طراحی و تحلیل الگوریتم‌ها

مبحث دهم

مطالعه‌ی موضوعی الگوریتم‌ها

الگوریتم‌های گراف

Graph Algorithms

کاظم فولادی
دانشکده مهندسی برق و کامپیوتر
دانشگاه تهران

http://courses.fouladi.ir/algorithm

گراف

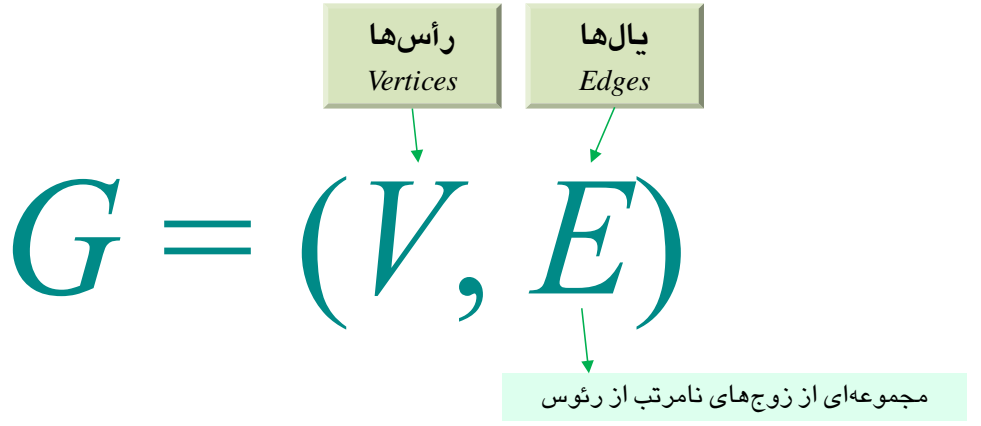
کاربردها

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

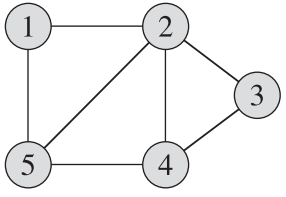
UNDIRECTED GRAPH



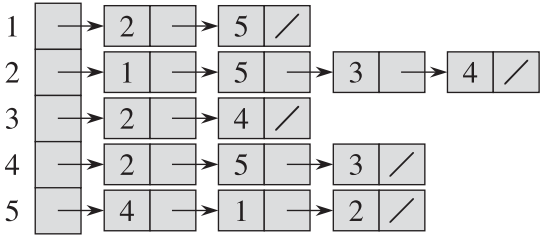
بازنمایی گراف

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

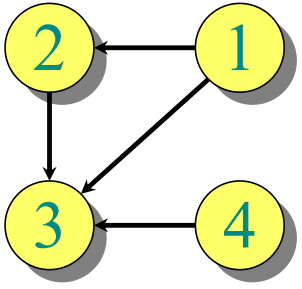
رأس‌ها
Vertices

یال‌ها
Edges

$$G = (V, E)$$

مجموعه‌ای از زوج‌های مرتب از رئوس

$$E \subseteq V \times V$$



بازنمایی گراف

گراف جهت‌دار

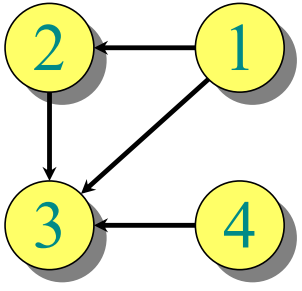
GRAPH REPRESENTATION

$$G = (V, E)$$

$$V = \{1, 2, \ldots, n\} \qquad E \subseteq V \times V$$

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

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



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

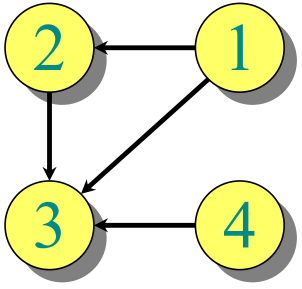
گراف جهت‌دار

GRAPH REPRESENTATION

$$G = (V, E)$$

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

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



لیست مجاورت
Adjacency List

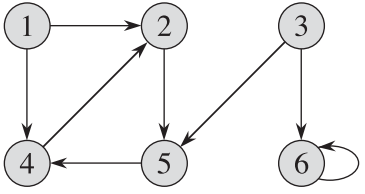
For undirected graphs, $|Adj[v]| = degree(v)$.
For digraphs, $|Adj[v]| = out\text{-}degree(v)$.

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

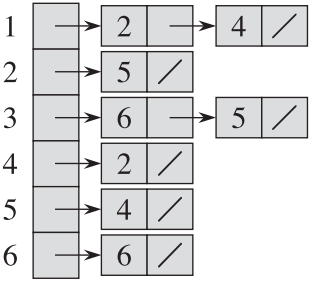
بازنمایی گراف

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

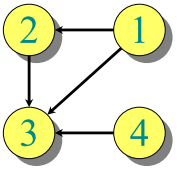
در گراف بدون جهت

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

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

مسیر *Path*	یک دنباله از یال‌های مجاور متمایز در یک گراف

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

$$u \rightsquigarrow v$$

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

مسیر ساده

SIMPLE PATH

مسیر ساده *Simple Path*	یک مسیر بدون رأس تکراری

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

$$u \rightsquigarrow v$$

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

همبندی

CONNECTIVITY

گراف همبند *Connected Graph*	گراف بدون جهت	گرافی که بین هر دو رأس دلخواه آن مسیری وجود داشته باشد.
	گراف جهت‌دار	گرافی که گراف بدون جهت متناظر با آن (پس از حذف جهت‌ها) همبند باشد.
گراف همبند قوی *Strongly Connected Graph*	گراف جهت‌دار	گرافی که بین هر دو رأس دلخواه آن مسیری جهت‌دار وجود داشته باشد.
گراف دوهمبند *Biconnected Graph*	گراف بدون جهت	گراف همبندی که پس از حذف یک یال دلخواه از آن همچنان همبند بماند.

در گراف همبند:

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

دور (چرخه)

CYCLE

دور *Cycle*	یک مسیر که رئوس ابتدا و انتهای آن یکسان است.

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

$$u \rightsquigarrow u$$

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 (به‌جز رئوس ابتدا و انتهای آن).

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

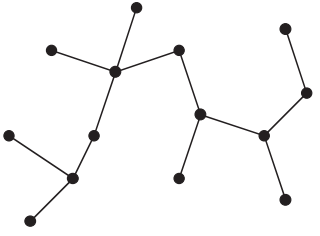
$$u \rightsquigarrow u$$

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

درخت

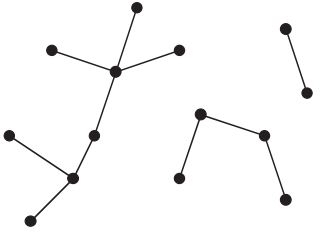
TREE

درخت *Tree*	گراف همبند بدون دور



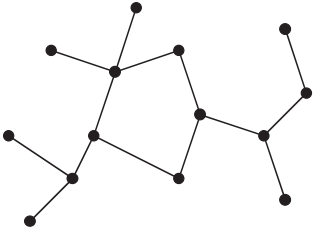
(a)

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

درخت *Tree*	گراف همبند بدون دور

Theorem B.2 (Properties of free trees)
Let $G = (V, E)$ be an undirected graph. The following statements are equivalent.

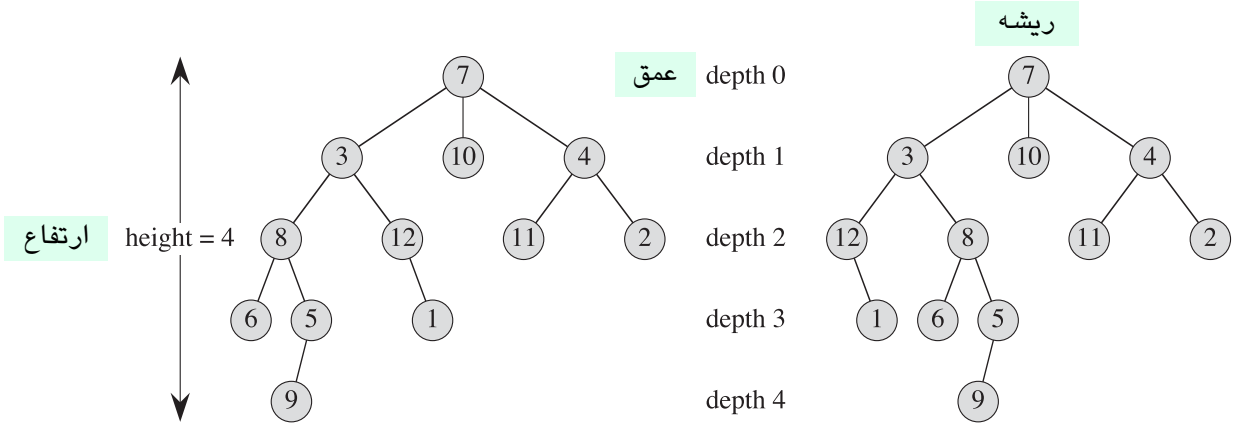
1. G is a free tree.
2. Any two vertices in G are connected by a unique simple path.
3. G is connected, but if any edge is removed from E, the resulting graph is dis-connected.
4. G is connected, and $|E| = |V| - 1$.
5. G is acyclic, and $|E| = |V| - 1$.
6. G is acyclic, but if any edge is added to E,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

درخت ریشه‌دار مرتب

ORDERED ROOTED TREE

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.

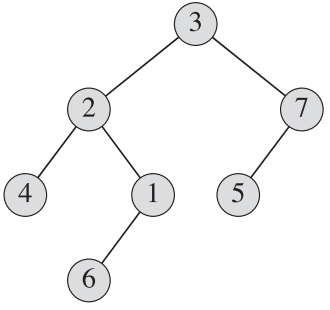


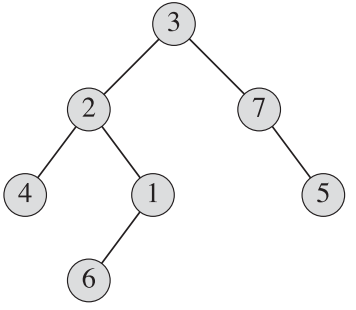
درخت دودویی

BINARY TREE

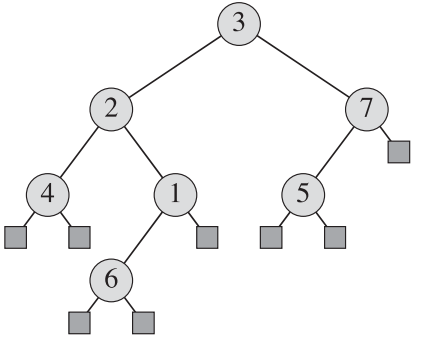
درخت دودویی
Binary Tree

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





برگ

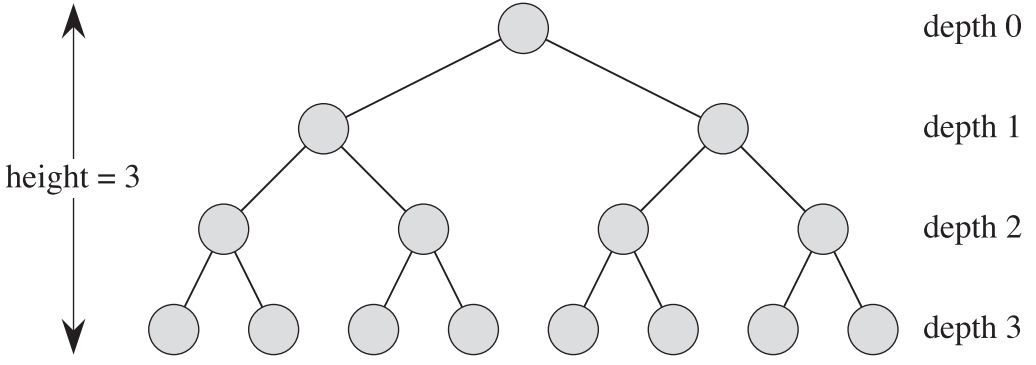


درخت دودویی کامل

COMPLETE BINARY TREE

درخت دودویی کامل
Complete Binary Tree

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



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

درخت k-تایی

مثال

K-ARY TREE

درخت k-تایی
k-ary Tree

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

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

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

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؟

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

تکنیک‌های پیمایش گراف	
جستجوی عرض–اول *Breadth-First Search (BFS)*	**جستجوی عمق–اول** *Depth-First Search (DFS)*
فرزندان یک گره به ترتیب **فاصله از گره‌ی شروع** پیمایش می‌شوند.	فرزندان یک گره **به‌صورت بازگشتی پیش از همزادهای آن** پیمایش می‌شوند.

جستجوی عرض-اول

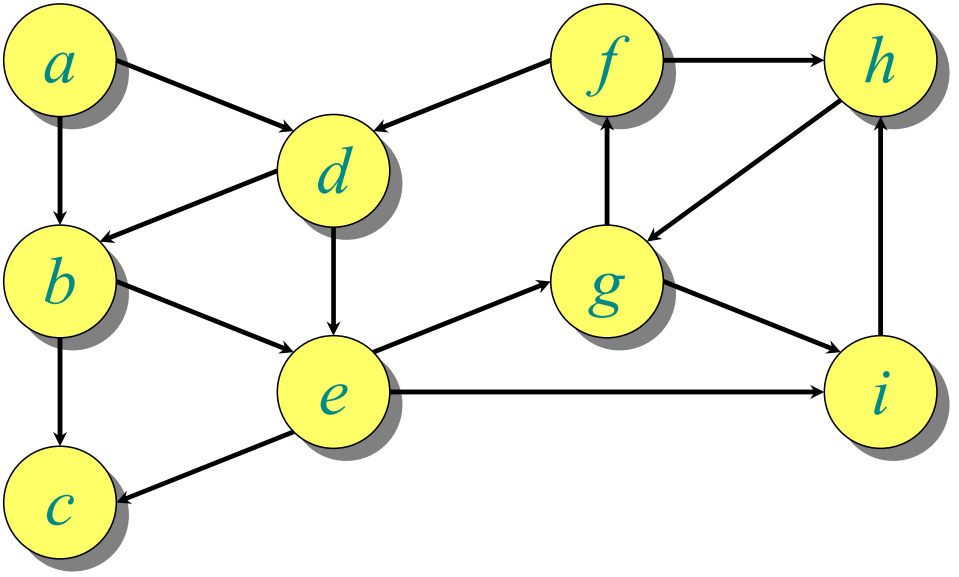
Breadth-first search

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

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

جستجوی عرض-اول

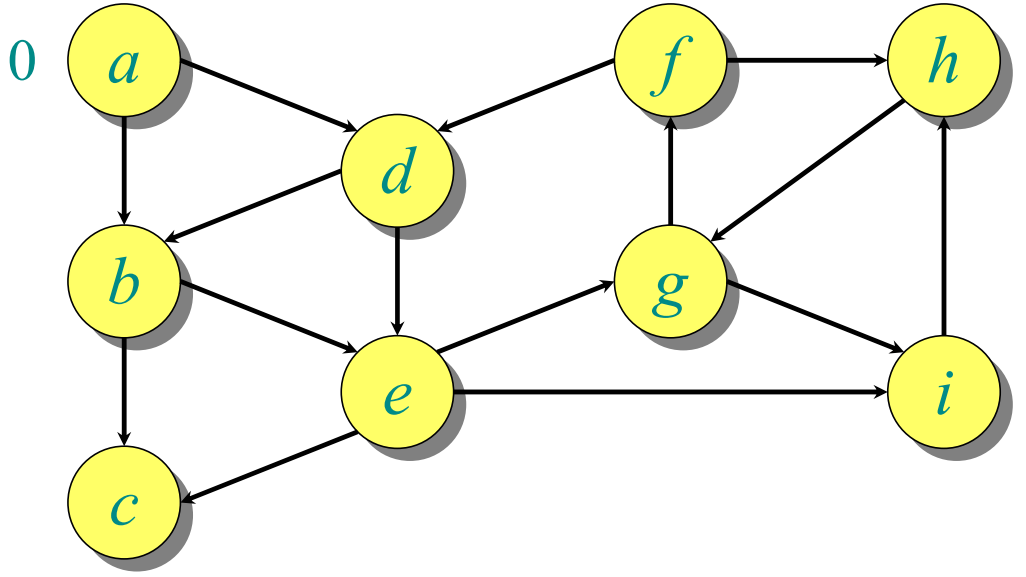
مثال (۱ از ۱۲)



Q:

جستجوی عرض-اول

مثال (۲ از ۱۲)

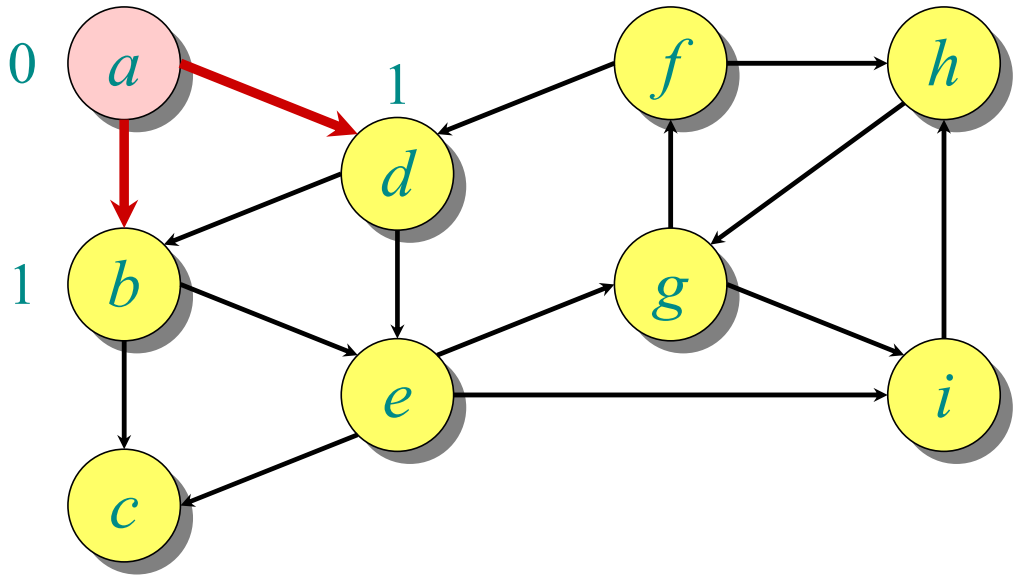


0

Q: a

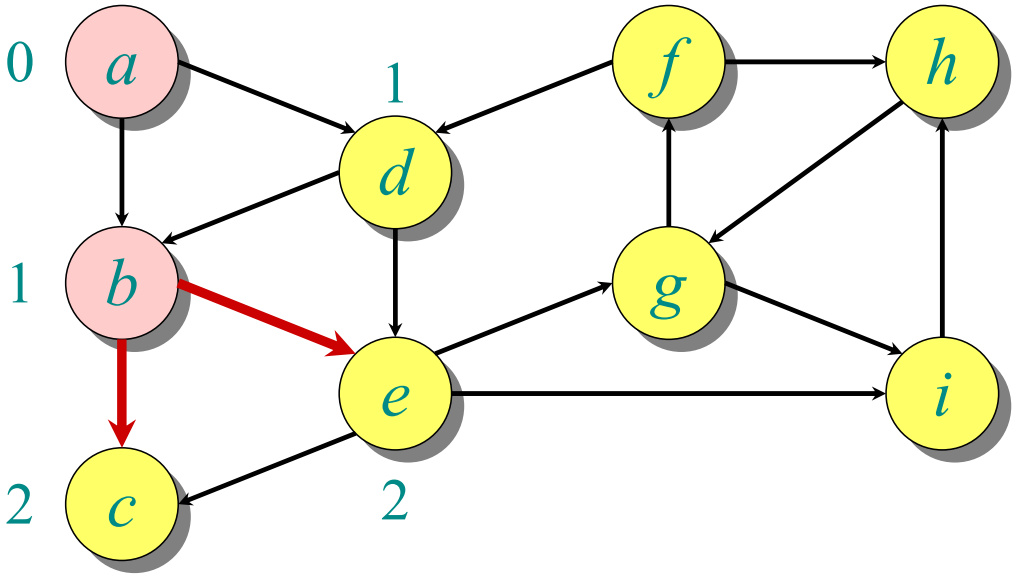
جستجوی عرض-اول

مثال (۳ از ۱۲)



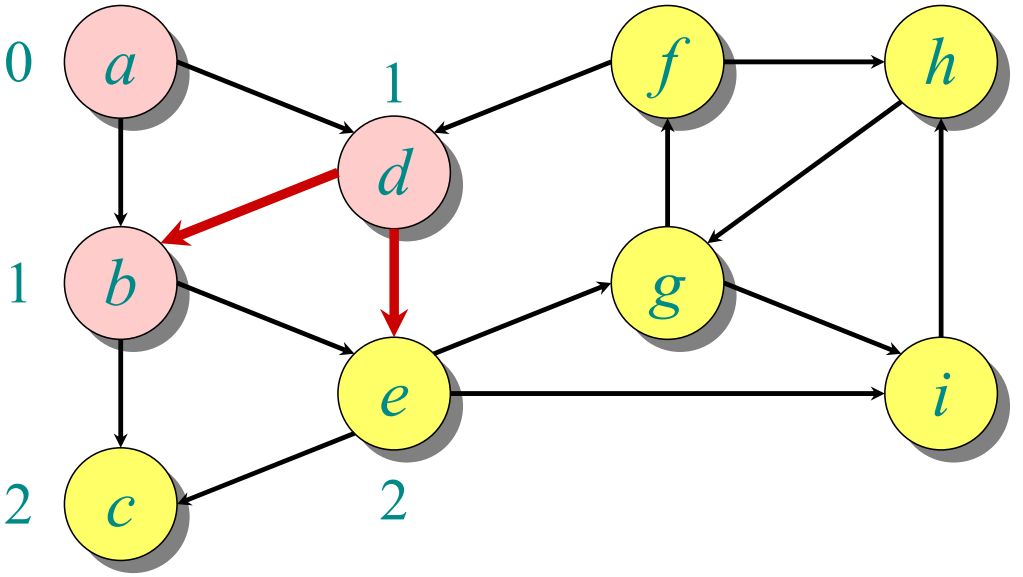
جستجوی عرض-اول

مثال (۴ از ۱۲)



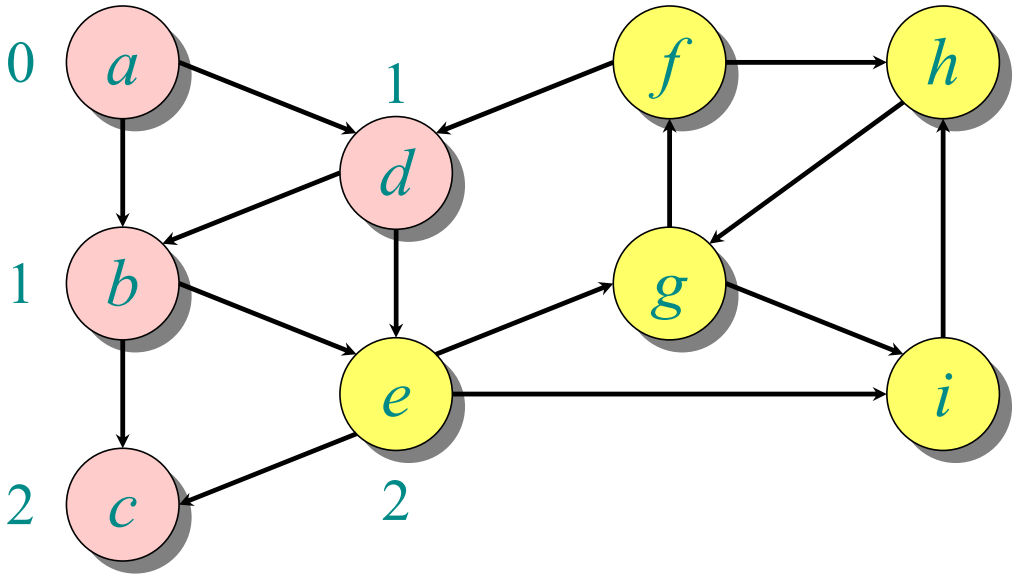
جستجوی عرض-اول

مثال (۵ از ۱۲)



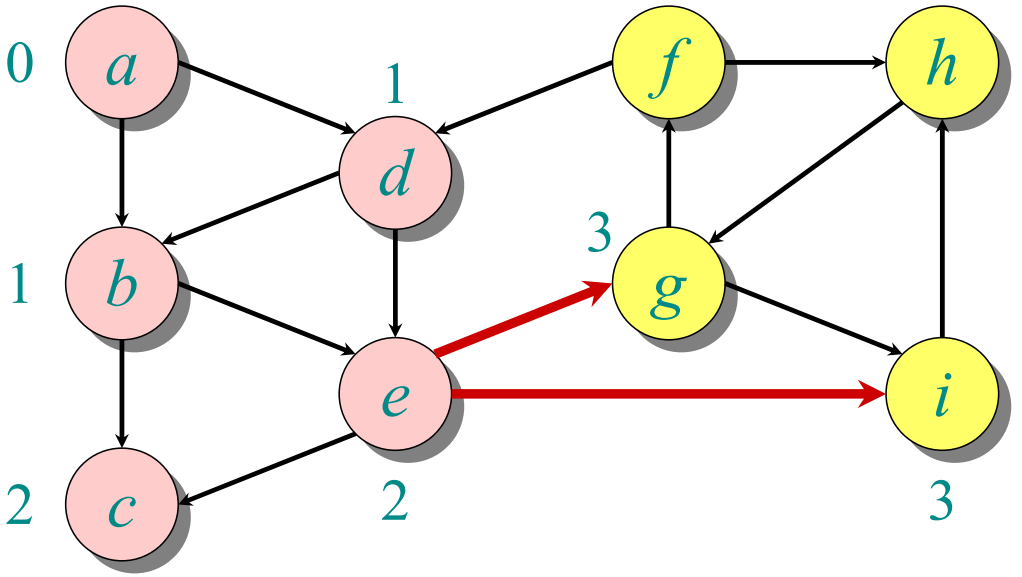
جستجوی عرض–اول

مثال (۶ از ۱۲)



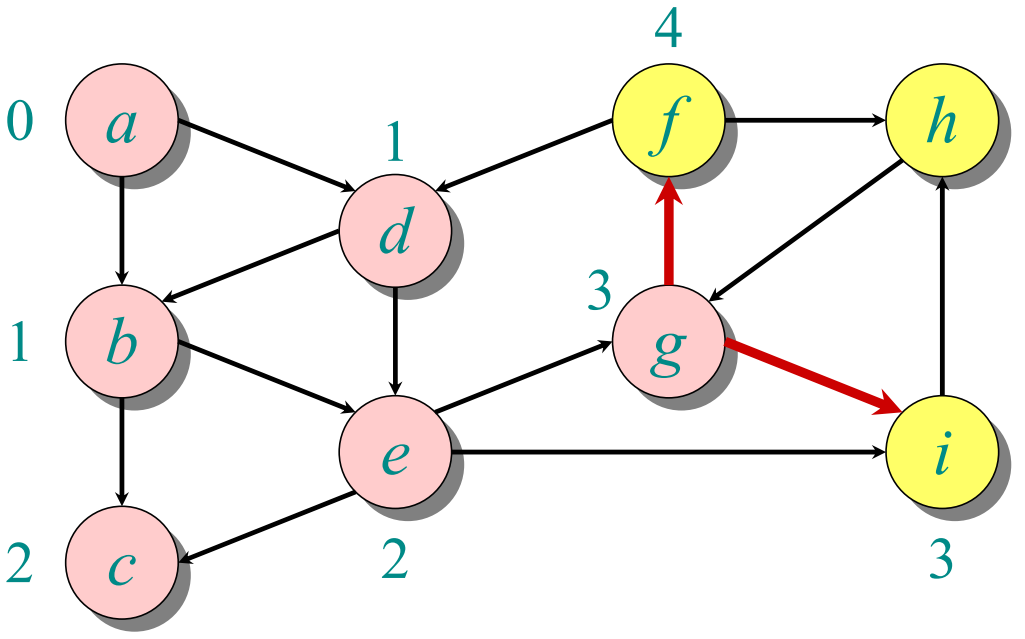
جستجوی عرض-اول

مثال (۷ از ۱۲)



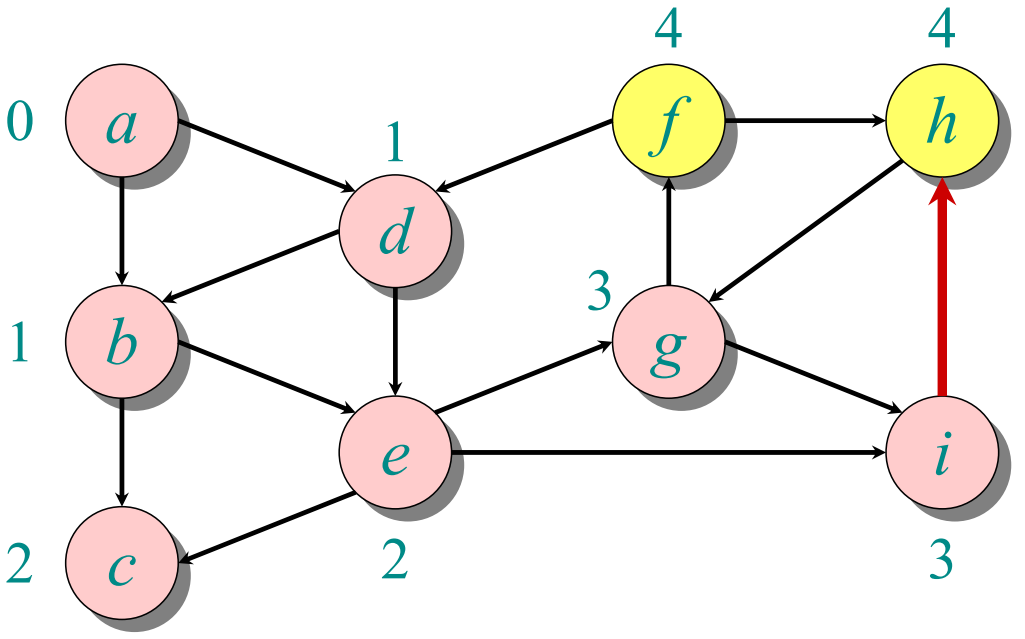
جستجوی عرض–اول

مثال (۸ از ۱۲)



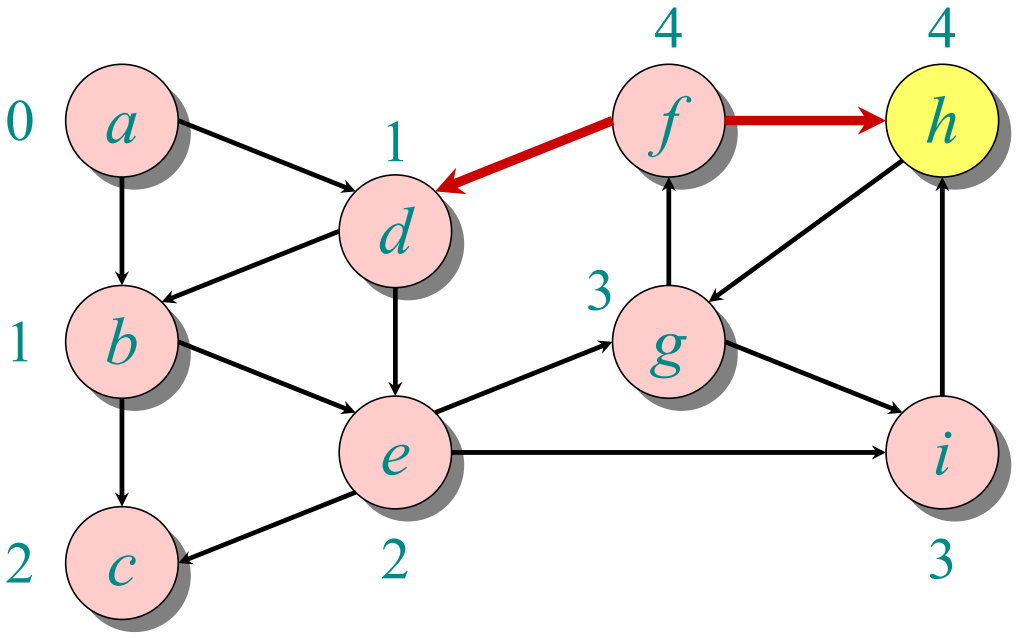
جستجوی عرض-اول

مثال (۹ از ۱۲)



جستجوی عرض–اول

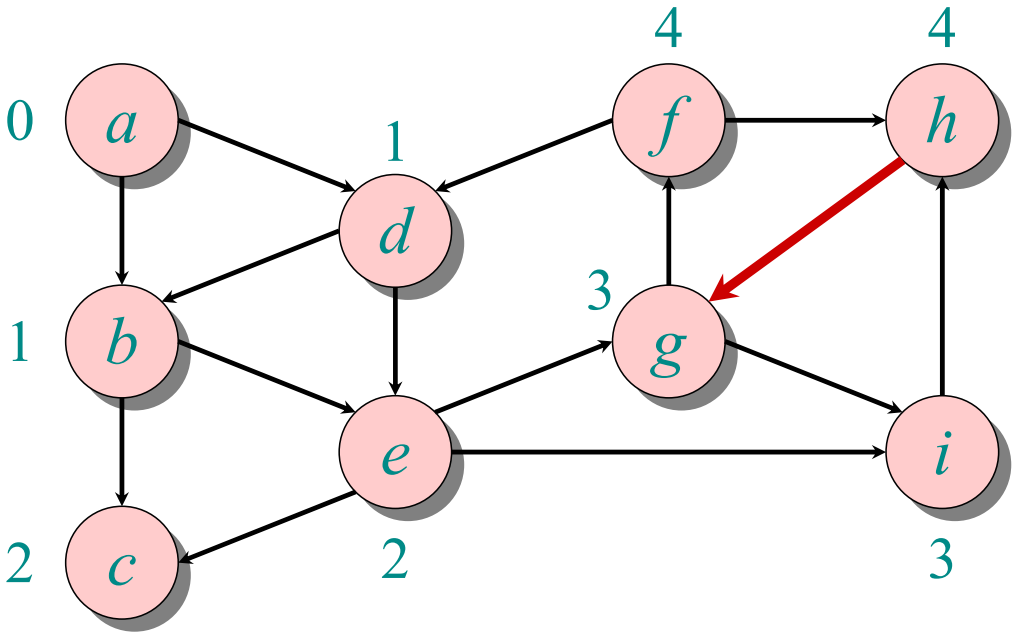
مثال (۱۰ از ۱۲)



Q: a b d c e g i f h

جستجوی عرض-اول

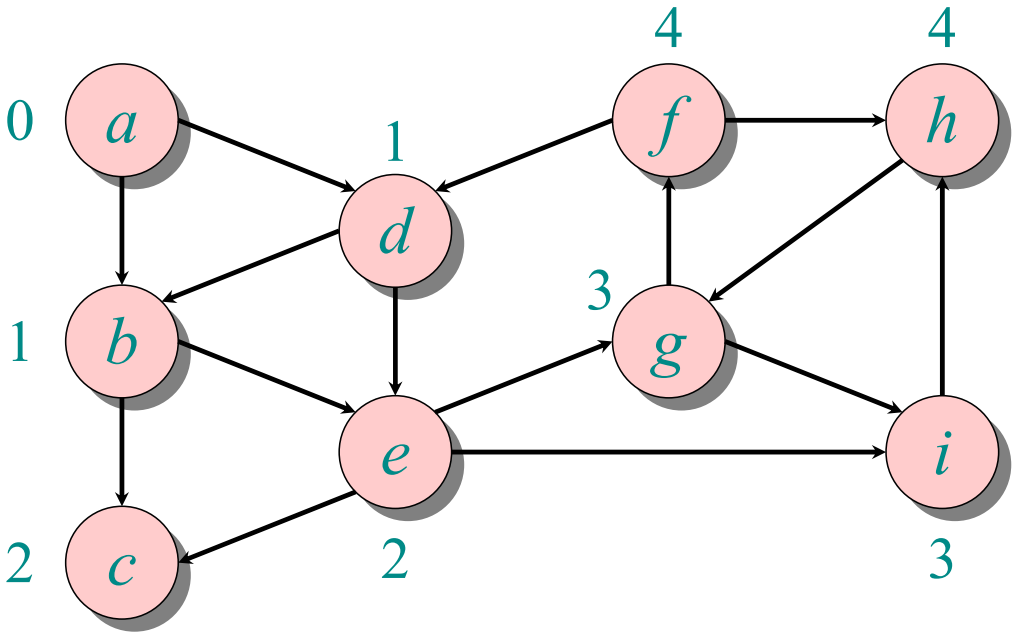
مثال (۱۱ از ۱۲)



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

Correctness of BFS

```
while Q ≠ ∅
    do u ← DEQUEUE(Q)
       for each v ∈ Adj[u]
           do if d[v] = ∞
                 then d[v] ← 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

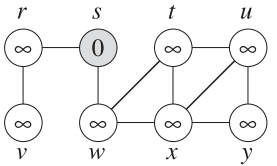
شبه‌کد

```
BFS(G, s)
 1  for each vertex u ∈ V[G] − {s}
 2       do color[u] ← WHITE
 3          d[u] ← ∞
 4          π[u] ← NIL
 5  color[s] ← GRAY
 6  d[s] ← 0
 7  π[s] ← NIL
 8  Q ← ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11       do u ← DEQUEUE(Q)
12          for each v ∈ Adj[u]
13               do if color[v] = WHITE
14                     then color[v] ← GRAY
15                          d[v] ← d[u] + 1
16                          π[v] ← u
17                          ENQUEUE(Q, v)
18          color[u] ← BLACK
```

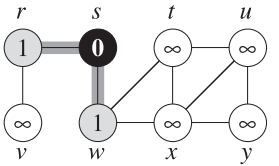
unvisited
in queue
visited

جستجوی عرض–اول

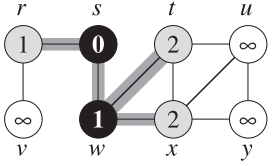
مثال

(a)


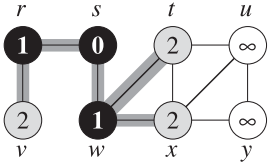
Q | s | | |
0

(b)


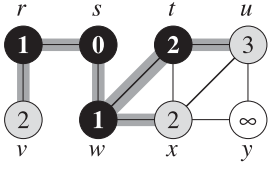
Q | w | r | |
1 1

(c)


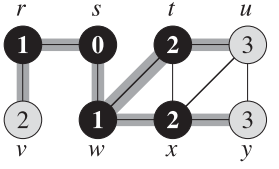
Q | r | t | x |
1 2 2

(d)


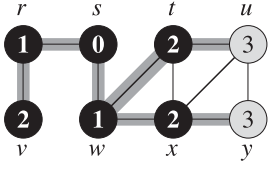
Q | t | x | v |
2 2 2

(e)


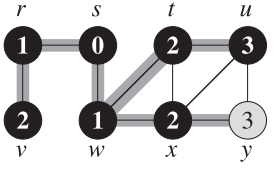
Q | x | v | u |
2 2 3

(f)


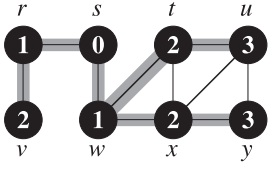
Q | v | u | y |
2 3 3

(g)


Q | u | y | |
3 3

(h)


Q | y | | |
3

(i)


Q $\emptyset$

unvisited
in queue
visited

جستجوی عرض–اول

تحلیل زمان اجرا

```
BFS(G, s)
 1  for each vertex u ∈ V[G] − {s}
 2       do color[u] ← WHITE
 3          d[u] ← ∞
 4          π[u] ← NIL
 5  color[s] ← GRAY
 6  d[s] ← 0
 7  π[s] ← NIL
 8  Q ← ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11       do u ← DEQUEUE(Q)
12          for each v ∈ Adj[u]
13               do if color[v] = WHITE
14                     then color[v] ← GRAY
15                          d[v] ← d[u] + 1
16                          π[v] ← u
17                          ENQUEUE(Q, v)
18          color[u] ← BLACK
```

برای هر رأس یک بار

$\Theta(V)$

برای هر یال حداکثر دو بار

$\Theta(E)$

در مجموع

$\Theta(V + E)$

جستجوی عمق-اول

شبه‌کد

```
DFS(G)
1  for each vertex u ∈ V[G]
2       do color[u] ← WHITE
3          π[u] ← NIL
4  time ← 0
5  for each vertex u ∈ V[G]
6       do if color[u] = WHITE
7             then DFS-VISIT(u)

DFS-VISIT(u)
1  color[u] ← GRAY          ▷ White vertex u has just been discovered.
2  time ← time + 1
3  d[u] ← time
4  for each v ∈ Adj[u]      ▷ Explore edge (u, v).
5       do if color[v] = WHITE
6             then π[v] ← u
7                  DFS-VISIT(v)
8  color[u] ← BLACK         ▷ Blacken u; it is finished.
9  f[u] ← time ← time + 1
```

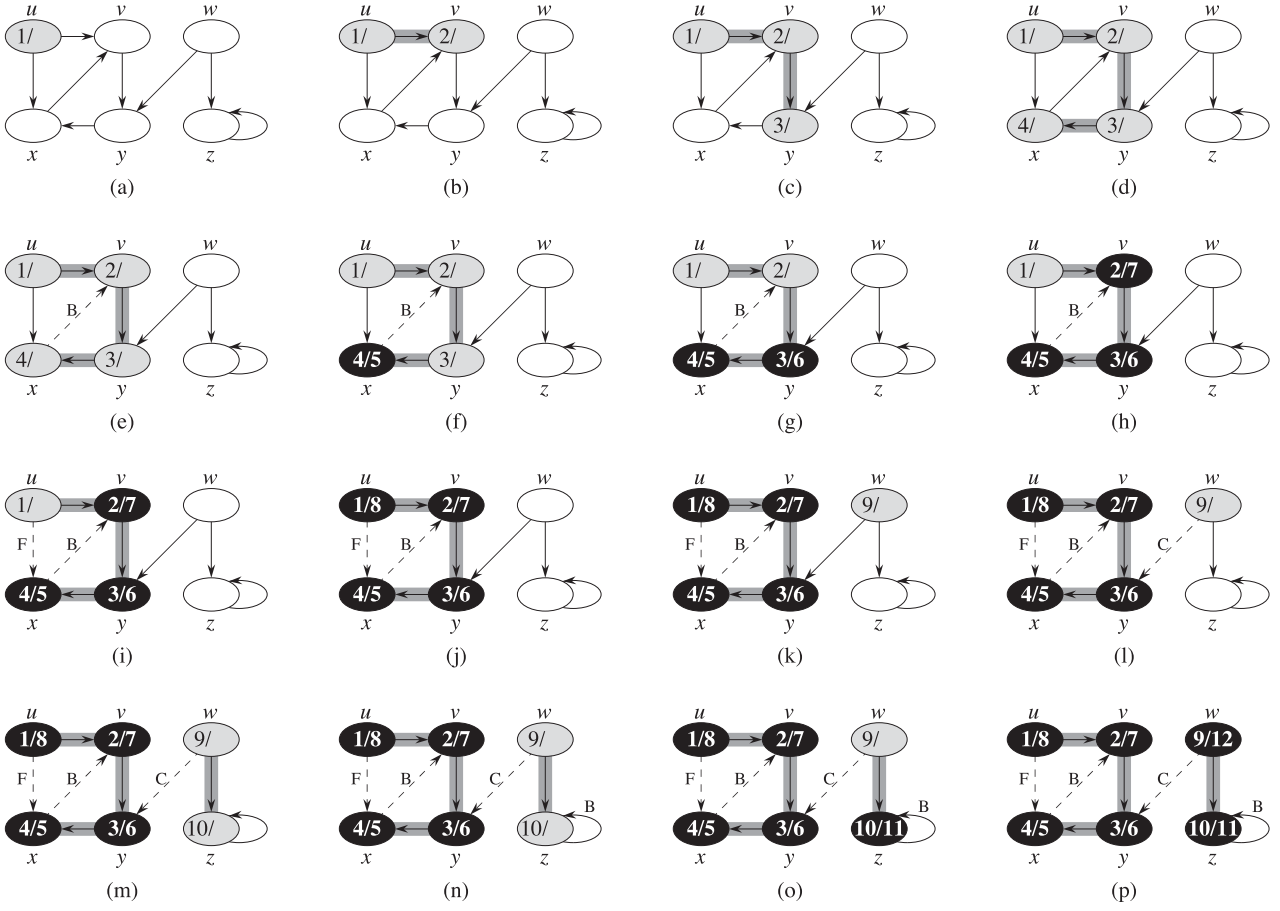
unvisited
in queue
visited

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

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

مرتب‌سازی توپولوژیکی *Topological Sort*	ارائه‌ی یک ترتیب از رئوس گراف که در آن هر رأس پس از همه‌ی ماقبل‌هایش ظاهر می‌شود.

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

مرتب‌سازی توپولوژیکی

شبه کد

TOPOLOGICAL SORT

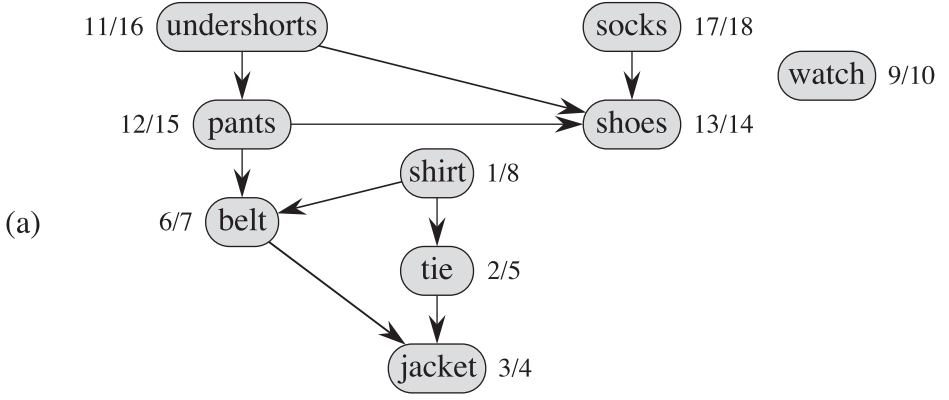
```
TOPOLOGICAL-SORT(G)
1  call DFS(G) to compute finishing times f[v] for each vertex v
2  as each vertex is finished, insert it onto the front of a linked list
3  return the linked list of vertices
```

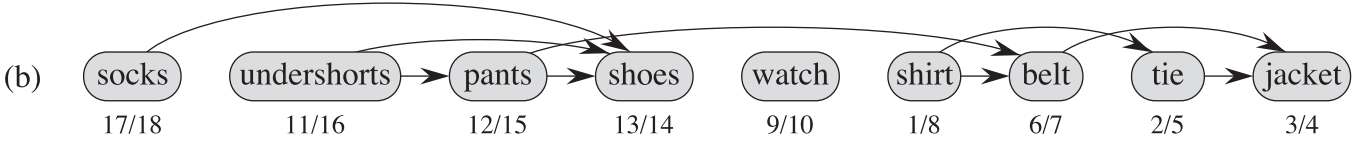
We can perform a topological sort in time $\Theta(V + E)$, since depth-first search takes $\Theta(V + E)$ time and it takes $O(1)$ time to insert each of the $|V|$ vertices onto the front of the linked list.

مرتب‌سازی توپولوژیکی

مثال

TOPOLOGICAL SORT





مؤلفه‌های همبند قوی

STRONGLY CONNECTED COMPONENTS

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

شبه کد

STRONGLY CONNECTED COMPONENTS

```
STRONGLY-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
```

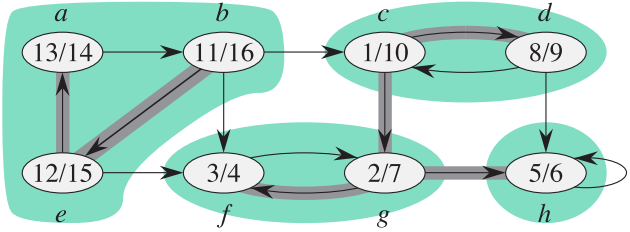
۱) با استفاده از DFS(G) زمان پایان $f[u]$ برای هر رأس u را می‌یابیم.
۲) با معکوس کردن یال‌های G^T، گراف را محاسبه می‌کنیم.
۳) DFS(G^T) را فراخوانی می‌کنیم (اما، در حلقه‌ی اصلی رأس‌ها را به ترتیب نزولی $f[u]$ در نظر می‌گیریم.)
۴) هر یک از درخت‌های پوشای حاصل، یک مؤلفه‌ی همبند قوی را نشان می‌دهد.

مؤلفه‌های همبند قوی

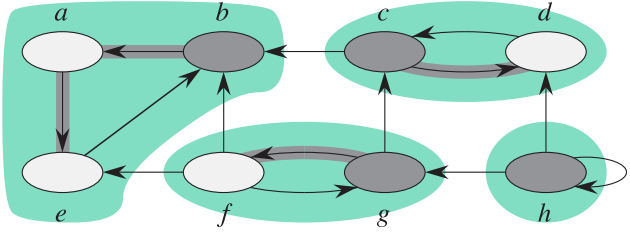
مثال

STRONGLY CONNECTED COMPONENTS

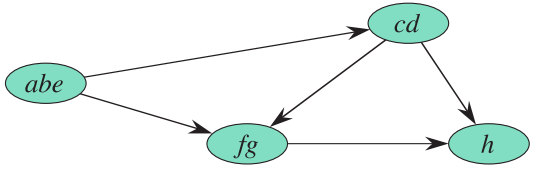
(a)



(b)



(c)



مؤلفه‌های دوهمبند

مثال

BICONNECTED COMPONENTS

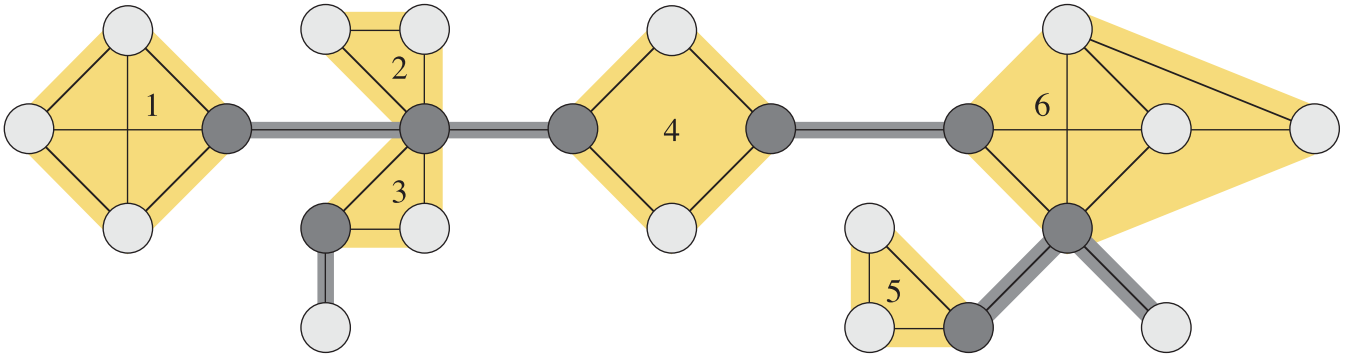
مؤلفه‌های دوهمبند *Biconnected Components*	بزرگ‌ترین زیرگراف‌های یک گراف که نقطه‌ی مفصلی ندارد. (هر مجموعه یک **مؤلفه‌ی دوهمبند** نام دارد)

نقطه‌ی مفصلی *Articulation Point*	رأسی که حذف آن و یال‌های متصل به آن، باعث ناهمبند شدن گراف شود.

مؤلفه‌های دوهمبند

مثال

BICONNECTED COMPONENTS



درخت پوشای می‌نیمال

MINIMAL SPANNING TREE

درخت پوشای می‌نیمال *Minimal Spanning Tree*	یک زیرگراف درختی پوشا از یک گراف که دارای کمترین وزن ممکن باشد.

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

کوتاه‌ترین مسیر تک‌منبع

SINGLE-SOURCE-SHORTEST PATH (SSSP)

الگوریتم دایکسترا	
الگوریتم بلمن–فورد	برای گراف‌های دارای وزن منفی

کوتاه‌ترین مسیر میان همه‌ی جفت‌ها

ALL-PAIRS SHORTEST PATH (APSP)

الگوریتم فلوید وارشال	
الگوریتم جانسون	مناسب برای گراف‌های خلوت

رنگ‌آمیزی گراف

GRAPH COLORING

رنگ‌آمیزی گراف *Graph Coloring*	انتساب رنگ به رأس‌های گراف، به‌گونه‌ای که هیچ دو رأس مجاوری هم‌رنگ نباشد.
رنگ‌آمیزی گراف با m رنگ *Graph m-Coloring*	آیا گراف مورد نظر با m رنگ قابل رنگ‌آمیزی است؟

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.)