال (۱ از ۱۲)

 $Q:$

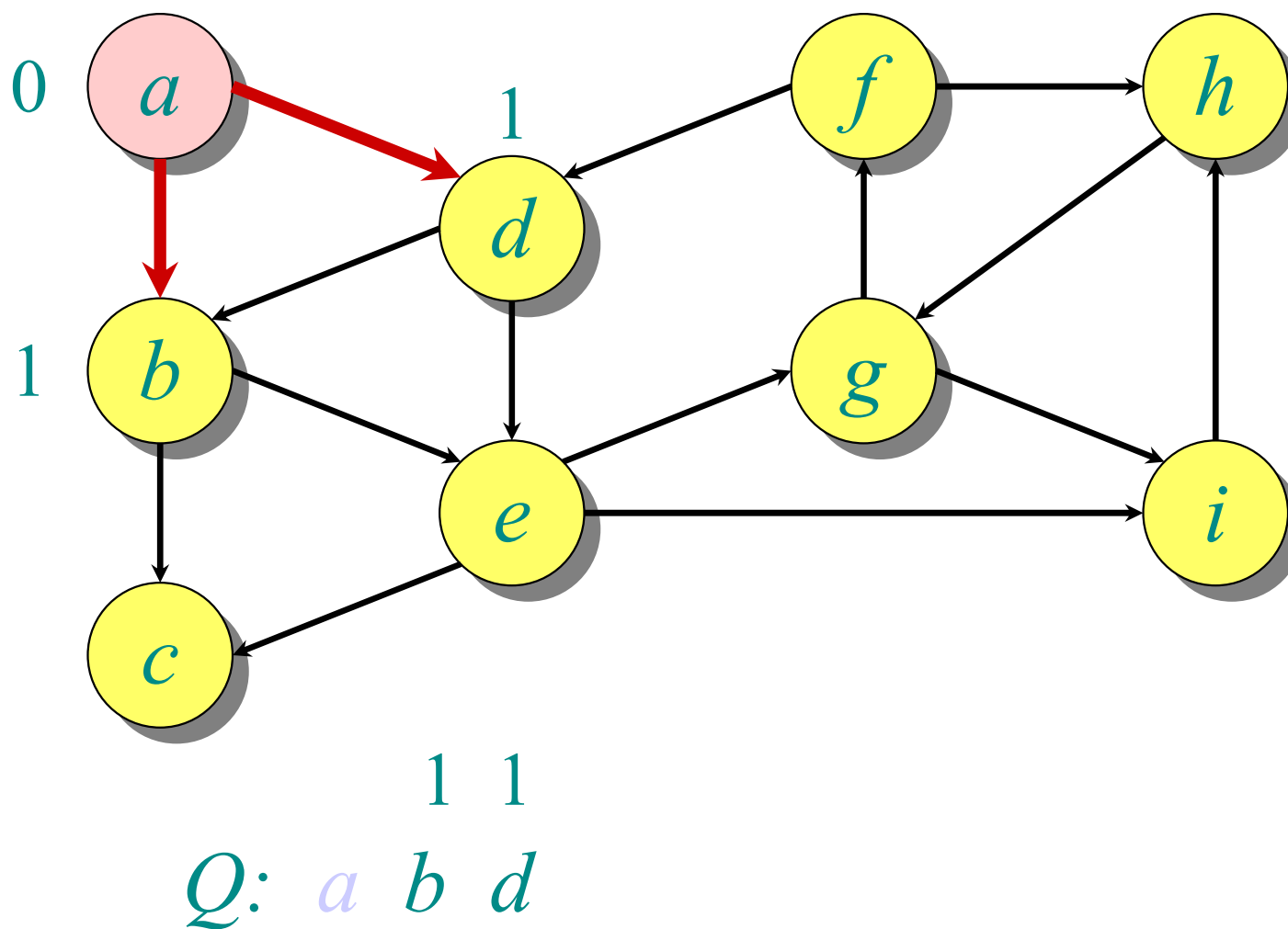
جستجوی عرض-اول

مثال (۲ از ۱۲)



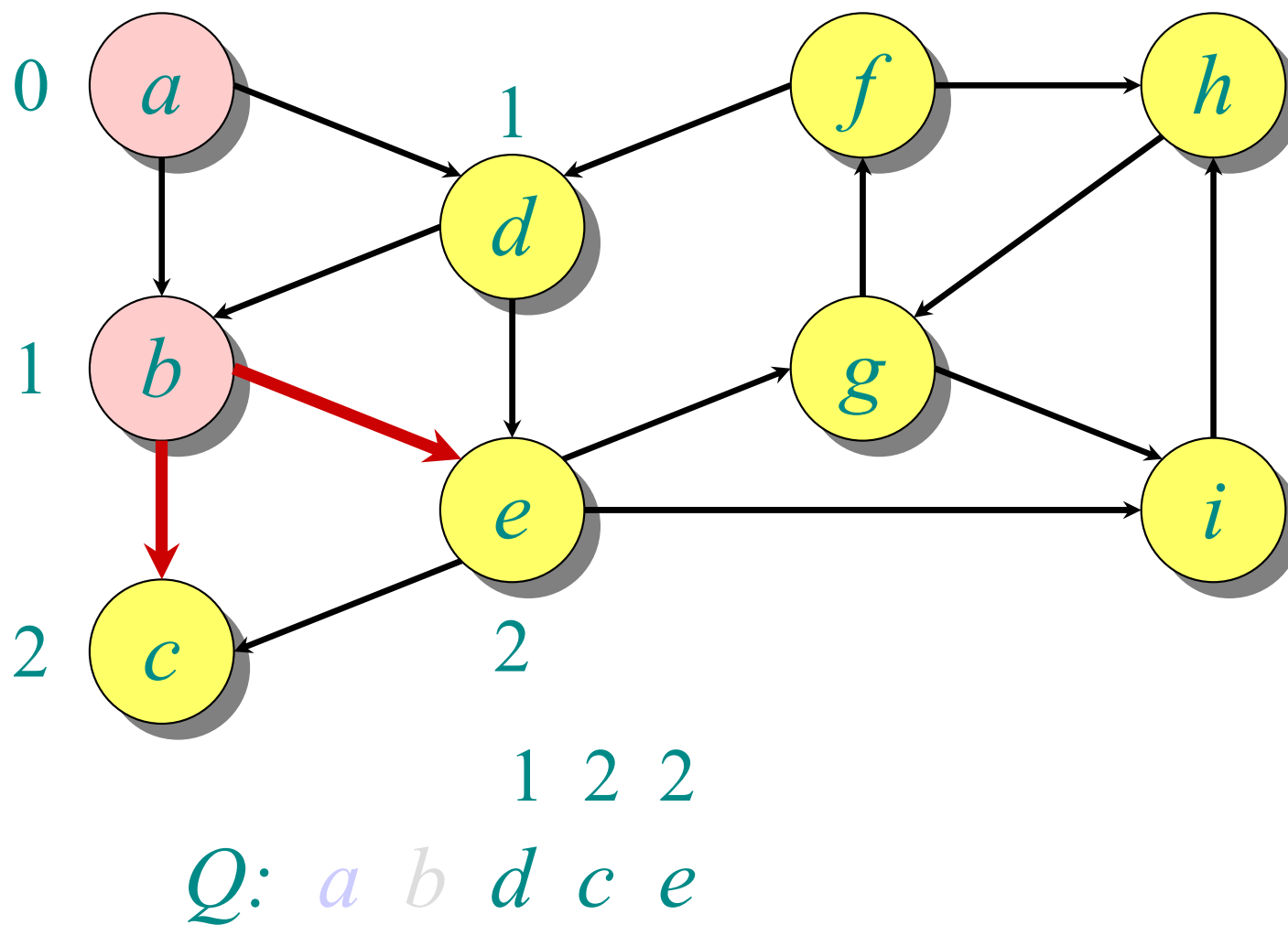
جستجوی عرض-اول

مثال (۳ از ۱۲)



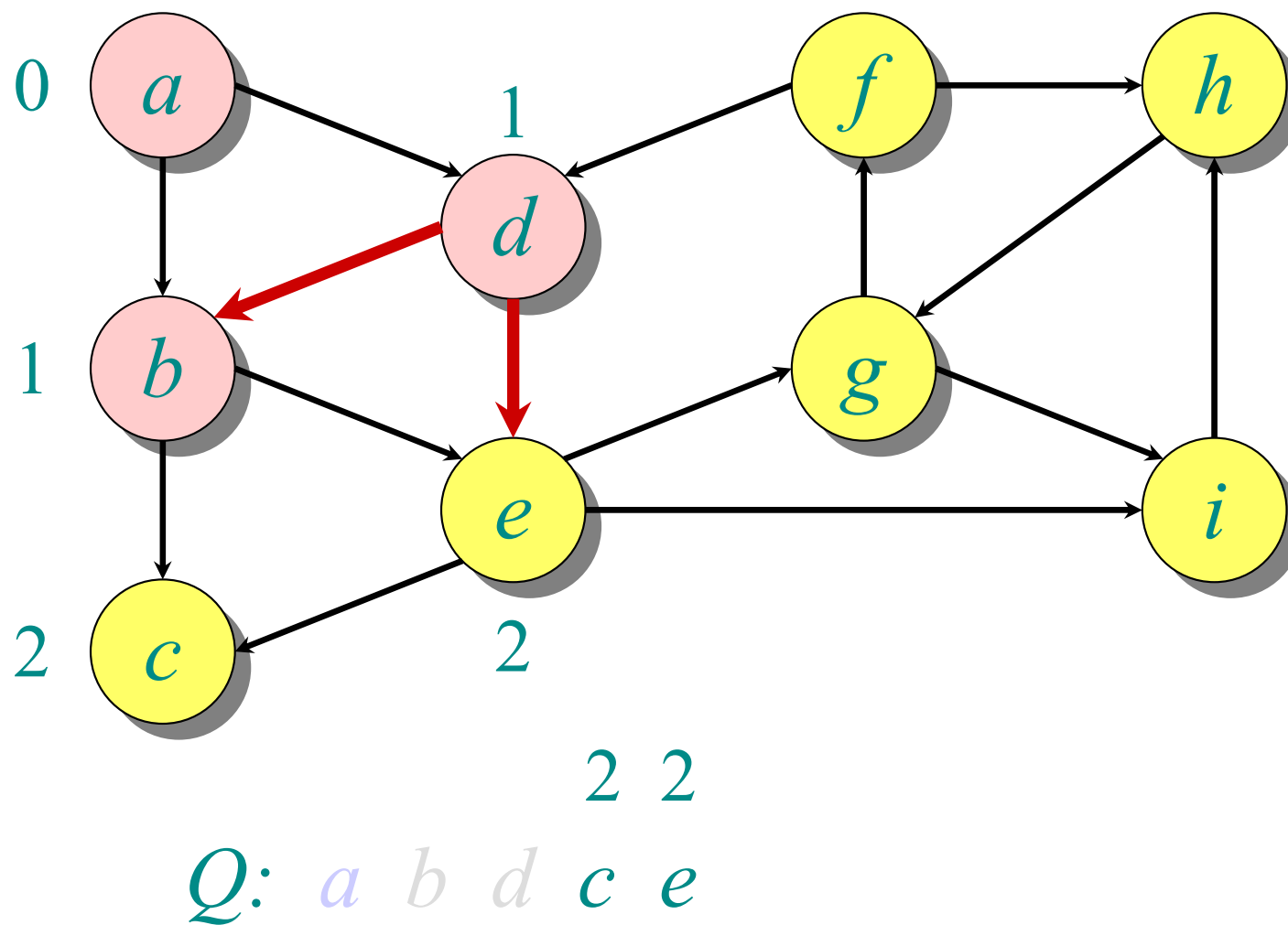
جستجوی عرض-اول

مثال (۴ از ۱۲)



جستجوی عرض-اول

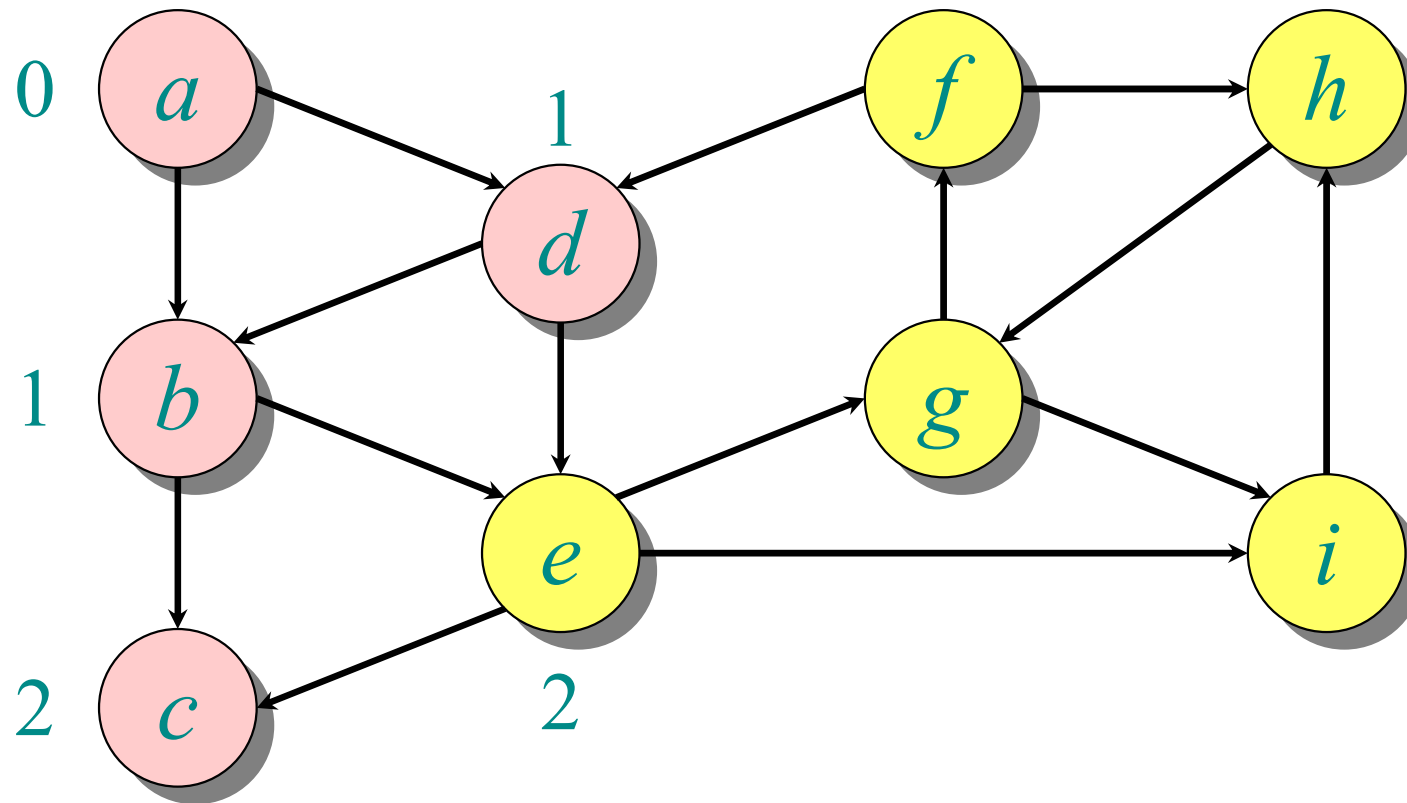
مثال (۵ از ۱۲)



$Q: a \ b \ d \ c \ e$

جستجوی عرض-اول

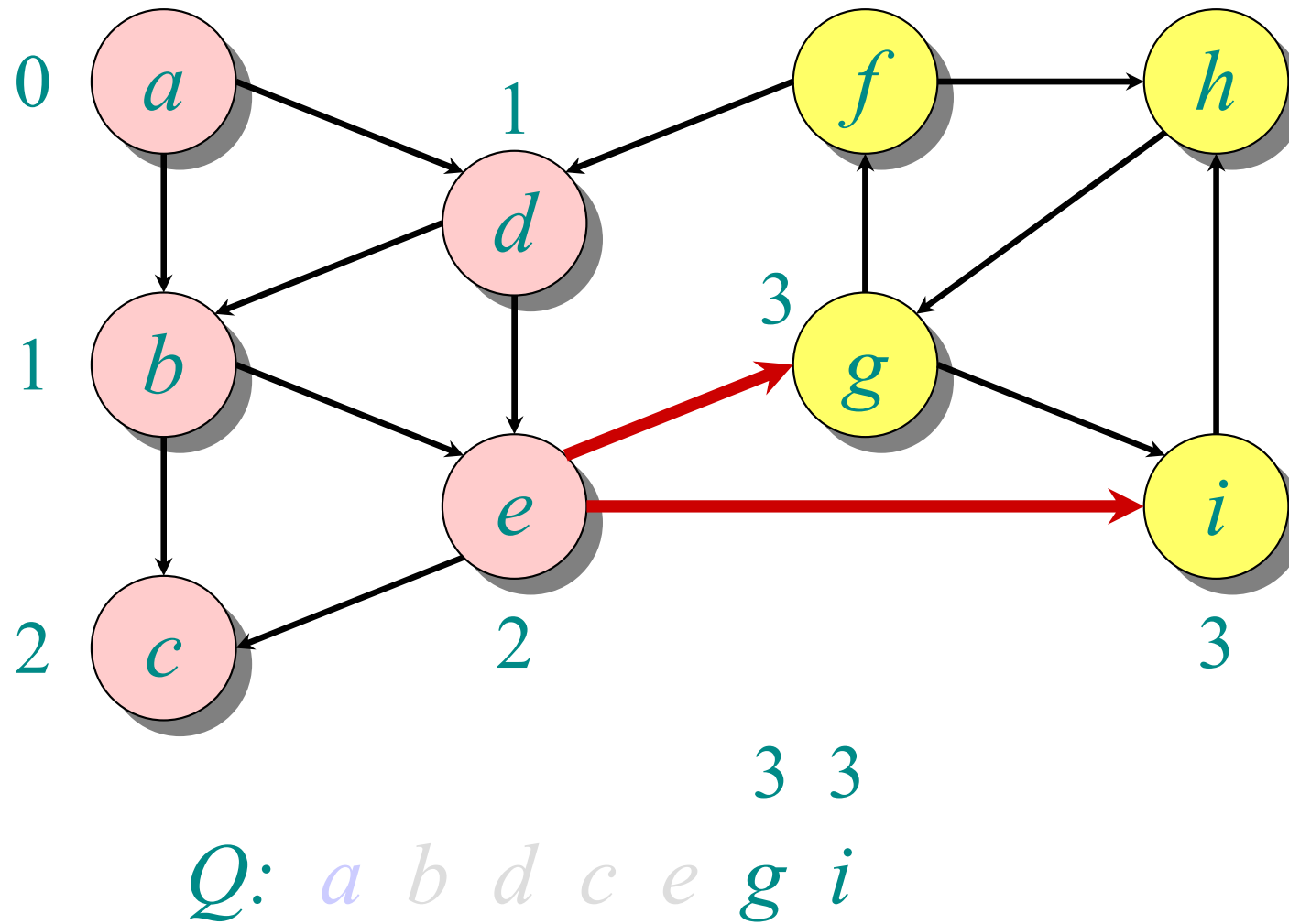
مثال (۶ از ۱۲)



2
 $Q: a \ b \ d \ c \ e$

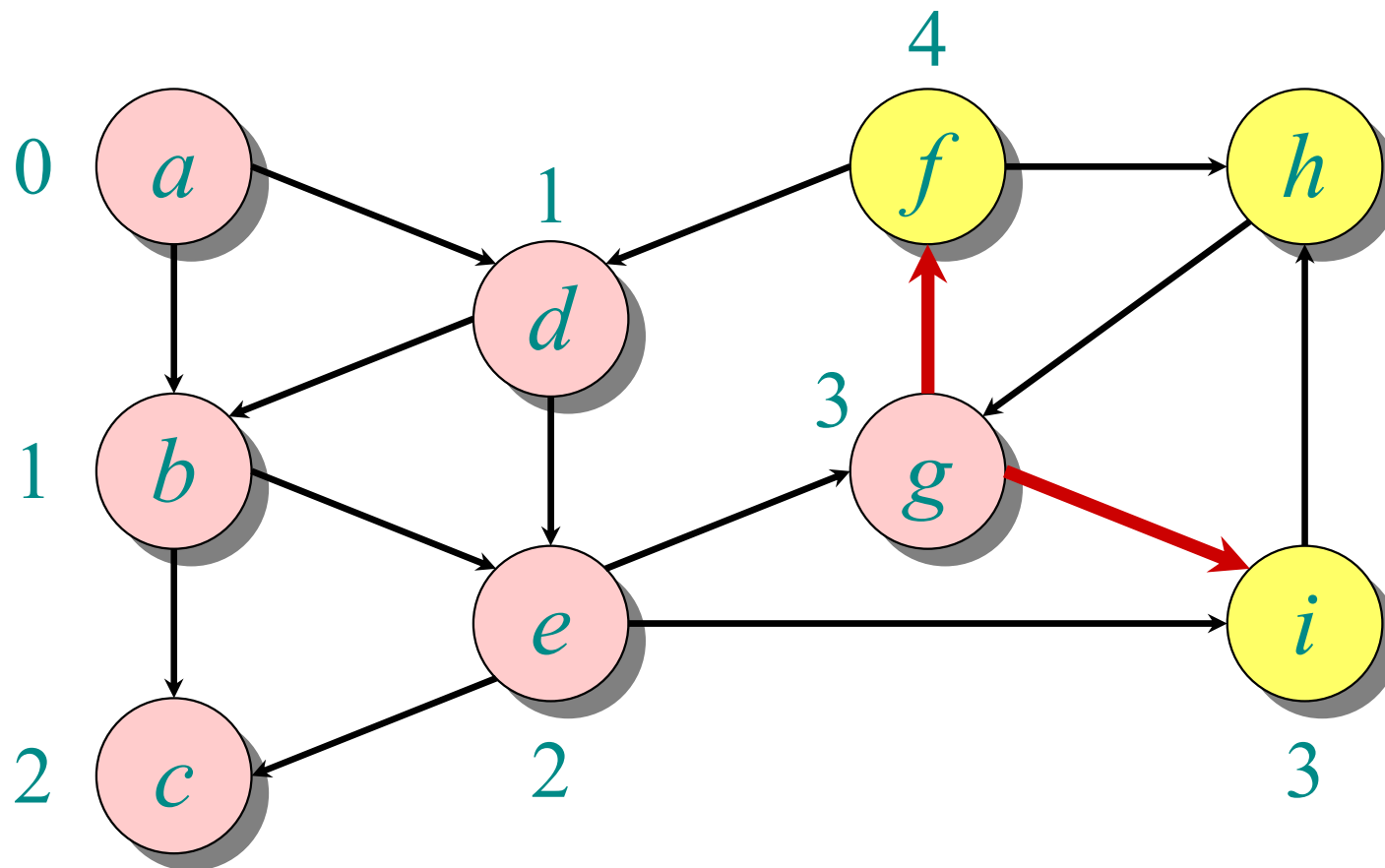
جستجوی عرض-اول

مثال (۷ از ۱۲)



جستجوی عرض-اول

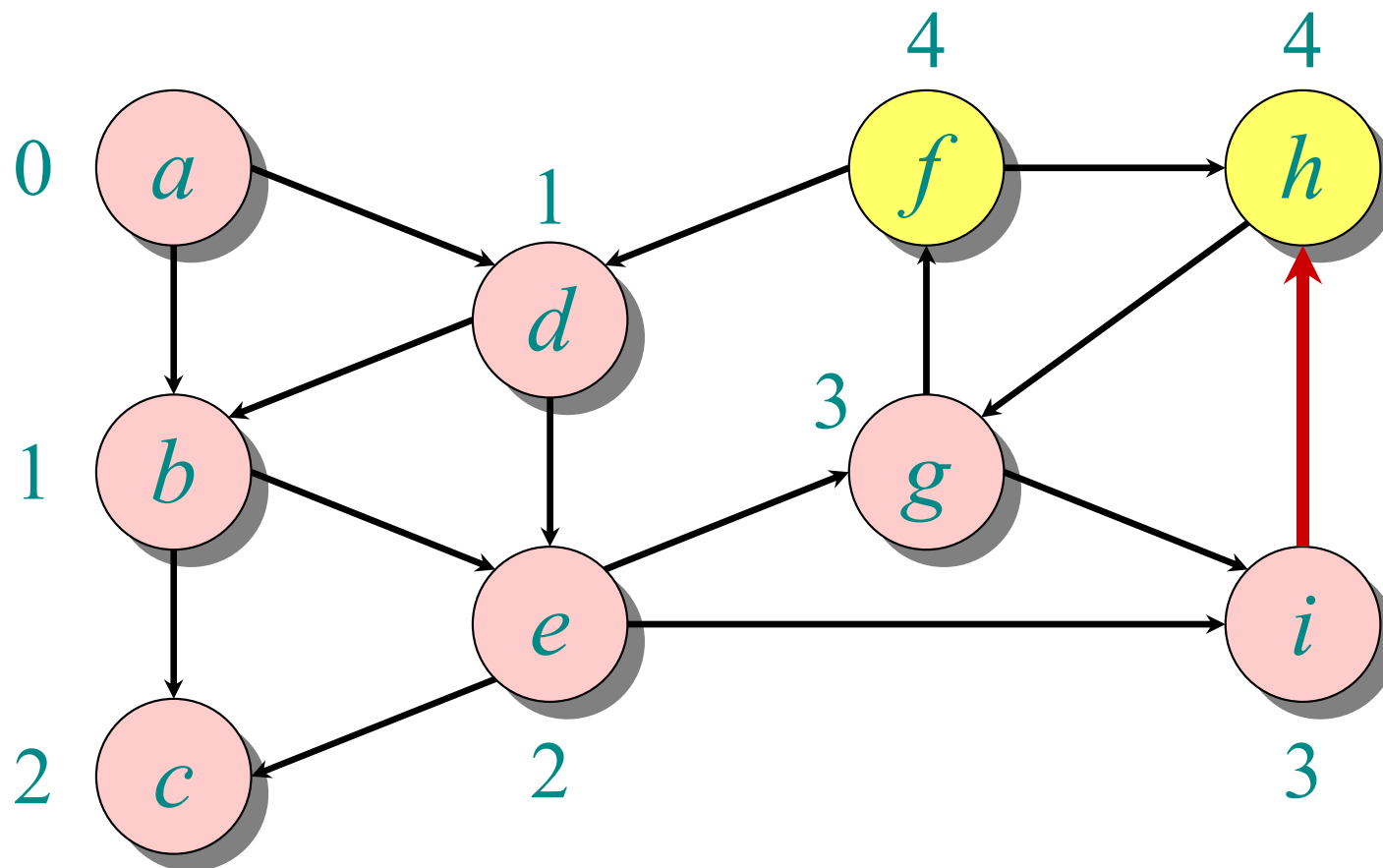
مثال (۸ از ۱۲)



3 4
Q: a b d c e g i f

جستجوی عرض-اول

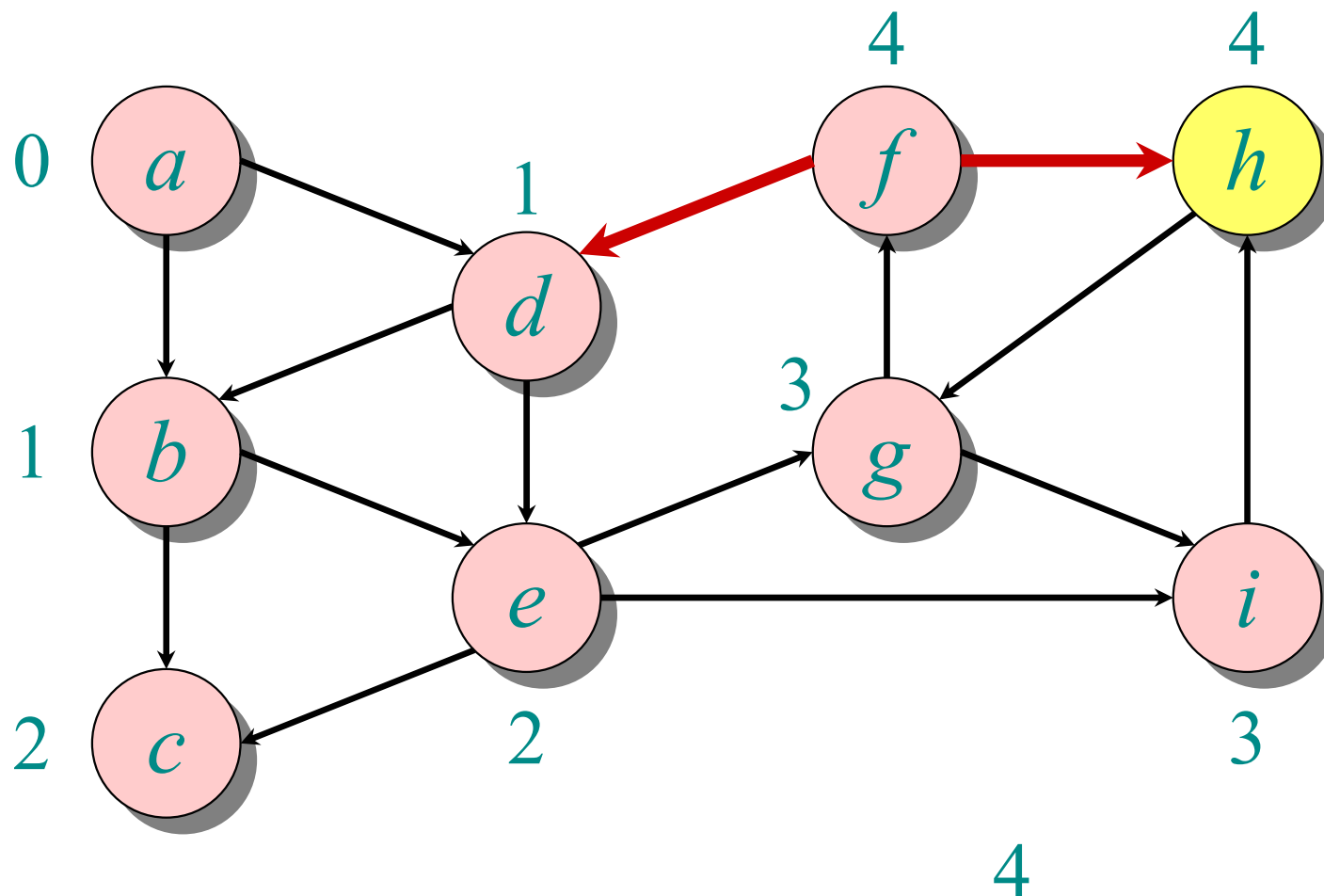
مثال (۹ از ۱۲)



4 4
Q: *a b d c e g i f h*

جستجوی عرض-اول

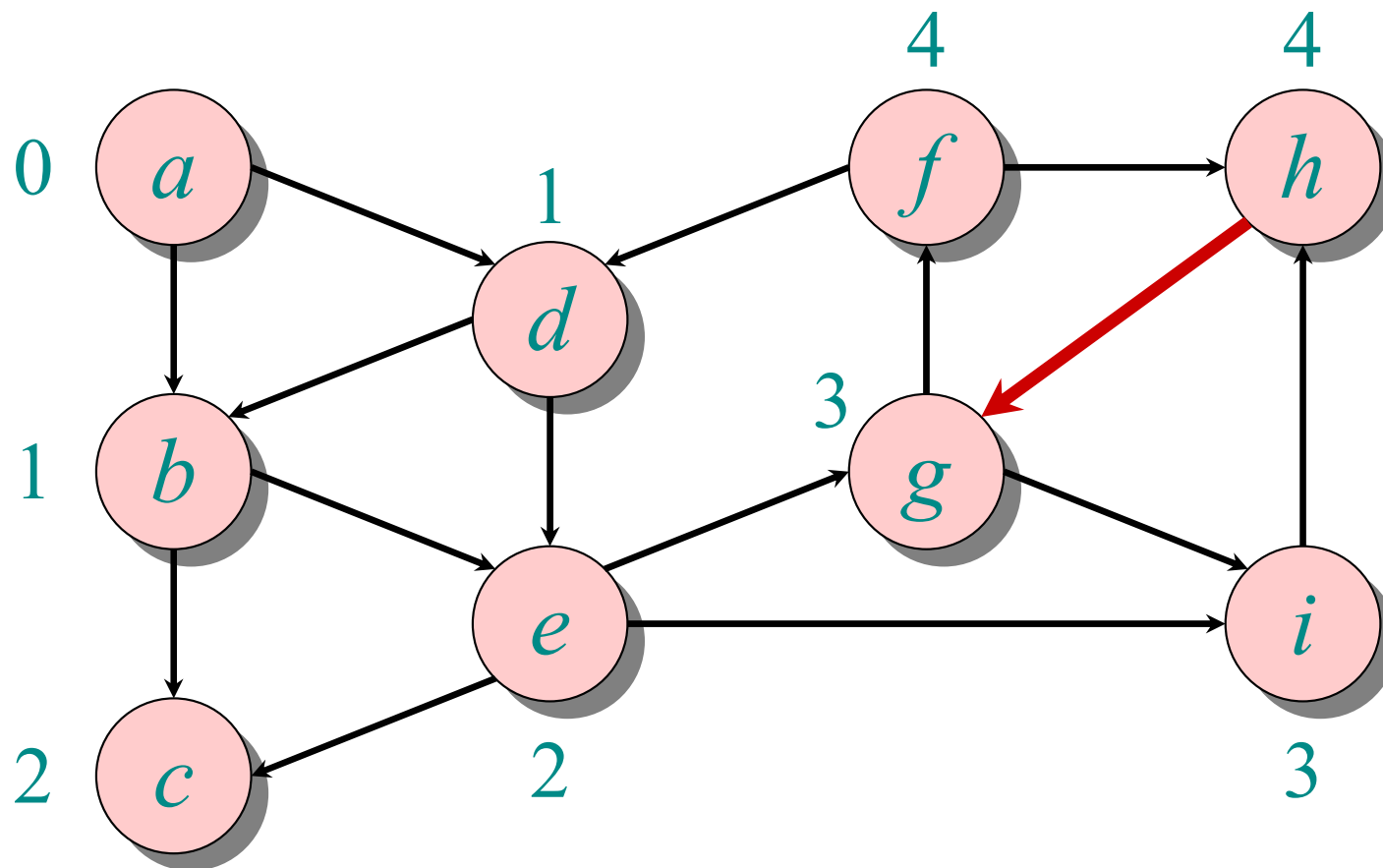
مثال (۱۰ از ۱۲)



$Q: a \ b \ d \ c \ e \ g \ i \ f \ h$

جستجوی عرض-اول

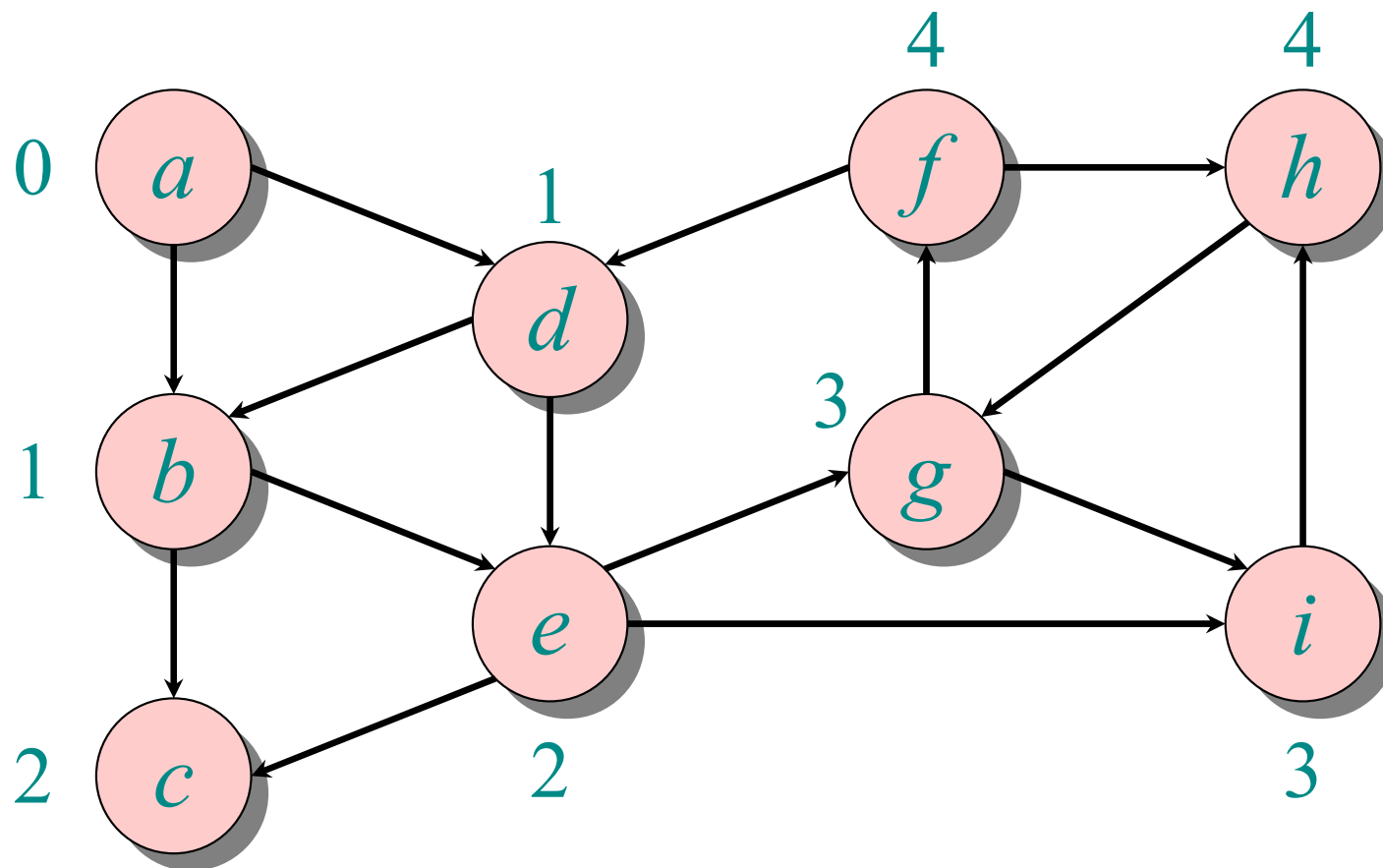
مثال (۱۱ از ۱۲)



$Q:$ *a b d c e g i f h*

جستجوی عرض-اول

مثال (۱۲ از ۱۲)



$Q: a \ b \ d \ c \ e \ g \ i \ f \ h$

جستجوی عرض-اول

اثبات درستی BFS

Correctness of BFS

```

while  $Q \neq \emptyset$ 
  do  $u \leftarrow \text{DEQUEUE}(Q)$ 
    for each  $v \in \text{Adj}[u]$ 
      do if  $d[v] = \infty$ 
        then  $d[v] \leftarrow d[u] + 1$ 
           ENQUEUE( $Q, v$ )

```

Key idea:

The FIFO Q in breadth-first search mimics the priority queue Q in Dijkstra.

- **Invariant:** v comes after u in Q implies that $d[v] = d[u]$ or $d[v] = d[u] + 1$.

جستجوی عرض-اول

شبکه‌کد

BFS(G, s)

```

1  for each vertex  $u \in V[G] - \{s\}$ 
2      do  $color[u] \leftarrow \text{WHITE}$ 
3           $d[u] \leftarrow \infty$ 
4           $\pi[u] \leftarrow \text{NIL}$ 
5   $color[s] \leftarrow \text{GRAY}$ 
6   $d[s] \leftarrow 0$ 
7   $\pi[s] \leftarrow \text{NIL}$ 
8   $Q \leftarrow \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11     do  $u \leftarrow \text{DEQUEUE}(Q)$ 
12         for each  $v \in \text{Adj}[u]$ 
13             do if  $color[v] = \text{WHITE}$ 
14                 then  $color[v] \leftarrow \text{GRAY}$ 
15                      $d[v] \leftarrow d[u] + 1$ 
16                      $\pi[v] \leftarrow u$ 
17                     ENQUEUE( $Q, v$ )
18      $color[u] \leftarrow \text{BLACK}$ 

```

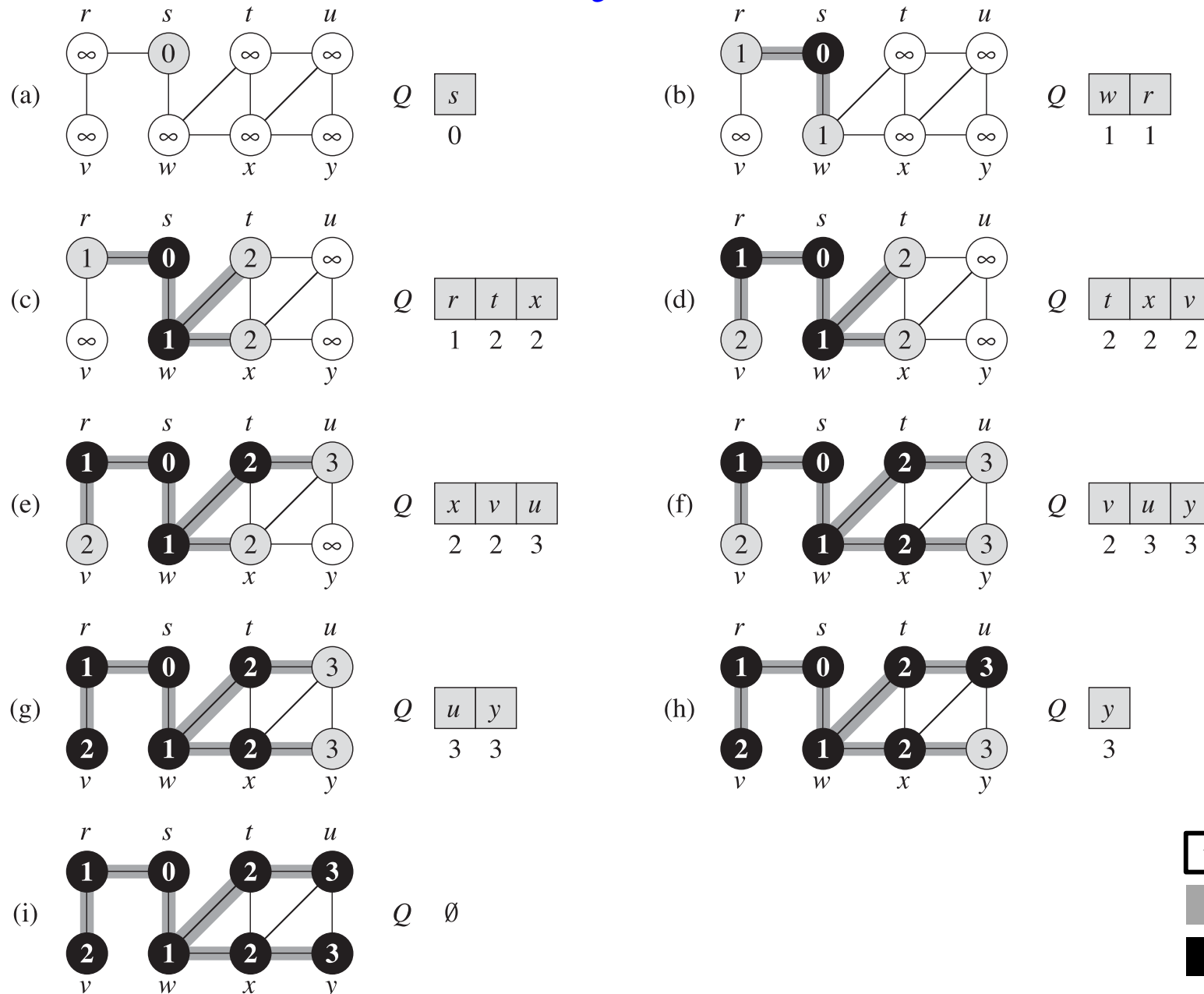
unvisited

in queue

visited

جستجوی عرض-اول

مثال



جستجوی عرض-اول

تحلیل زمان اجرا

BFS(G, s)

```

1  for each vertex  $u \in V[G] - \{s\}$ 
2      do  $color[u] \leftarrow \text{WHITE}$ 
3           $d[u] \leftarrow \infty$ 
4           $\pi[u] \leftarrow \text{NIL}$ 
5   $color[s] \leftarrow \text{GRAY}$ 
6   $d[s] \leftarrow 0$ 
7   $\pi[s] \leftarrow \text{NIL}$ 
8   $Q \leftarrow \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11     do  $u \leftarrow \text{DEQUEUE}(Q)$ 
12         for each  $v \in \text{Adj}[u]$ 
13             do if  $color[v] = \text{WHITE}$ 
14                 then  $color[v] \leftarrow \text{GRAY}$ 
15                      $d[v] \leftarrow d[u] + 1$ 
16                      $\pi[v] \leftarrow u$ 
17                     ENQUEUE( $Q, v$ )
18      $color[u] \leftarrow \text{BLACK}$ 

```

برای هر رأس یک بار

$\Theta(V)$

برای هر یال حداکثر دو بار

$\Theta(E)$

در مجموع

$\Theta(V + E)$

جستجوی عمق-اول

شبکه‌کد

DFS(G)

```

1  for each vertex  $u \in V[G]$ 
2      do  $color[u] \leftarrow \text{WHITE}$ 
3          $\pi[u] \leftarrow \text{NIL}$ 
4   $time \leftarrow 0$ 
5  for each vertex  $u \in V[G]$ 
6      do if  $color[u] = \text{WHITE}$ 
7          then DFS-VISIT( $u$ )

```

DFS-VISIT(u)

```

1   $color[u] \leftarrow \text{GRAY}$        $\triangleright$  White vertex  $u$  has just been discovered.
2   $time \leftarrow time + 1$ 
3   $d[u] \leftarrow time$ 
4  for each  $v \in Adj[u]$        $\triangleright$  Explore edge  $(u, v)$ .
5      do if  $color[v] = \text{WHITE}$ 
6          then  $\pi[v] \leftarrow u$ 
7              DFS-VISIT( $v$ )
8   $color[u] \leftarrow \text{BLACK}$      $\triangleright$  Blacken  $u$ ; it is finished.
9   $f[u] \leftarrow time \leftarrow time + 1$ 

```

unvisited

in queue

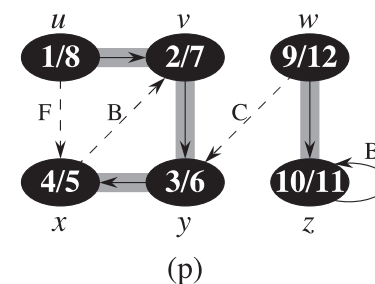
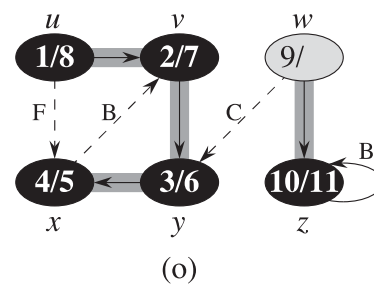
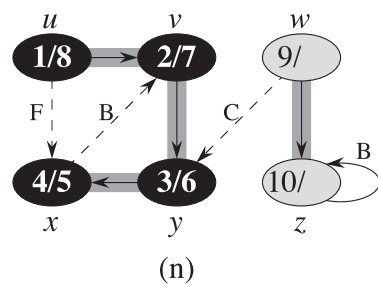
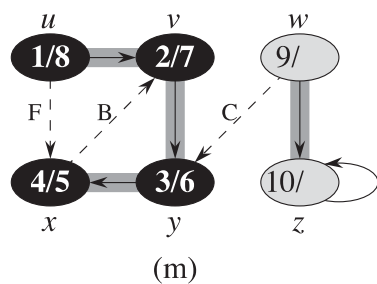
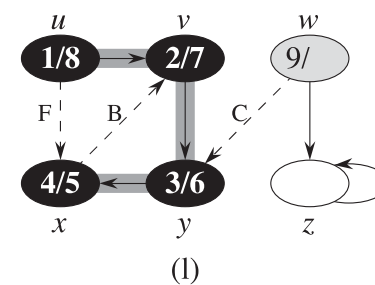
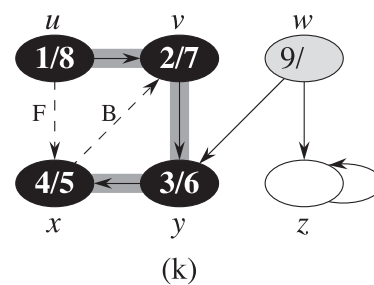
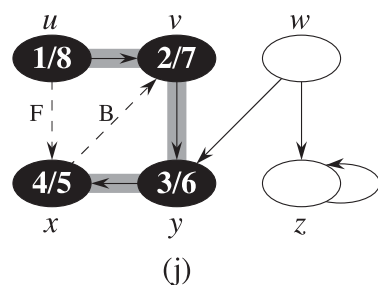
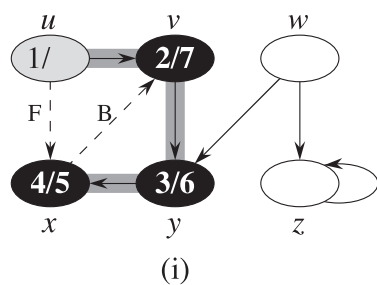
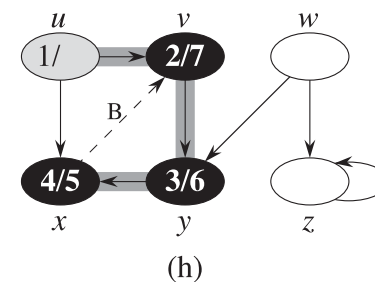
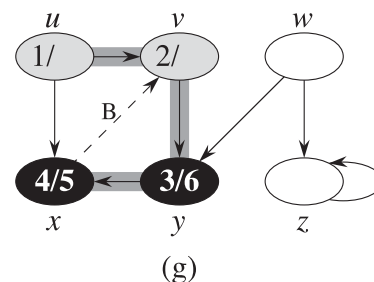
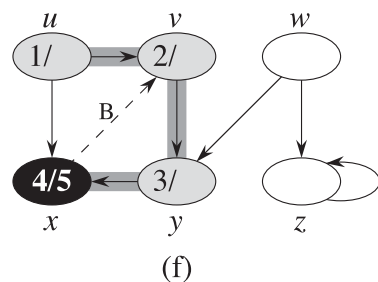
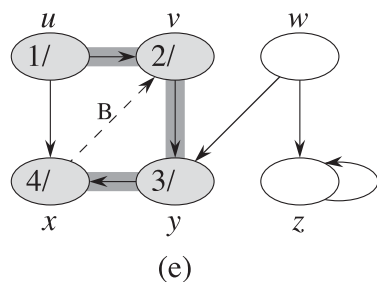
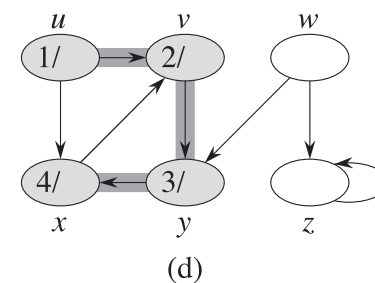
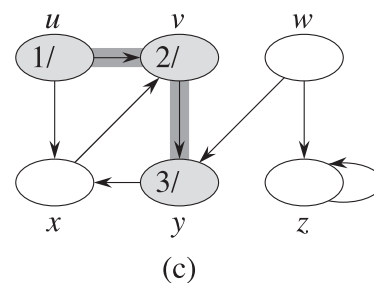
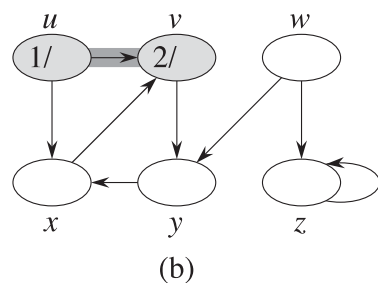
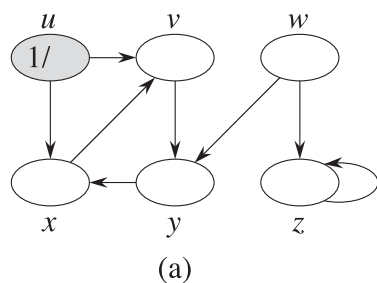
visited

$$1 \leq d[v] \leq f[v] \leq 2n$$

discovery time $d[u]$ finishing time $f[u]$

جستجوی عمق-اول

مثال



جستجوی عمق-اول

تحلیل زمان اجرا

پیاده‌سازی با لیست مجاورت

$$\Theta(V + E)$$

پیاده‌سازی با ماتریس مجاورت

$$\Theta(V^2)$$

مرتب‌سازی توپولوژیکی

TOPOLOGICAL SORT

ارائه‌ی یک ترتیب از رئوس گراف که در آن
هر رأس پس از همه‌ی ماقبل‌هایش ظاهر می‌شود.

مرتب‌سازی توپولوژیکی
Topological Sort

شرط استفاده: گراف باید جهت‌دار بدون دور (DAG: *Directed Acyclic Graph*) باشد.

مرتب‌سازی توپولوژیکی

شبه کد

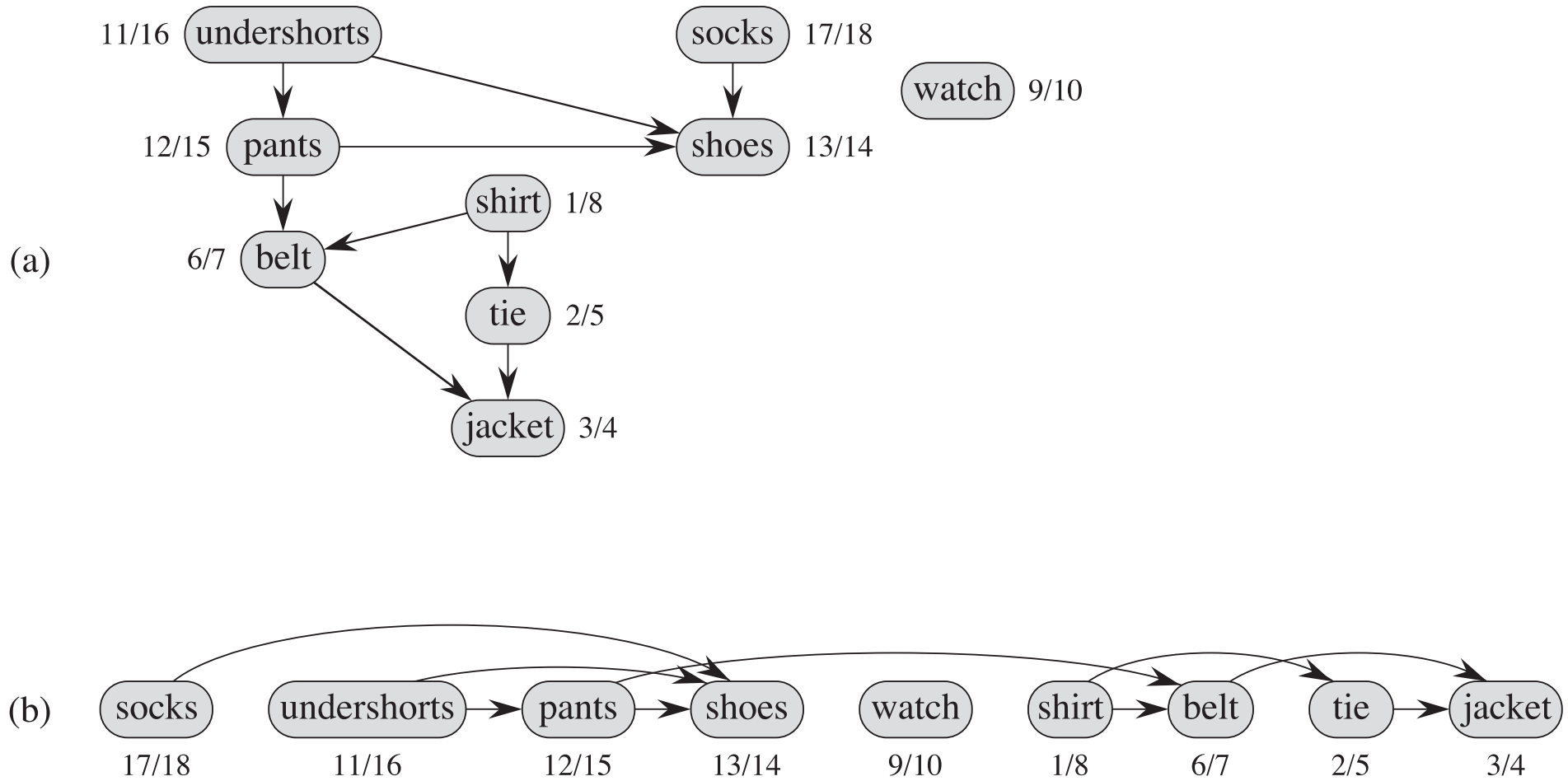
TOPOLOGICAL SORTTOPOLOGICAL-SORT(G)

- 1 call DFS(G) to compute finishing times $f[v]$ for each vertex v
- 2 as each vertex is finished, insert it onto the front of a linked list
- 3 **return** the linked list of vertices

We can perform a topological sort in time $\Theta(V + E)$, since depth-first search takes $\Theta(V + E)$ time and it takes $O(1)$ time to insert each of the $|V|$ vertices onto the front of the linked list.

مرتب‌سازی توپولوژیکی

مثال

TOPOLOGICAL SORT

مؤلفه‌های همبند قوی

STRONGLY CONNECTED COMPONENTS

بزرگ‌ترین زیرمجموعه‌های مجزا از رئوس یک گراف
که در هر مجموعه همگی رئوس از یکدیگر دسترس‌پذیرند.
(هر مجموعه یک مؤلفه‌ی همبند قوی نام دارد)

مؤلفه‌های همبند قوی

Strongly Connected Components

مؤلفه‌های همبند قوی

شبه کد

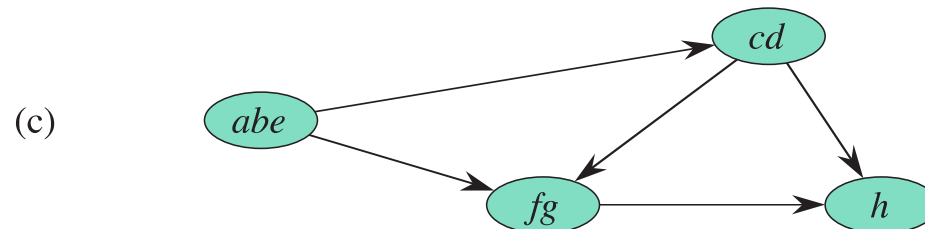
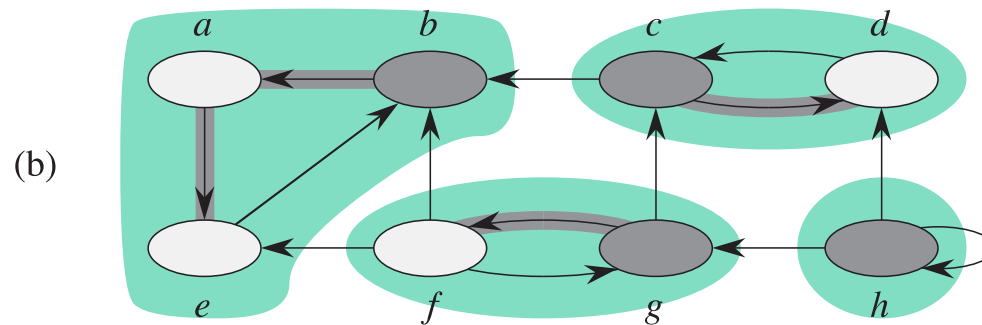
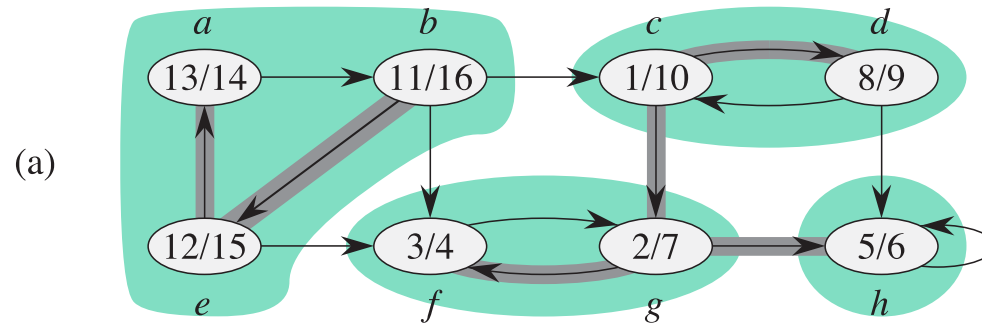
STRONGLY CONNECTED COMPONENTSSTRONGLY-CONNECTED-COMPONENTS (G)

- 1 call DFS(G) to compute finishing times $f[u]$ for each vertex u
- 2 compute G^T
- 3 call DFS(G^T), but in the main loop of DFS, consider the vertices in order of decreasing $f[u]$ (as computed in line 1)
- 4 output the vertices of each tree in the depth-first forest formed in line 3 as a separate strongly connected component

- (۱) با استفاده از DFS(G) زمان پایان $f[u]$ برای هر رأس u را می‌یابیم.
- (۲) با معکوس کردن یال‌های گراف، G^T (گراف ترانهاده) را محاسبه می‌کنیم.
- (۳) DFS(G^T) را فراخوانی می‌کنیم (اما، در حلقه‌ی اصلی رأس‌ها را به ترتیب نزولی $f[u]$ در نظر می‌گیریم).
- (۴) هر یک از درخت‌های پوشای حاصل، یک مؤلفه‌ی همبند قوی را نشان می‌دهد.

مؤلفه‌های همبند قوی

مثال

STRONGLY CONNECTED COMPONENTS

مؤلفه‌های دوهمبند

مثال

BICONNECTED COMPONENTS

بزرگ‌ترین زیرگراف‌های یک گراف که نقطه‌ی مفصلی ندارد.
(هر مجموعه یک مؤلفه‌ی دوهمبند نام دارد)

مؤلفه‌های دوهمبند

Biconnected Components

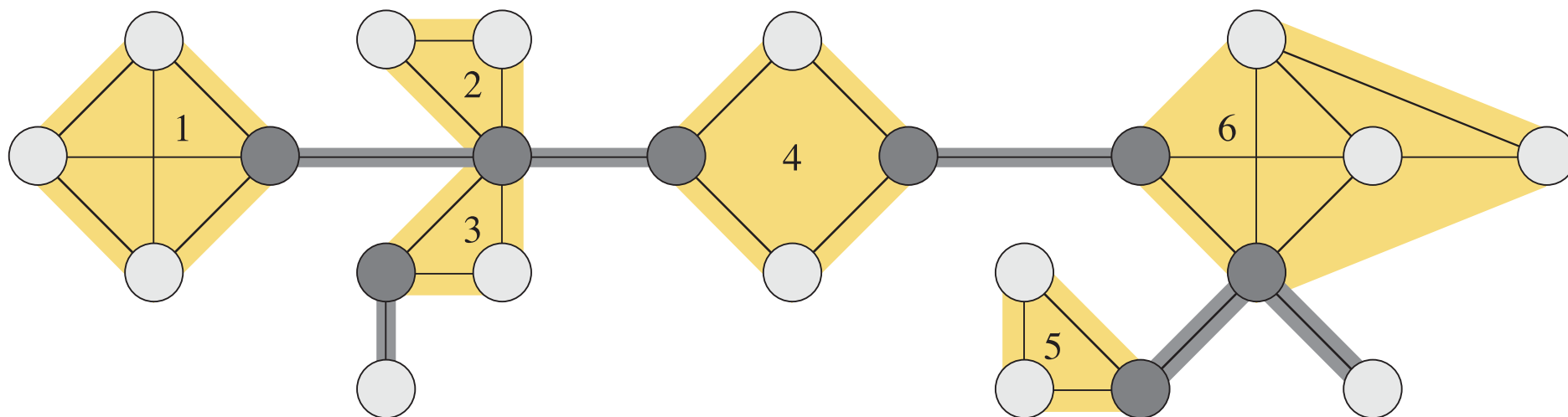
رأسی که حذف آن و یال‌های متصل به آن،
باعث ناهمبند شدن گراف شود.

نقطه‌ی مفصلی

Articulation Point

مؤلفه‌های دوهمبند

مثال

BICONNECTED COMPONENTS

درخت پوشای می‌نیمال

MINIMAL SPANNING TREE

یک زیرگراف درختی پوشا از یک گراف که دارای کمترین وزن ممکن باشد.

درخت پوشای می‌نیمال
Minimal Spanning Tree

الگوریتم پریم	مناسب برای گراف‌های پر
الگوریتم کراسکال	مناسب برای گراف‌های خلوت

کوتاه‌ترین مسیر تک‌منبع

SINGLE-SOURCE-SHORTEST PATH (SSSP)

	الگوریتم دایکسترا
برای گراف‌های دارای وزن منفی	الگوریتم بلمن-فورد

کوتاه‌ترین مسیر میان همه‌ی جفت‌ها

ALL-PAIRS SHORTEST PATH (APSP)

	الگوریتم فلویید وارشال
مناسب برای گراف‌های خلوت	الگوریتم جانسون

رنگ‌آمیزی گراف

GRAPH COLORING

انتساب رنگ به رأس‌های گراف،
به گونه‌ای که هیچ دو رأس مجاوری هم‌رنگ نباشد.

رنگ‌آمیزی گراف
Graph Coloring

آیا گراف مورد نظر با m رنگ قابل رنگ‌آمیزی است؟

رنگ‌آمیزی گراف با m رنگ
Graph m -Coloring

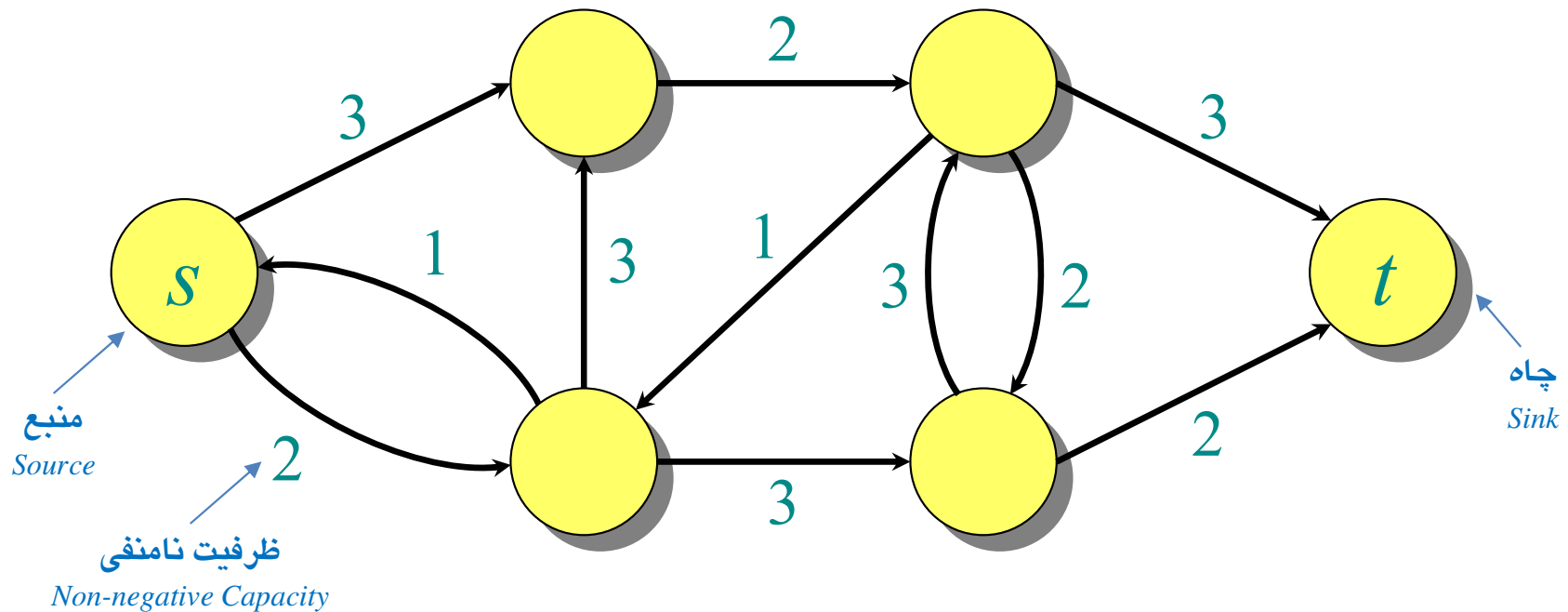
ماکزیمم شار

MAXIMUM FLOW

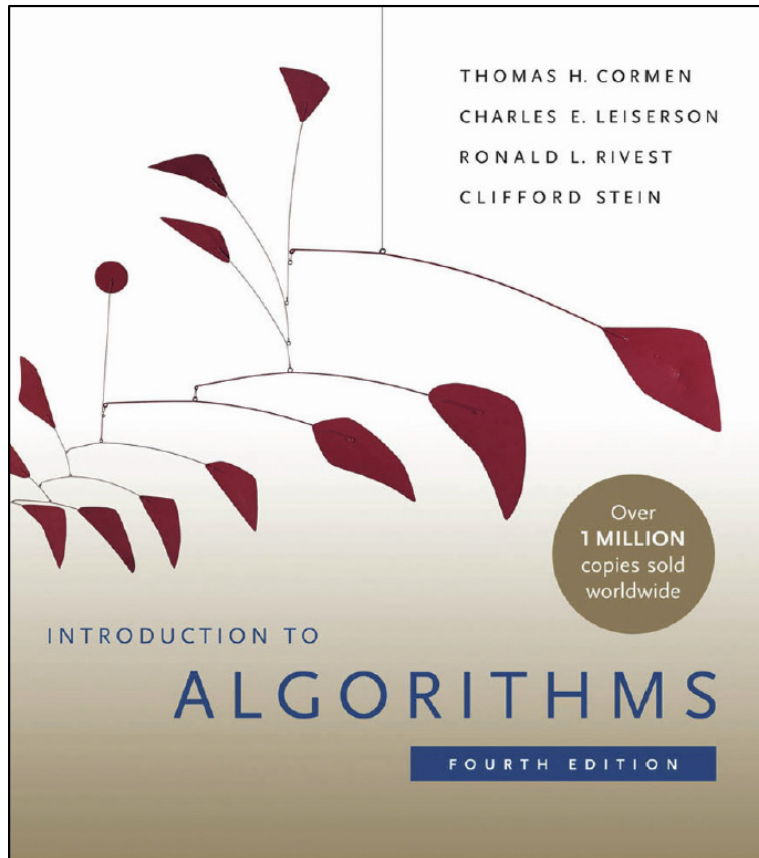
چه میزان سیال می‌توان از منبع s به چاه t جریان پیدا کند؟

فرض می‌کنیم سیال با سرعت یکنواخت از رأس منبع s وارد شبکه می‌شود و از رأس چاه t از شبکه خارج می‌شود.

- یال‌ها: مجرای عبور سیال با ظرفیت مشخص
- رأس‌ها: محل تقاطع مجراها (با فرض اینکه هیچ سیالی در رأس‌ها ذخیره نمی‌شود).



مرجع



T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein,
Introduction to Algorithms,
 4th Edition, MIT Press, 2022.

Chapter 20

20 Elementary Graph Algorithms

This chapter presents methods for representing a graph and for searching a graph. Searching a graph means systematically following the edges of the graph so as to visit the vertices of the graph. A graph-searching algorithm can discover much about the structure of a graph. Many algorithms begin by searching their input graph to obtain this structural information. Several other graph algorithms elaborate on basic graph searching. Techniques for searching a graph lie at the heart of the field of graph algorithms.

Section 20.1 discusses the two most common computational representations of graphs: as adjacency lists and as adjacency matrices. Section 20.2 presents a simple graph-searching algorithm called breadth-first search and shows how to create a breadth-first tree. Section 20.3 presents depth-first search and proves some standard results about the order in which depth-first search visits vertices. Section 20.4 provides our first real application of depth-first search: topologically sorting a directed acyclic graph. A second application of depth-first search, finding the strongly connected components of a directed graph, is the topic of Section 20.5.

20.1 Representations of graphs

You can choose between two standard ways to represent a graph $G = (V, E)$: as a collection of adjacency lists or as an adjacency matrix. Either way applies to both directed and undirected graphs. Because the adjacency-list representation provides a compact way to represent *sparse* graphs—those for which $|E|$ is much less than $|V|^2$ —it is usually the method of choice. Most of the graph algorithms presented in this book assume that an input graph is represented in adjacency-list form. You might prefer an adjacency-matrix representation, however, when the graph is *dense*— $|E|$ is close to $|V|^2$ —or when you need to be able to tell quickly whether there is an edge connecting two given vertices. For example, two of the