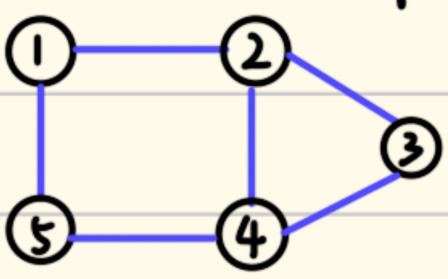
# Note 12 Graph Algorithm I

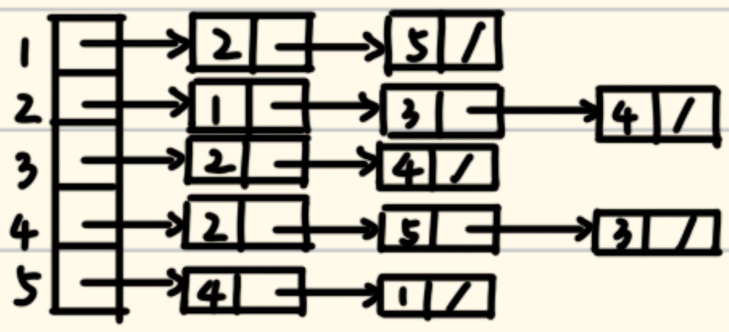
## BFS, DFS, Topological Sort

### (I) Representation of A Graph

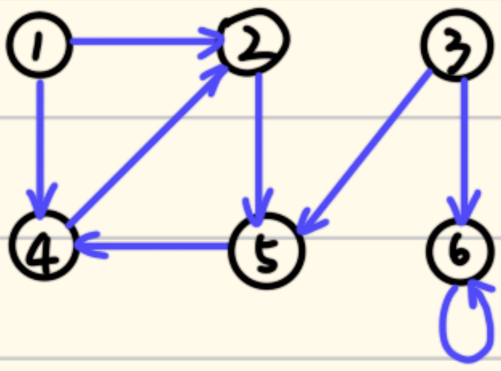
#### 1. Adjacency-List representation

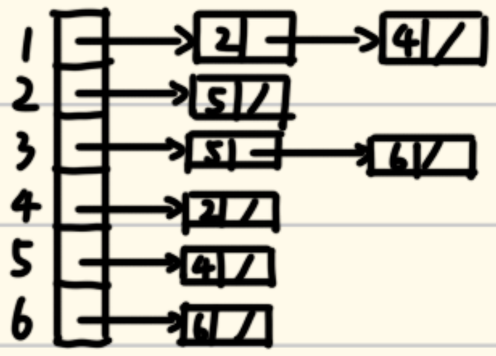
- Directed Graph





- Undirected Graph





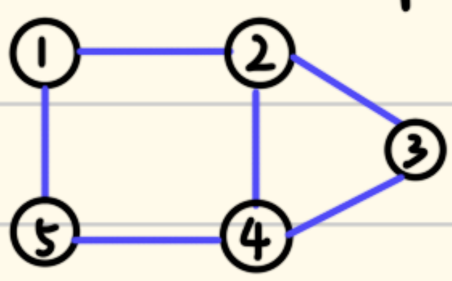
(若有权，在node中加一元素)

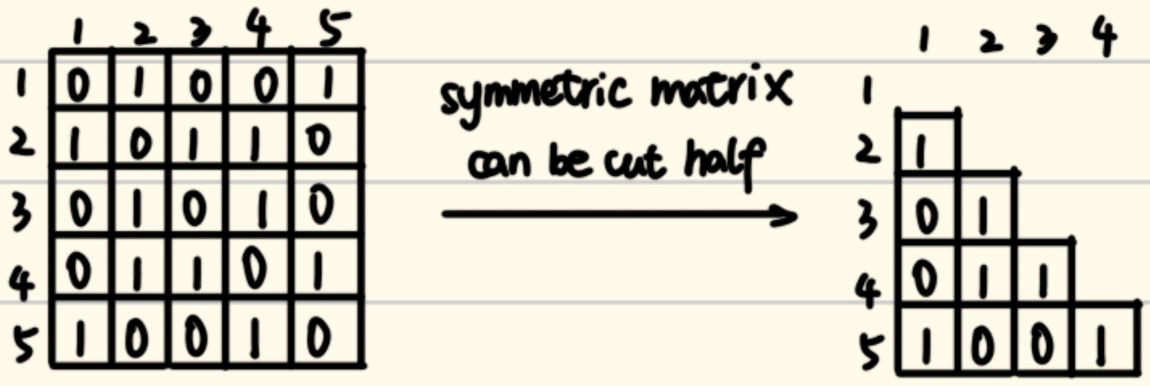
Space Complexity: $\Theta(|V|+|E|)$

适用于稀疏图

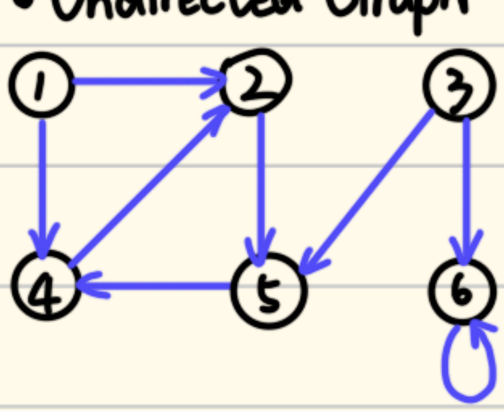
#### 2. Adjacency-Matrix representation

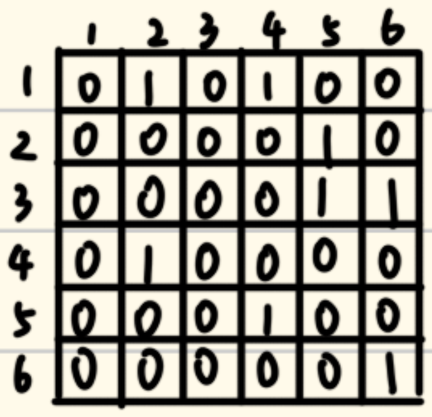
- Directed Graph






- Undirected Graph





(若有权，直接改矩阵元素的值)

Space Complexity: $\Theta(|V|^2)$

适用于稠密图

# (11) Breadth-First-Search

## 1. Algorithm

### (1) 基本思想：

为了跟踪算法的进展，广度优先搜索在概念上将每个结点涂上白色、灰色或黑色。所有结点在一开始的时候均涂上白色。在算法推进过程中，这些结点可能会变成灰色或者黑色。在搜索过程中，第一次遇到一个结点就称该结点被“**发现**”，此时该结点的颜色将发生改变。因此，凡是灰色和黑色的结点都是已被发现的结点。但广度优先搜索对灰色和黑色结点加以区别，以确保搜索按照广度优先模式进行推进[⊖]。如果边 $(u,v) \in E$ 且结点 $u$ 是黑色，则结点 $v$ 既可能是灰色也可能是黑色。也就是说，所有与黑色结点邻接的结点都已经被发现。对于灰色结点来说，其邻接结点中可能存在未被发现的白色结点。灰色结点所代表的就是已知和未知两个集合之间的边界。

用 Queue 作为 fringe　　(白：未发现，灰：正处理，黑：处理完)

第一次发现：白→灰　　从 Queue 中弹出：灰→黑

Fringe 里面全是灰色结点

### (2) 伪码：　(从一个源出发的 BFS/遍历)　　(无论是否有向，均是如此)

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

π 是先导，用于重建搜索出来

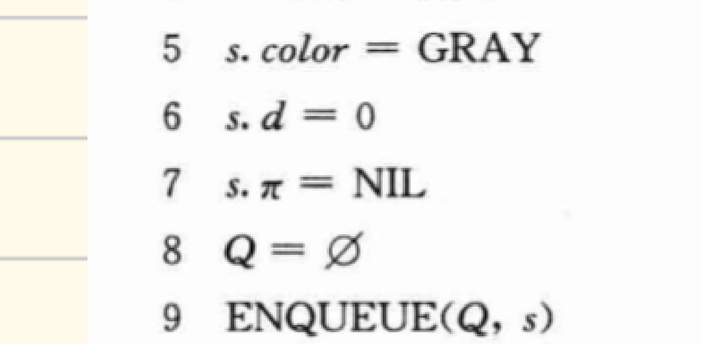
### (3) Time Complexity：

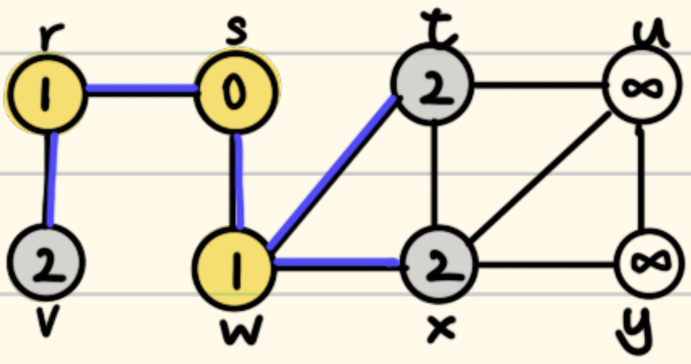
每个顶点出一次 Fringe (第11行) ⇒ O(V)

每条边考察一次 (第12行) ⇒ O(E)

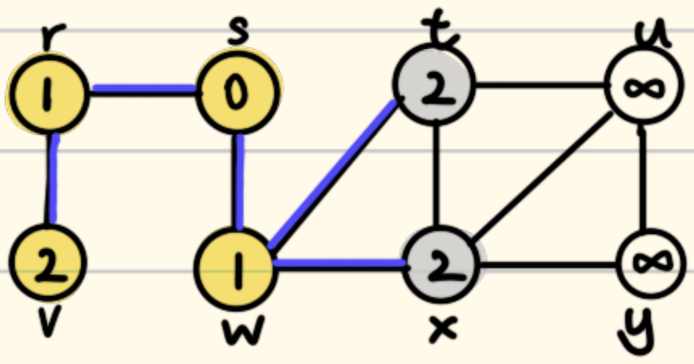
⇒ O(V+E)

## 2. Example:

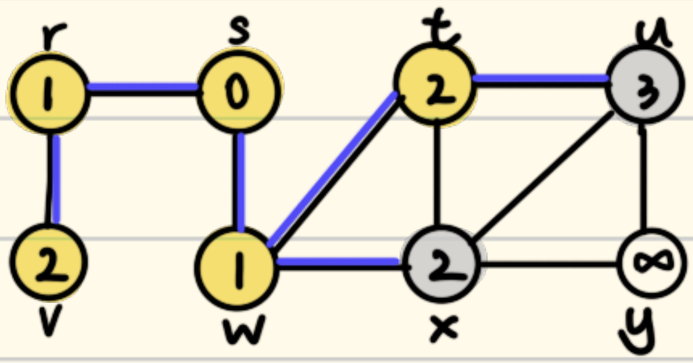




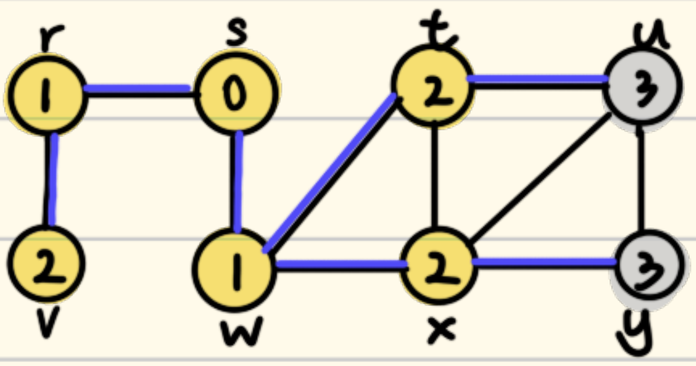

Q | v | t | x |



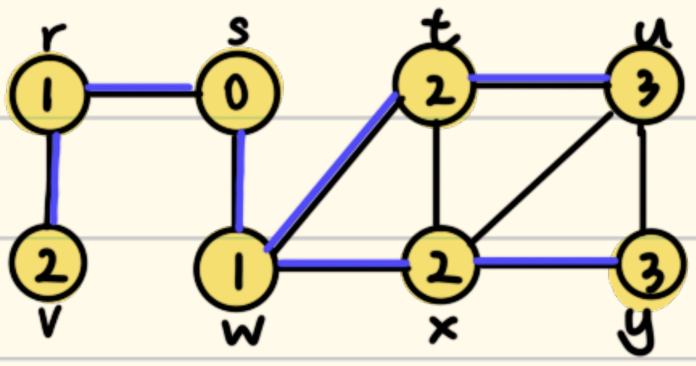

Q | t | x |




Q | x | u |



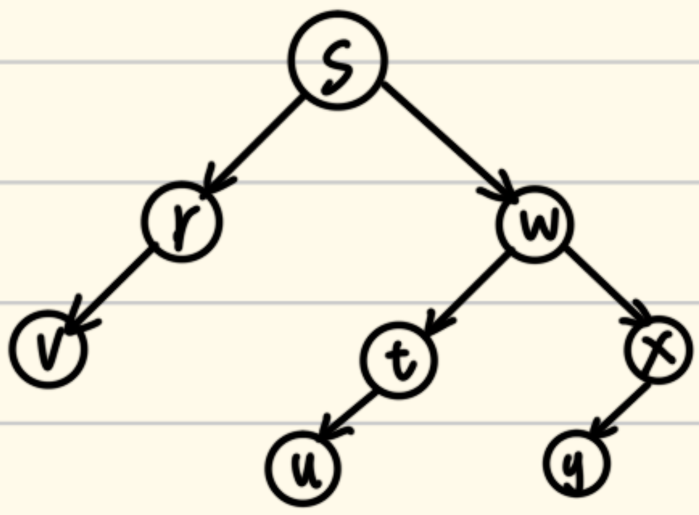

Q | u | y |




Q |

## 3. 广度优先树

单源 BFS 遍历会创造一棵广度优先树 (结点的父亲是它的 先导 π)




多源 BFS 可能会创造广度优先森林

(不连通的无向图，弱连通的有向图，……)

# (III) Depth-First-Search

## 1. Algorithm

### (1) 基本思想

像广度优先搜索算法一样，深度优先搜索算法在搜索过程中也是对结点进行涂色来指明结点的状态。每个结点的初始颜色都是白色，在结点被**发现**后变为灰色，在其邻接链表被扫描**完成**后变为黑色。该方法可以保证每个结点仅在一棵深度优先树中出现，因此，所有的深度优先树是不相交的(disjoint)。

除了创建一个深度优先搜索森林外，深度优先搜索算法还在每个结点盖上一个**时间戳**。每个结点 $v$ 有两个时间戳：第一个时间戳 $v.d$ 记录结点 $v$ 第一次被发现的时间(涂上灰色的时候)，第二个时间戳 $v.f$ 记录的是搜索完成对 $v$ 的邻接链表扫描的时间(涂上黑色的时候)。这些时间戳提供了图结构的重要信息，通常能够帮助推断深度优

用 Stack 作为 fringe

(白:未发现，灰:正处理，黑:处理完)

第一次发现: 白→灰

完成对其邻接链表的扫描: 灰→黑.

Fringe 里面全是灰色结点.

### (2) 伪码

(图中所有可能的不相连部分，分别做DFS遍历)

(无论是否有向，均是如此)

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

→如果有部分还没进行过DFS遍历，就对这部分DFS遍历

(从单源出发进行DFS遍历)

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

递归写法，也可用栈作为 fringe.

### (3) Time Complexity:

每个顶点出一次 Fringe (第11行) ⇒ O(

每条边考察一次 (第12行) ⇒ O(E)

} ⇒ O(V+E)

## 2 · example

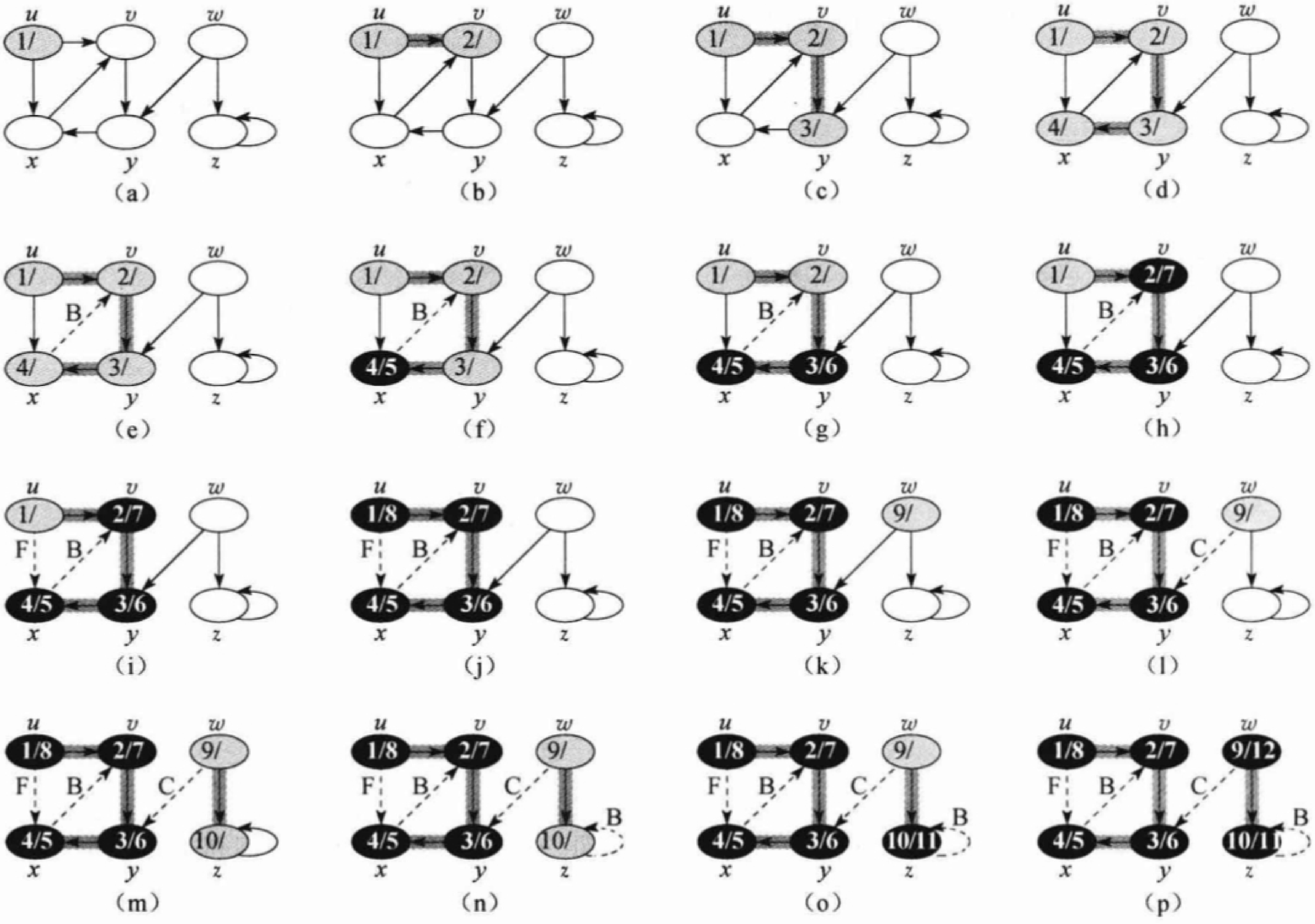
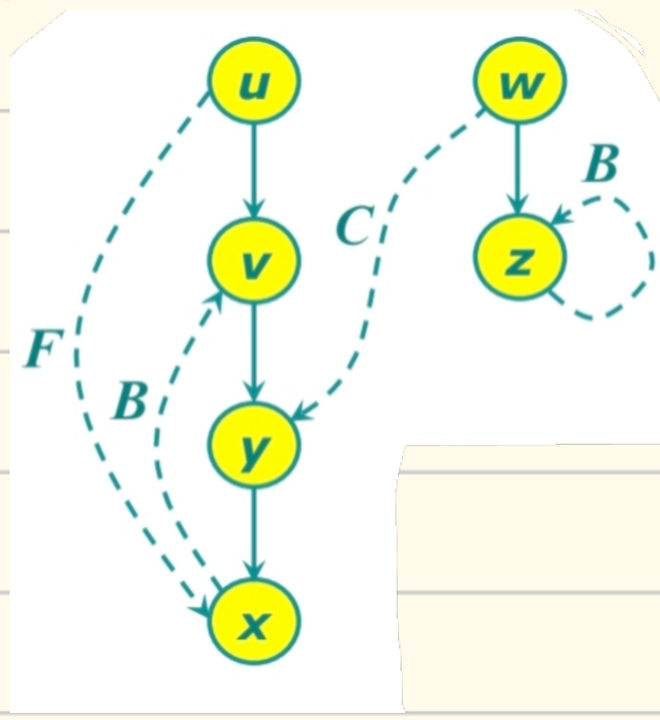



图 22-4 深度优先搜索算法 DFS 在有向图上的运行过程。随着算法对边的探索的推进，这些边或者变成有阴影的边(如果它们是树边)，或者变为虚线边(其他情况)。非树边则根据其为后向(back)边、横向(cross)边或前向(forward)边而分别标记为 B、C 或 F。结点中的时间戳表明该结点的发现时间和完成时间

## 3 · 深度优先树/森林

- 单源 DFS 遍历会创造一棵深度优先树 (结点的父亲是它的 父亲π)
- 多源DFS 可能会创造深度优先森林
  (不连通的无向图，弱连通的 有向图，·····)




边的分类：边(u,v)的颜色可根据边第一次被探索时 V的颜色分类

- V是白色 ⇒ 实边，tree edge
- V是灰色 ⇒ 虚边，back edge
- V是黑色 ⇒ 虚边
  - d(u) < d(v)　forward edge
  - d(u) > d(v)　cross edge

# (IV) Topological Sort : Application of DFS

下面的简单算法可以对一个有向无环图进行拓扑排序：

```
TOPOLOGICAL-SORT(G)
1  call DFS(G) to compute finishing times v.f 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
```

按照完成时间戳 f 进行从大到小的排序

A ***topological sort*** of a directed acyclic graph or "***dag***" $G = (V, E)$ is a linear ordering of all its vertices such that if $G$ contains an edge $(u, v)$, then $u$ appears before $v$ in the ordering.

- Topological sort of a graph can be viewed as an ***linear ordering*** of its vertices.
- Topological sorting is ***different*** from the usual kind of "sorting" studied before.
- If the graph is ***not acyclic***, then no linear ordering is possible.

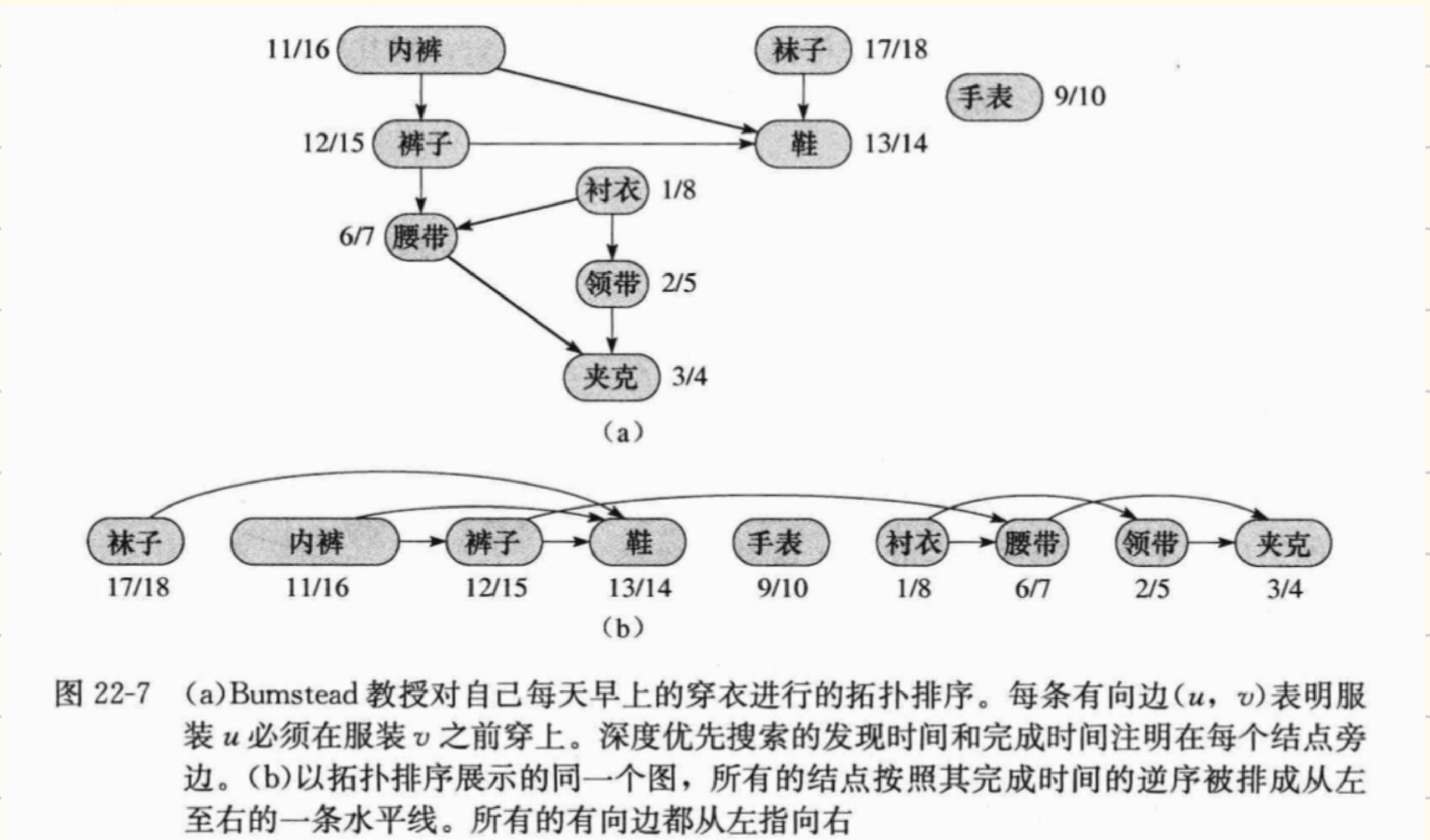



图 22-7 (a)Bumstead 教授对自己每天早上的穿衣进行的拓扑排序。每条有向边$(u, v)$表明服装 $u$ 必须在服装 $v$ 之前穿上。深度优先搜索的发现时间和完成时间注明在每个结点旁边。(b)以拓扑排序展示的同一个图，所有的结点按照其完成时间的逆序被排成从左至右的一条水平线。所有的有向边都从左指向右