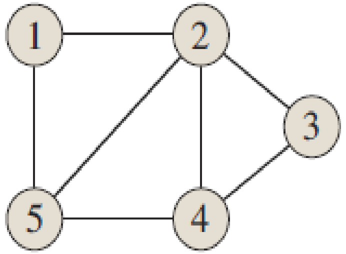
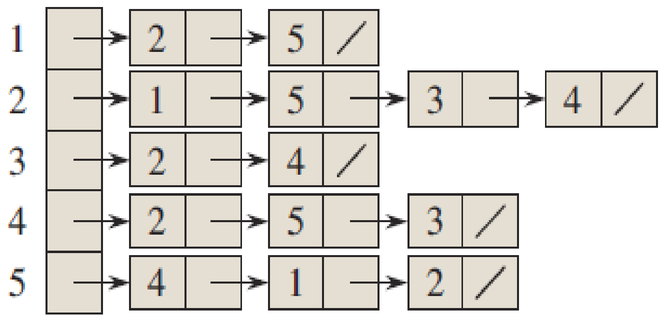
计算机问题求解 – 论题3-6
-图的计算机表示以及遍历

2016年10月08日

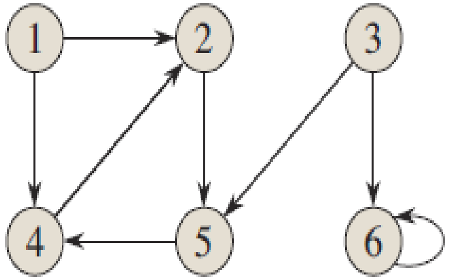


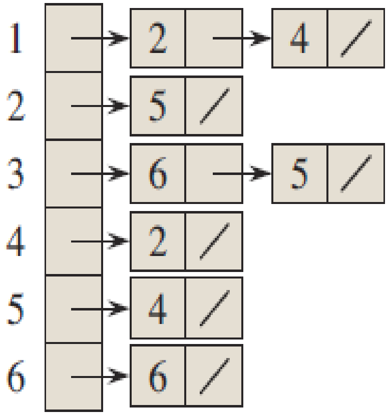


	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

问题1:

你能否根据这两组图解释计算机中最主要的图表示方式?





	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

问题2:

我们讨论表示方法是否合适主要根据什么？

你能否结合上述两种方式给以说明？

关键操作的效率 vs. 存储需求

问题3:

通常图中与应用相关的附加信息有些什么?他们对表示方法的选择有什么影响?

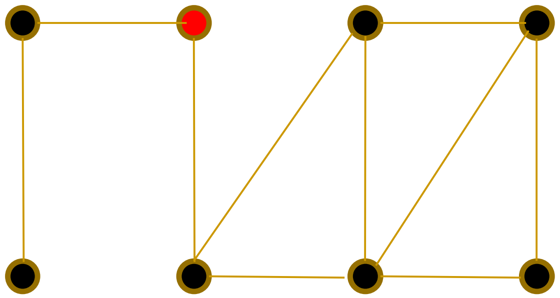
问题4:

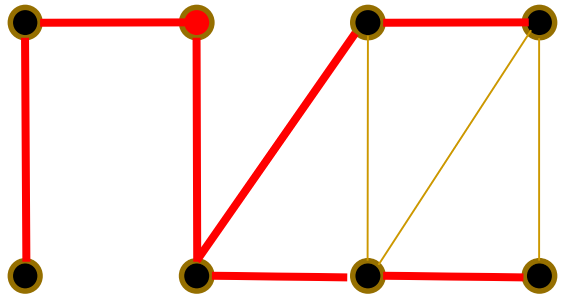
图的搜索是什么意思？为什么它是用图模型解决问题的基本操作？

图搜索经常被称为“遍历”(traversal)

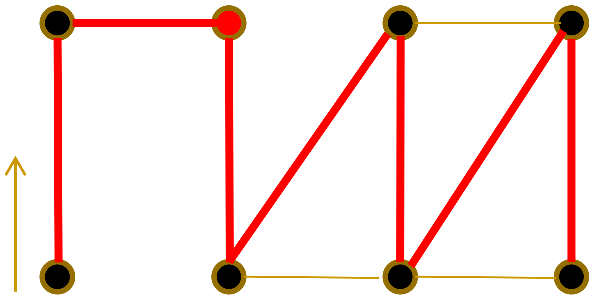
广度与深度

- 在一个连通图中，选定一个起点总可以到达所有其它点，如果我们确保任一顶点只“到达”一次，则“经过”的边不会构成回路。





广度
优先



深度
优先

搜索所“经过”的边构成的是原来图的“生成树”。

Backtracking(回朔)

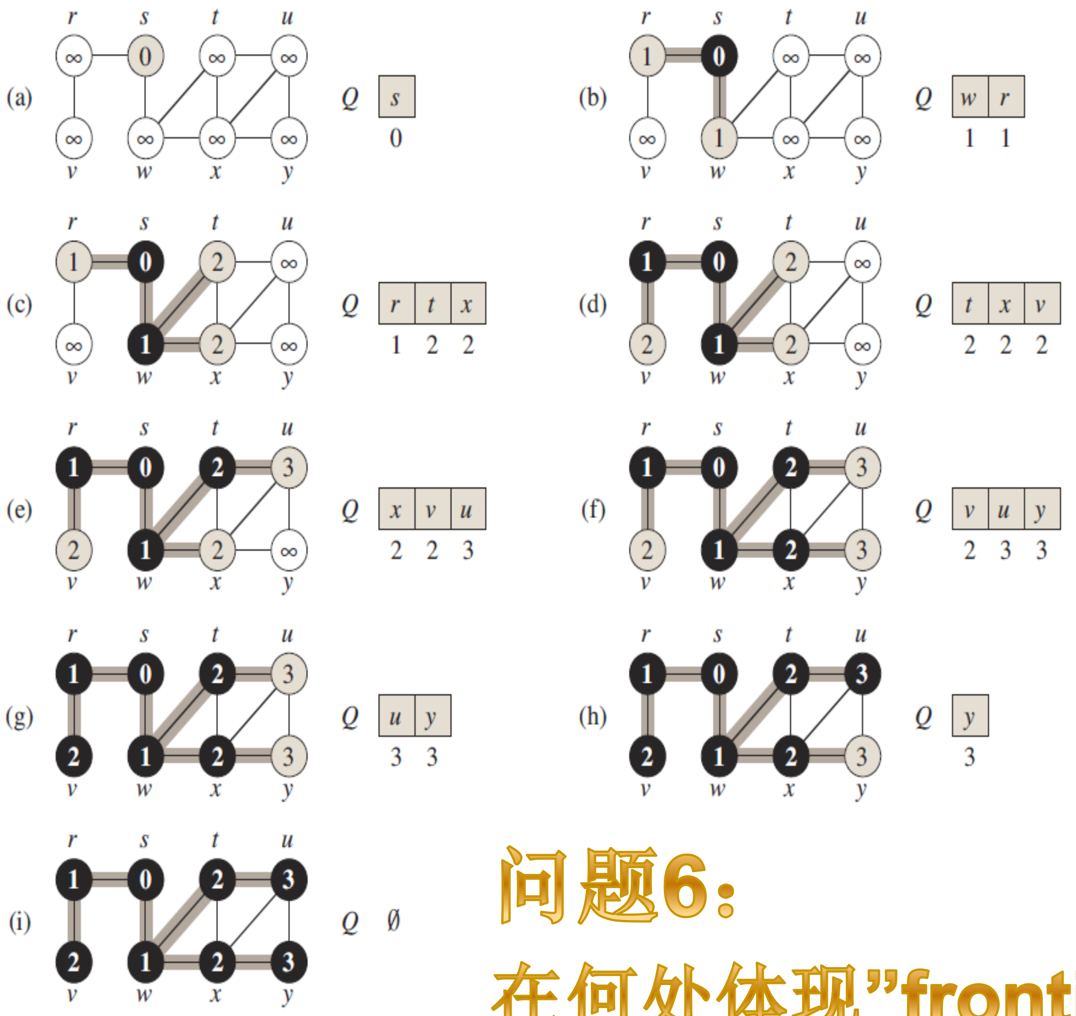
广度优先

Given a graph $G = (V, E)$ and a distinguished ***source*** vertex s, breadth-first search systematically explores the edges of G to "discover" every vertex that is reachable from s. It computes the distance (smallest number of edges) from s to each reachable vertex. It also produces a "breadth-first tree" with root s that contains all reachable vertices. For any vertex v reachable from s, the simple path in the breadth-first tree from s to v corresponds to a "shortest path" from s to v in G, that is, a path containing the smallest number of edges. The algorithm works on both directed and undirected graphs.

两个关键的动词

问题5：

图搜索时用什么办法来跟踪搜索的进度？




```
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
```

问题6：

在何处体现"frontier expansion"？为什么"扩张"的结果一定是树？

问题7：

为什么说广度优先搜索的代价是线性的？其问题规模是用什么参数表示的？

```
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
```

问题8:

为什么我们在讨论BFS算法时特别关注算法能够正确计算出最短路径距离?

$v.d$ 是δ(s,v)的上界

我们要证明的结论“$v.d = \delta(s,v)$”

和“$v.d$ 是δ(s,v)的上界”

有什么关系？

广度优先搜索计算最短路长度

算法终止时 $v.d = \delta(s, v)$ for all $v \in V$

Why?

证明要点：假设顶点v是不满足上述条件的定点中δ(s,v)值最小的一个，针对v用反证法证明。

设u是从s到v的最短路上v的直接前驱点，则 $v.d > \delta(s,v) = \boxed{\delta(s,u) + 1 = u.d + 1}$

Now consider the time when BFS chooses to dequeue vertex u from Q in line 11. At this time, vertex v is either white, gray, or black. We shall show that in each of these cases, we derive a contradiction to inequality (22.1). If v is white, then line 15 sets $v.d = u.d + 1$, contradicting inequality (22.1). If v is black, then it was already removed from the queue and, by Corollary 22.4, we have $v.d \le u.d$, again contradicting inequality (22.1). If v is gray, then it was painted gray upon dequeuing some vertex w, which was removed from Q earlier than u and for which $v.d = w.d + 1$. By Corollary 22.4, however, $w.d \le u.d$, and so we have $v.d = w.d + 1 \le u.d + 1$, once again contradicting inequality (22.1).

$v.d$ 是δ(s,v)的上界

问题8：

你能解释是怎么归纳的吗？

Lemma 22.2

Let $G = (V, E)$ be a directed or undirected graph, and suppose that BFS is run on G from a given source vertex $s \in V$. Then upon termination, for each vertex $v \in V$, the value $v.d$ computed by BFS satisfies $v.d \geq \delta(s, v)$.

Proof We use induction on the number of ENQUEUE operations. Our inductive hypothesis is that $v.d \geq \delta(s, v)$ for all $v \in V$.

The basis of the induction is the situation immediately after enqueuing s in line 9 of BFS. The inductive hypothesis holds here, because $s.d = 0 = \delta(s, s)$ and $v.d = \infty \geq \delta(s, v)$ for all $v \in V - \{s\}$.

For the inductive step, consider a white vertex v that is discovered during the search from a vertex u. The inductive hypothesis implies that $u.d \geq \delta(s, u)$. From the assignment performed by line 15 and from Lemma 22.1, we obtain

$$\begin{aligned} v.d &= u.d + 1 \\ &\geq \delta(s, u) + 1 \\ &\geq \delta(s, v) . \end{aligned}$$

为什么？

Vertex v is then enqueued, and it is never enqueued again because it is also grayed and the **then** clause of lines 14–17 is executed only for white vertices. Thus, the value of $v.d$ never changes again, and the inductive hypothesis is maintained. ■

进队列的次序与$v.d$值的大小

Suppose that vertices v_i and v_j are enqueued during the execution of BFS, and that v_i is enqueued before v_j. Then $v_i.d \le v_j.d$ at the time that v_j is enqueued.

注意：每个顶点被赋一次有限的$.d$值，之后再不改变。

其实：同时在队列中的顶点的 $.d$ 值是非递减的，差值最多为1

while $Q \neq \emptyset$
　　$u = \text{DEQUEUE}(Q)$
　　for each $v \in G.Adj[u]$
　　　　if $v.color == \text{WHITE}$
　　　　　　$v.color = \text{GRAY}$
　　　　　　$\boxed{v.d = u.d + 1}$
　　　　　　$v.\pi = u$
　　　　　　$\text{ENQUEUE}(Q, v)$
　　$u.color = \text{BLACK}$

问题9：

你能根据代码直观地解释一下为什么吗？

广度优先搜索计算最短路长度

算法终止时 $v.d = \delta(s, v)$ for all $v \in V$

证明要点：假设顶点v是不满足上述条件的定点中$.d$值最小的一个，针对v用反证法证明。

Why?

设u是从s到v的最短路上v的直接前驱点，则 $v.d > \delta(s,v) = \boxed{\delta(s,u) + 1 = u.d + 1}$

Now consider the time when BFS chooses to dequeue vertex u from Q in line 11. At this time, vertex v is either white, gray, or black. We shall show that in each of these cases, we derive a contradiction to inequality (22.1). If v is white, then line 15 sets $v.d = u.d + 1$, contradicting inequality (22.1). If v is black, then it was already removed from the queue and, by Corollary 22.4, we have $v.d \le u.d$, again contradicting inequality (22.1). If v is gray, then it was painted gray upon dequeuing some vertex w, which was removed from Q earlier than u and for which $v.d = w.d + 1$. By Corollary 22.4, however, $w.d \le u.d$, and so we have $v.d = w.d + 1 \le u.d + 1$, once again contradicting inequality (22.1).

深度优先搜索

Depth-first search explores edges out of the most recently discovered vertex v that still has unexplored edges leaving it.

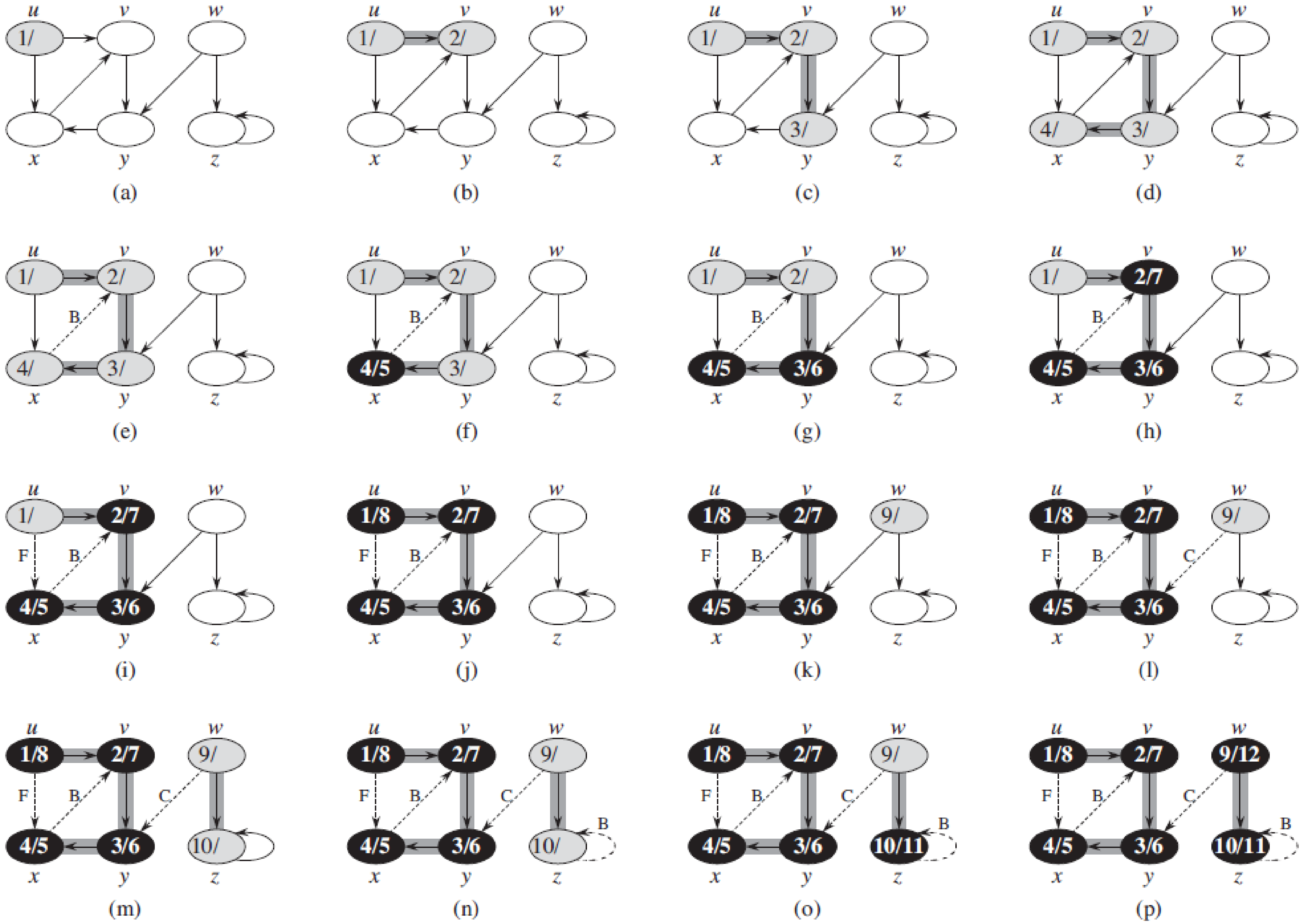
Each vertex is initially white, is grayed when it is ***discovered*** in the search, and is blackened when it is ***finished***, that is, when its adjacency list has been examined completely.

深度优先搜索

注意：*time*

```
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              // white vertex u has just been discovered
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]        // explore edge (u, v)
    if v.color == WHITE
        v.π = u
        DFS-VISIT(G, v)
u.color = BLACK              // blacken u; it is finished
time = time + 1
u.f = time
```


(a)
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)
(k)
(l)
(m)
(n)
(o)
(p)

深度优先搜索也是线性算法

What is the running time of DFS? The loops on lines 1–3 and lines 5–7 of DFS take time $\Theta(V)$, exclusive of the time to execute the calls to DFS-VISIT. As we did for breadth-first search, we use aggregate analysis. The procedure DFS-VISIT is called exactly once for each vertex $v \in V$, since the vertex u on which DFS-VISIT is invoked must be white and the first thing DFS-VISIT does is paint vertex u gray. During an execution of DFS-VISIT(G, v), the loop on lines 4–7 executes $|Adj[v]|$ times. Since

$$\sum_{v \in V} |Adj[v]| = \Theta(E) ,$$

the total cost of executing lines 4–7 of DFS-VISIT is $\Theta(E)$. The running time of DFS is therefore $\Theta(V + E)$.

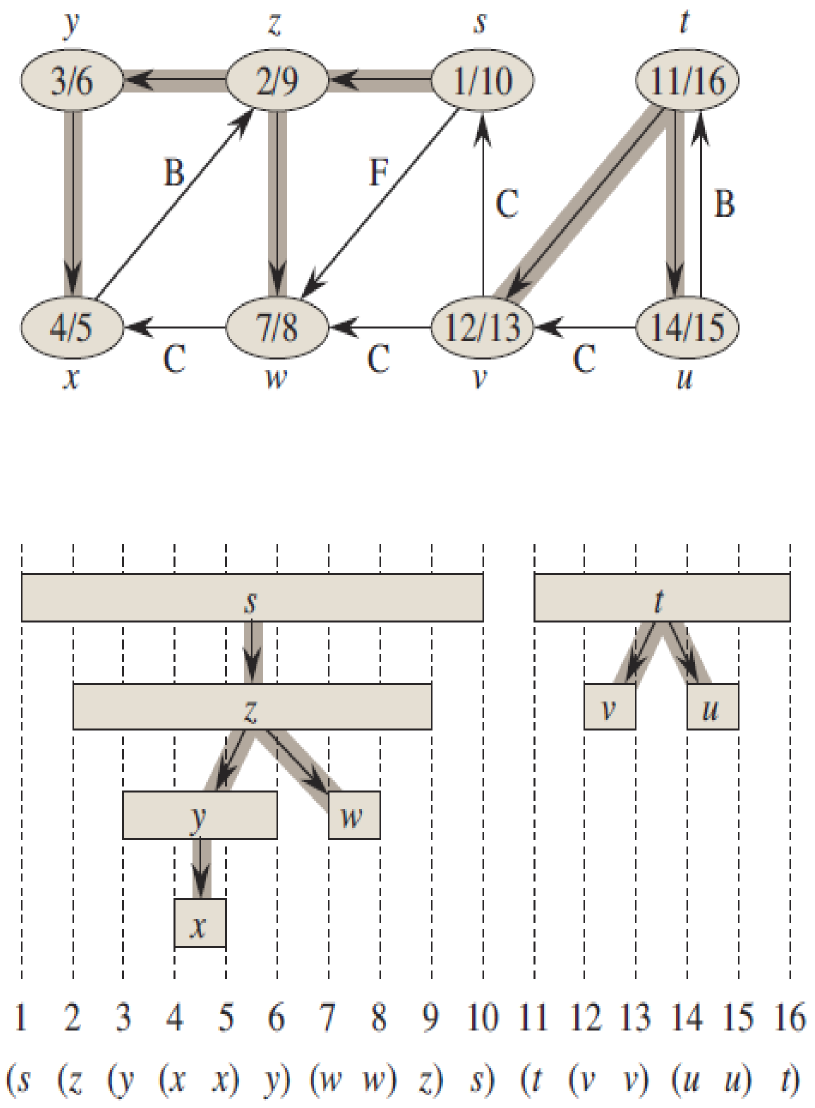
问题10：

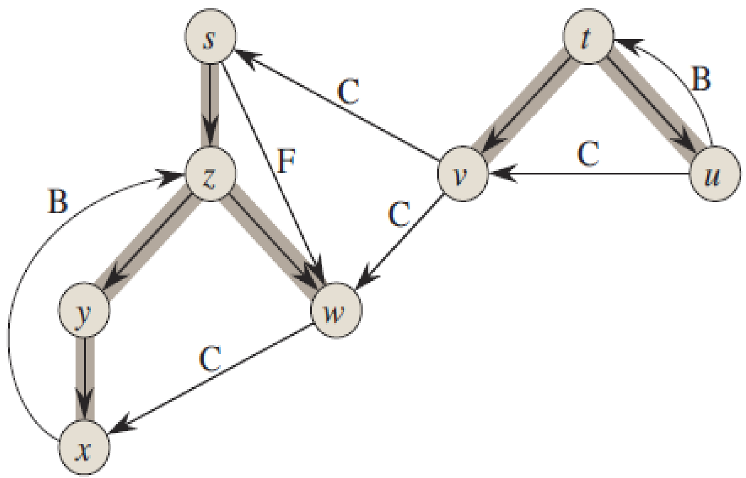
为什么区分黑色顶点与灰色定点对于深度优先很重要，对于广度优先其实不重要？

问题11：

为什么对深度优先需要引入“时间戳”？

$u.d$ 和 $u.f$ 的含义是什么？





问题12：

你能解释深度优先搜索森林内/外的边与顶点活动时间段之间的关系吗？

Theorem 22.9 (White-path theorem)

In a depth-first forest of a (directed or undirected) graph $G = (V, E)$, vertex v is a descendant of vertex u if and only if at the time $u.d$ that the search discovers u, there is a path from u to v consisting entirely of white vertices.

Proof $\Rightarrow$: If $v = u$, then the path from u to v contains just vertex u, which is still white when we set the value of $u.d$. Now, suppose that v is a proper descendant of u in the depth-first forest. By Corollary 22.8, $u.d < v.d$, and so v is white at time $u.d$. Since v can be any descendant of u, all vertices on the unique simple path from u to v in the depth-first forest are white at time $u.d$.

$\Leftarrow$: Suppose that there is a path of white vertices from u to v at time $u.d$, but v does not become a descendant of u in the depth-first tree. Without loss of generality, assume that every vertex other than v along the path becomes a descendant of u. (Otherwise, let v be the closest vertex to u along the path that doesn't become a descendant of u.) Let w be the predecessor of v in the path, so that w is a descendant of u (w and u may in fact be the same vertex). By Corollary 22.8, $w.f \leq u.f$. Because v must be discovered after u is discovered, but before w is finished, we have $u.d < v.d < w.f \leq u.f$. Theorem 22.7 then implies that the interval $[v.d, v.f]$ is contained entirely within the interval $[u.d, u.f]$. By Corollary 22.8, v must after all be a descendant of u. ■

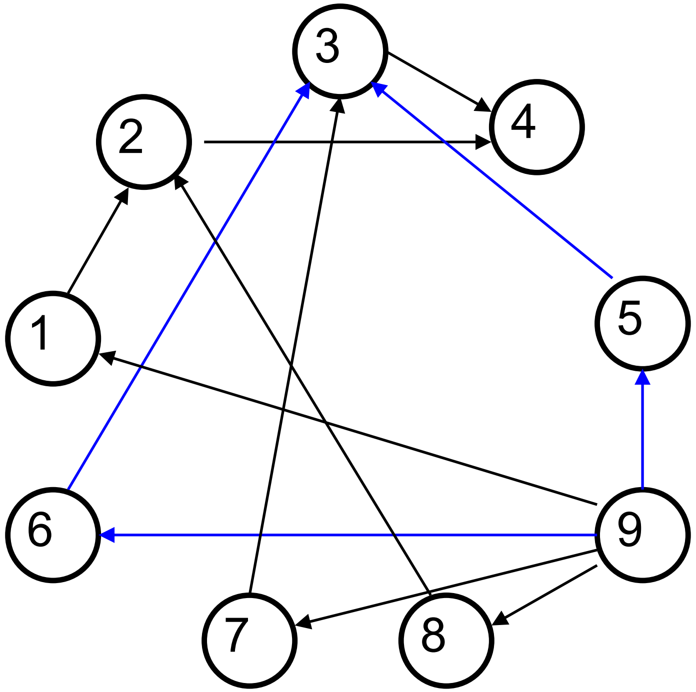
问题13：

深度优先搜索对于无向图与有向图有何不同？

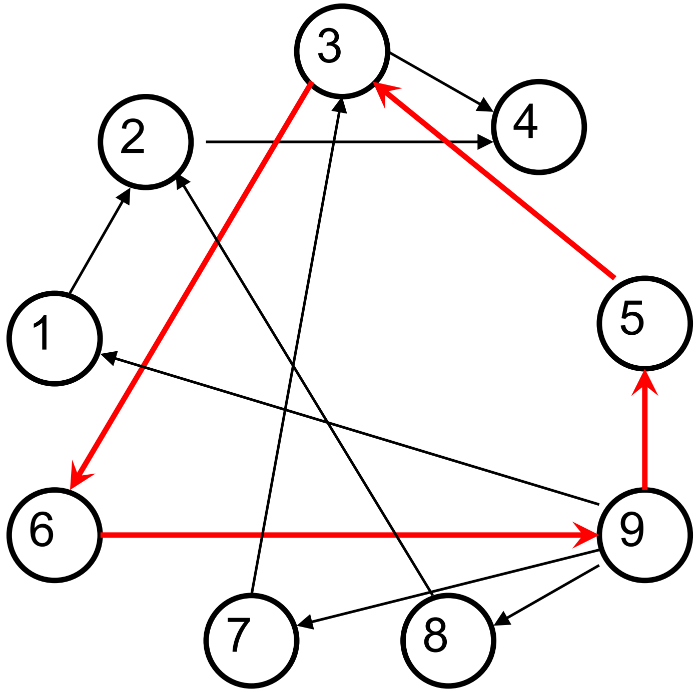
问题14:

广度优先搜索是利用队列实现的，那深度优先搜索用什么数据结构呢?为什么有这样的差别?

Directed Acyclic Graph (DAG)



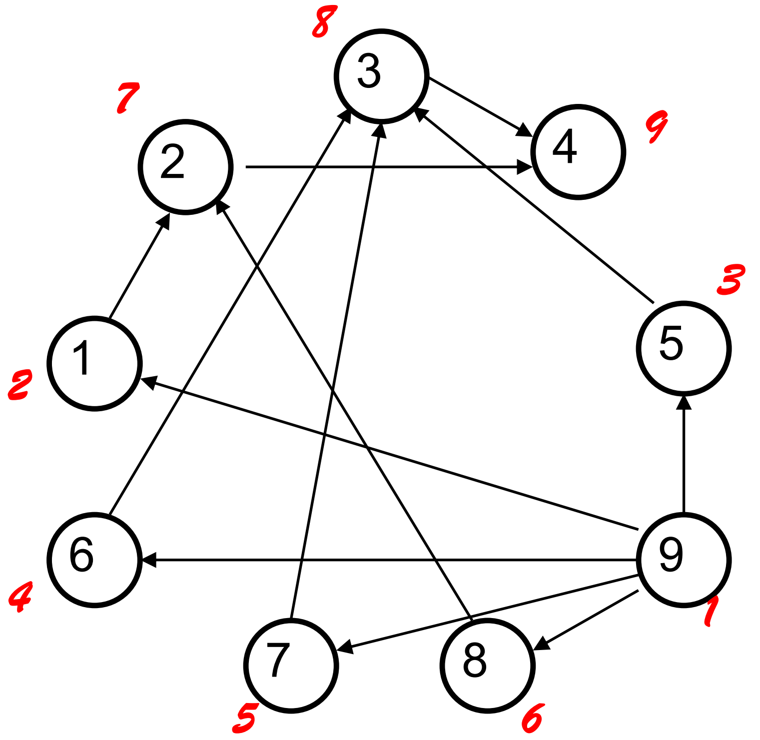
A Directed Acyclic Graph



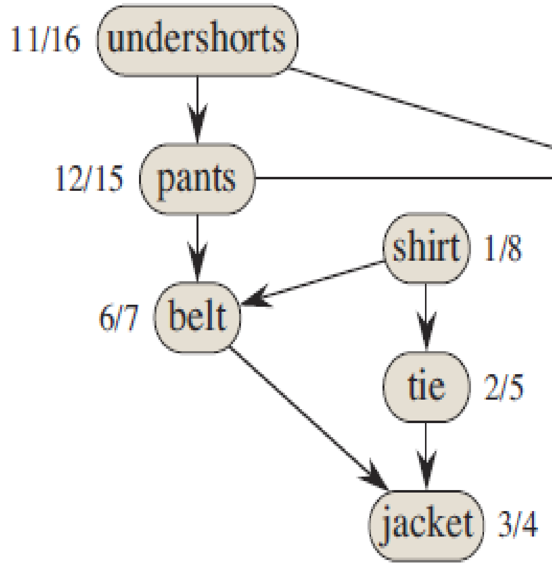
Not a DAG

Topological Order(拓扑序)

- G=(V,E) is a directed graph with *n* vertices. A **topological order** for G is an assignment of distinct integer 1,2,…, *n* to the vertices of V as their **topological number**, such that, for every $vw \in E$, the topological number of v is less than that of w.
- Reverse topological order can be defined similarly, ("greater than")



(a)



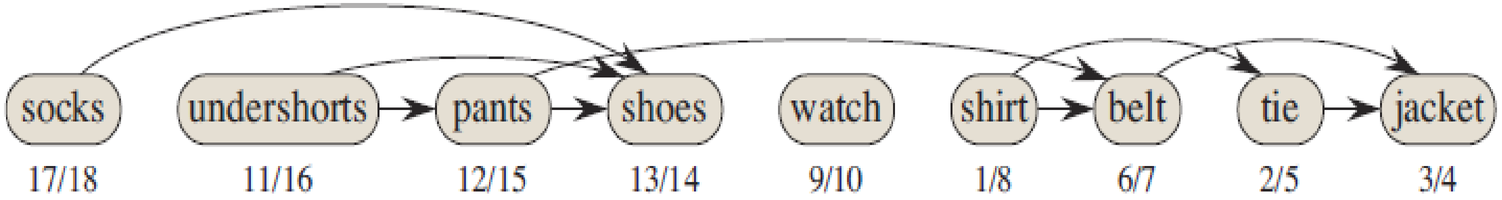
其实，可以将DFS看作一个算法的skeleton!

TOPOLOGICAL-SORT(G)

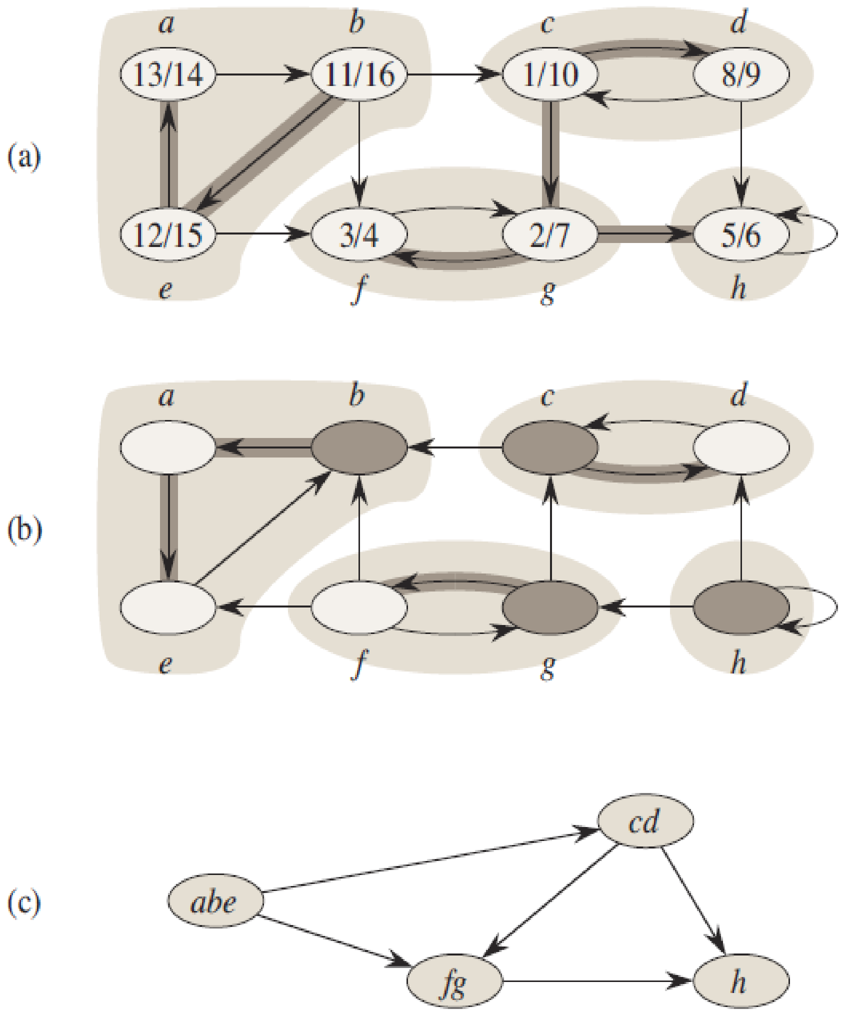
```
call DFS(G) to compute finishing times v.f for each vertex v
as each vertex is finished, insert it onto the front of a linked list
return the linked list of vertices
```

排序的结果不是唯一的。

(b)



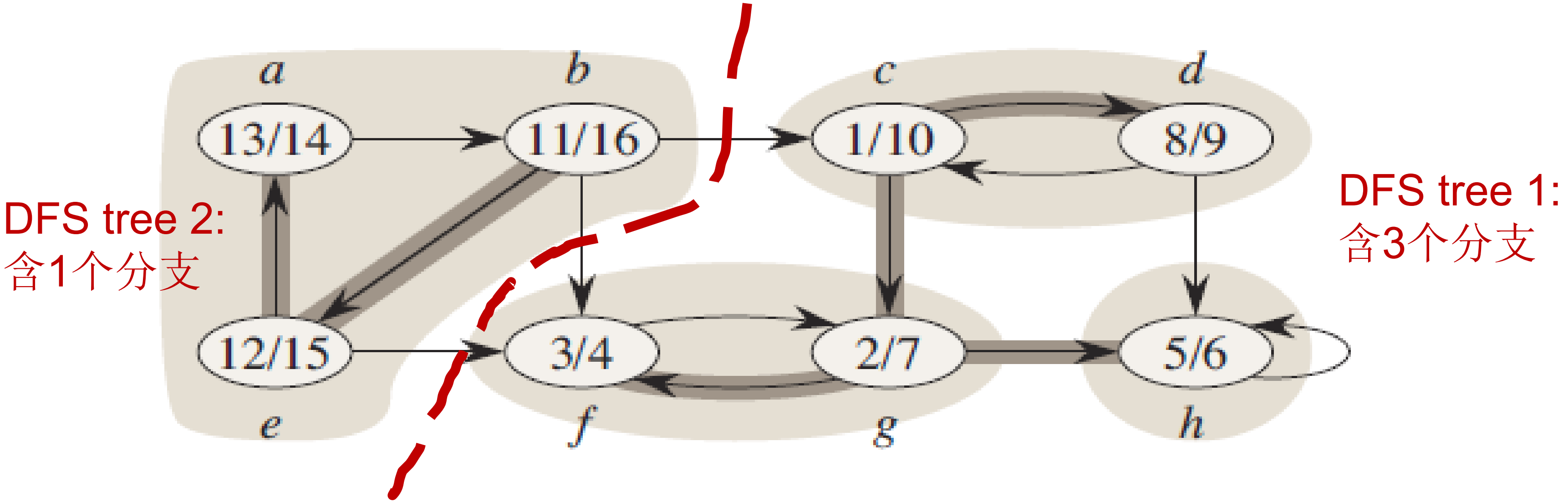
有向图中的强连通分支问题



此有向图含4个强连通分支

在“转置图”中，结果不变。

理解强分支算法的关键



- DFS算法能将图分割成DFS trees, 但不能区分强分支。
- 任何一个强分支一定完整的包含在一个DFS tree中。
- 位于同一DFS tree，但不同强分支中两点通路一定是单向的。因此将一个分支看作一个点得到的图是DAG。
- 解决问题的关键：先将图分解为正常的DFS trees, 分别对各个tree的转置图再做DFS, 按照特别的顺序使得从选定顶点出发只能达到一个强分支内的顶点，不能到达其它顶点的原因可能是因为通路已不存在（转置），或者那些顶点已经是黑色的了。

STRONGLY-CONNECTED-COMPONENTS(G)

1 call DFS(G) to compute finishing times $u.f$ 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 $u.f$ (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

即在前一次DFS中后finish的
点会先被搜索，这如何实现呢？

问题15：

从应用角度看，你认为两种图遍历方法最大的差别是什么？

课外作业

- TC pp.592-: ex.22.1-3; 22.1-8
- TC pp.601-: ex.22.2-3; 22.2-4; 22.2-5
- TC pp.610-: ex.22.2-6; 22.3-7; 22.3-8; 22.3-9; 22.3-12
- TC pp.614-: ex.22.4-2; 22.4-3
- TC pp.620: ex.22.5-5; 22.5-7