图搜索算法

目录

图的基本概念(1/5)

定义

图(*graph*)G是一个二元组(V, E), 其中:

- V是结点(*vertice*)集合,
- $E \subseteq V \times V$是结点上的边(*edge*)的关系的集合.

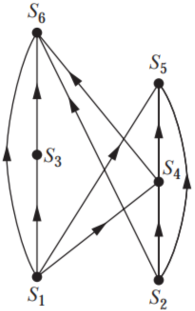
application	item	connection
map	intersection	road
web content	page	link
circuit	device	wire
schedule	job	constraint
commerce	customer	transaction
matching	student	application
computer network	site	connection
software	method	call
social network	person	friendship

Typical graph applications

图的基本概念(2/5)

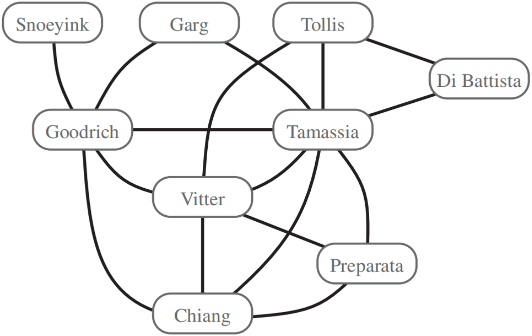
有向图(directed graph):

S_1	$a := 0$
S_2	$b := 1$
S_3	$c := a + 1$
S_4	$d := b + a$
S_5	$e := d + 1$
S_6	$e := c + d$



图的基本概念(3/5)

无向图(undirected graph): 研究人员之间的合作关系



图的基本概念(4/5)

定义

(1) 图的**路径**(*path*)是指结点和边交替连接的一个序列. 它从一个结点开始, 到另一个结点结束, 使得每一条边都与它的前一个结点和下一个结点相关联.

(2) 如果一条路径上的每个结点都是不同的, 则称该路径是**简单的**(*simple*).

(3) **回路**(*cycle*)指一条起点和终点相同的路径. 如果回路上的每个结点都是不同的, 第一个和最后一个结点除外, 称此回路为**简单回路**.

定理

在图G中, 如果从结点u到结点v($u \neq v$)存在一条路径, 则从结点u到结点v必存在一条不多于$|V|-1$条边的简单路径.

图的基本概念(5/5)

定义

(1) 图$G'=(V', E')$是图G的**子图**(*subgraph*), 如果$V'\subseteq V$, $E'\subseteq E$.

(2) 图G的**生成子图**(*spanning subgraph*)是包含G中所有结点的子图.

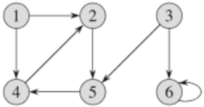
(3) 如果一个图的任意两个结点之间都存在一条路径, 则称该图是**连通的**(*connected*).

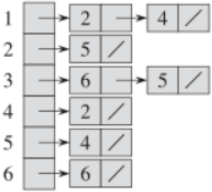
(4) 连通无(简单)回路的图称为**(无根)树**(*tree*).

(5) 无(简单)回路的图称为**森林**(*forest*).

(6) 一个图的**生成树**是该图的一个生成子图, 而且是一个树.

图的表示





	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

图的搜索

- 图的搜索指的是系统化地跟随图中的边来访问图中的每个结点.
- **图搜索算法**可以用来发现图的结构.
- 图的搜索技巧是整个图算法领域的核心.

DFS算法的导入


```
DFS(v):
    if v is unmarked
        mark v
        for each edge v→w
            DFS(w)
```

```
DFS(v):
    mark v
    PREVISIT(v)
    for each edge vw
        if w is unmarked
            parent(w) ← v
            DFS(w)
    POSTVISIT(v)
```

```
DFSALL(G):
    PREPROCESS(G)
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            DFS(v)
```

深度优先搜索基本思想

设初始状态时图中的所有顶点未被访问, 则:

(1) 先访问图中某个(未访问过的)结点v, 然后选择一个v邻接到的未被访问过的结点w, 再访问w, 并按同样方法前进;

(2) 当遇到一个所有邻接于它的结点都被访问过了的结点时, 退回到已访问结点序列中最后一个拥有相邻结点未被访问过的结点, 访问它的一个未被访问过的相邻结点u, 再从u 出发按同样方法前进;

(3) 当所有已被访问过的结点的相邻结点都被访问时, 如果图中还有未被访问的顶点, 则从另一未被访问过的顶点出发重复上述过程, 直到图中所有顶点都被访问过时, 搜索结束.

(1) 存储结点v在搜索过程的状态

- $v.c$ =white: 表示一个结点没有被遍历过.
- $v.c$ =gray: 表示一个结点已经被遍历到, 但是对于它的遍历没有结束, 即该结点还有若干邻居结点尚未遍历.
- $v.c$ =black: 表示一个结点的所有邻居结点已经被遍历完成, 其自身的遍历也已经结束.

(2) 变量$v.p$记录在**DFS**树中结点v 的双亲结点.

(3) 结点v被遍历的时间

- 结点的第一次被**发现时间**$v.d$
- 结点遍历结束的**完成时间**$v.f$

深度优先搜索算法(1/2)

算法2.1 DFS(*G*)

```
for each vertex u ∈ V
        u.c ← white
        u.p ← NIL
time ← 0            //time 是一个全局变量, 用来计算时间戳.
for each vertex u ∈ V
        if u.c=white then
            DFS-visit(G, u)
```

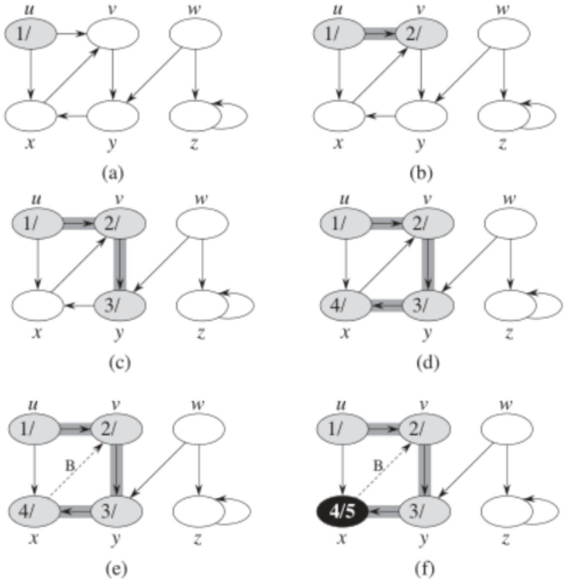
深度优先搜索算法(2/2)

算法2.2 **DFS-visit**(G, u)

```
time ← time + 1                      // white vertex u has just been discovered
u.d ← time
u.c ← gray
for each edge (u, v) ∈ E             // explore edge (u, v)
        if v.c=white then
                v.p ← u
                DFS-visit(G, v)
u.c ← black                          // blacken u; it is finished
time ← time + 1
u.f ← time
```

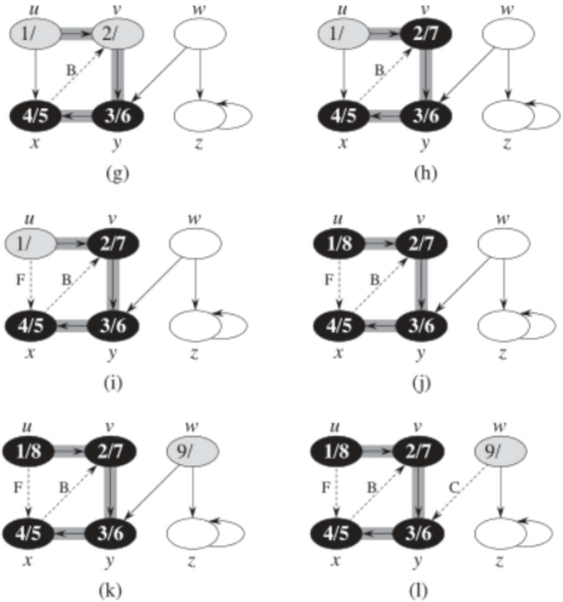
图的存储结构为链接表时, **DFS**算法的时间复杂性: $O(|V| + |E|)$.

实例(1/3)


u
v
w
1/
x
y
z
(a)
u
v
w
1/
2/
x
y
z
(b)
u
v
w
1/
2/
3/
x
y
z
(c)
u
v
w
1/
2/
4/
3/
x
y
z
(d)
u
v
w
1/
2/
B
4/
3/
x
y
z
(e)
u
v
w
1/
2/
B
4/5
3/
x
y
z
(f)

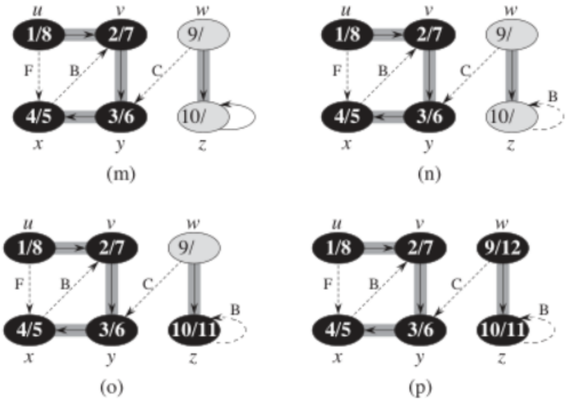
实例(2/3)

图搜索算法



u
v
w
1/
2/
B
4/5
3/6
x
y
z
(g)
u
v
w
1/
2/7
B
4/5
3/6
x
y
z
(h)
u
v
w
1/
2/7
F
B
4/5
3/6
x
y
z
(i)
u
v
w
1/8
2/7
F
B
4/5
3/6
x
y
z
(j)
u
v
w
1/8
2/7
9/
F
B
4/5
3/6
x
y
z
(k)
u
v
w
1/8
2/7
9/
F
B
C
4/5
3/6
x
y
z
(l)

实例(3/3)



DFS算法的性质(1/4)

命题

(1) $u = v.p$当且仅当***DFS-visit***(G, v)在算法对结点u的邻接链表进行搜索时被调用.

(2) 结点v是**深度优先森林**里结点u 的后代当且仅当结点v在结点u为灰色的时间段里被发现.

DFS算法的性质(2/4)

定理 (Parenthesis theorem)

在对图G进行的任意**深度优先搜索**中，对于任意的结点u和v来说，下面三种情况只有一种成立:

(1) 区间$[u.d,\ u.f]$和区间$[v.d,\ v.f]$互不相交，在**深度优先森林中**，结点u不是结点v的后代，结点v不是结点u的后代.

(2) 区间$[u.d,\ u.f]$完全包含在区间$[v.d,\ v.f]$，在**深度优先树**中，结点u是结点v的后代.

(3) 区间$[v.d,\ v.f]$完全包含在区间$[u.d,\ u.f]$，在**深度优先树**中，结点v是结点u的后代.

证明 不失一般性，设$u.d<v.d$. 根据$v.d<u.f$是否成立分两种情形讨论:

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

DFS算法的性质(3/4)

推论

在图G的深度优先森林中，结点v是结点u的真后代当且仅当$u.d < v.d < v.f < u.f$成立.

DFS算法的性质(4/4)

定理 (White-path theorem)

在有向或无向图G的**深度优先森林**中，结点v是结点u的真后代当且仅当在发现结点u的时间$u.d$，存在一条从结点u到结点v的全部由白色结点所构成的路径.

证明 ($\Rightarrow$): 考察深度优先森林中从结点u到结点v的路径和根据括号化定理.

($\Leftarrow$): 假定在时刻$u.d$时存在一条从结点u 到结点v的全部由白色结点组成的路径. 我们采用**数学归纳法**来证明，对白色路径的长度k作归纳.

初始情况, $k=1$时:

接下来我们假设长度等于k时，结点v是结点u的真后代.

当长度为$k+1$时:

边的分类(1/2)

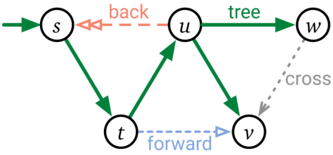
对于在图G上运行**DFS算法**所生成的**深度优先森林**，定义以下边的类型:

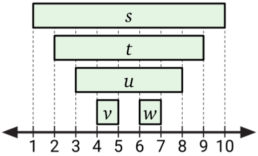
(1) **树边**(Tree edges): 为深度优先森林中的边.

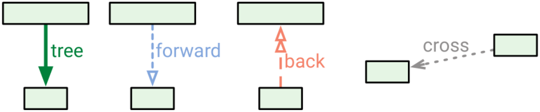
(2) **后向边**(Back edges): 后向边(u, v)是将结点u连接到其在深度优先树一个祖先结点v的边. 自循环被认为是后向边.

(3) **前向边**(Forward edges): 前向边(u, v)是将结点u连接到其在深度优先树一个后代结点v的边.

(4) **横向边**(Cross edges): 指其他所有的边.







边的分类(2/2)

当第一次探索边(u, v)时:

(1) v.c=white: 边(u, v)是树边.

(2) v.c=gray: 边(u, v)是后向边.

(3) v.c=black: 边(u, v)是前向边或横向边.

定理

在对无向图G进行深度优先搜索时, 每条边要么是树边, 要么是后向边.

问题2.1

(1) 判断有向图G中是否存在由结点v到结点u的路径.

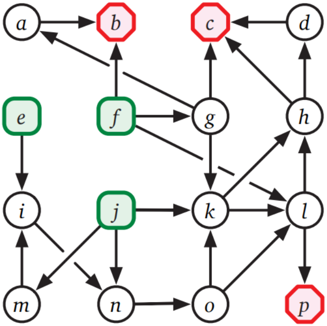
(2) 判断一个无向图是否是连通图. 若不是连通的, 输出图中连通分量的个数.

目录
图的基础知识
深度优先搜索
有向图DFS算法应用
有向无环图
拓扑排序
强连通分量

有向无环图的定义

无回路的有向图称为**有向无环图**(directed acyclic graph, DAG), 它的应用包括:

- C++类或Java接口之间的继承.
- 学位课程的先行课.
- 一个项目中有一定约束的任务间的调度.



DAG的特征

定理

有向图G是无环的当且仅当对其进行**深度优先搜索**不产生后向边.

证明 ($\Rightarrow$):用反证法.假设对图G进行**DFS**产生了一条后向边(u, v),

在**深度优先森林**中,结点v 是结点u的祖先.因此,图G包含一条从v到u的路径,该路径与后向边(u, v)一起形成了一个回路.

($\Leftarrow$):用反证法.假设图G包含一个回路c. 以下证明**DFS**将产生一条后向边.设结点v是环路c上第一个被发现的结点, 设(u, v)是回路c中结点v前面的一条边. 在时刻$v.d$, 环路c中结点形成一条从结点v到u的全白色结点路径. 根据白色路径定理, 结点u将在**深度优先森林**中成为结点v 的后代. 因此(u, v)是一条后向边.

问题2.2

(1) 设计算法判断一个有向图是否是DAG?

(2) 在DAG中, 至少存在一个源点和一个汇点. 为什么?

拓扑排序的定义

定义

如果为有向图G的每个结点$v_1, v_2, ..., v_n$分配一个序号$\tau_1, \tau_2, ..., \tau_n$, 满足:

- 所有序号为正整数1到n的某个排列.
- 对任意有向边(v_i, v_j), 满足$\tau_i<\tau_j$.

则$\tau_1, \tau_2, ..., \tau_n$为图$G$中结点集的一个**拓扑排序**(*topological sort*). 如果要求对任意的有向边(v_i, v_j), 满足$\tau_i > \tau_j$, 则所得结果称为一个“**逆拓扑排序**”.

定理

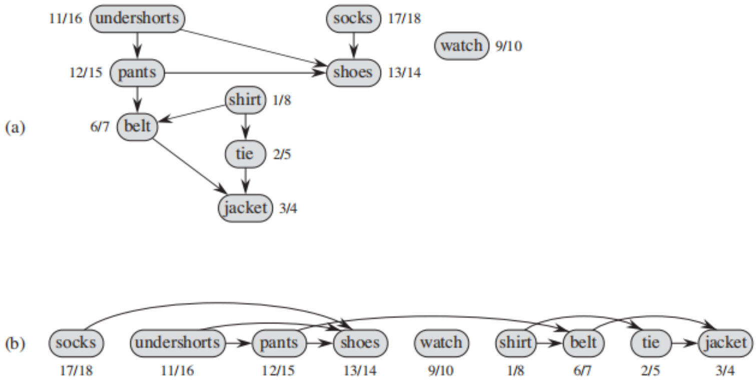
一个有向图具有拓扑排序当且仅当它是无环图.

算法2.3 **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

拓扑排序算法的时间复杂度: $O(|V| + |E|)$.

拓扑排序算法的实例


(a)
11/16
undershorts
socks
17/18
watch
9/10
12/15
pants
shoes
13/14
shirt
1/8
6/7
belt
tie
2/5
jacket
3/4
(b)
socks
undershorts
pants
shoes
watch
shirt
belt
tie
jacket
17/18
11/16
12/15
13/14
9/10
1/8
6/7
2/5
3/4

拓扑排序算法的正确性

定理

拓扑排序算法***Topological-Sort***生成的是***DAG***的拓扑排序.

证明 假设在**DAG** G上运行**DFS算法**来计算结点的完成时间. 我们只需要证明,

$$\text{对于任意两个不同的结点} u, v \text{和} (u, v) \in E, \text{则} v.f < u.f.$$

考虑**DFS算法**所搜索的任意一条边(u, v). 当这条边被搜索时, 则:

- 结点v不可能是灰色.
- 若结点v是黑色: $v.f < u.f$.
- 若结点v是白色: v将成为结点u的后代, 因此$v.f < u.f$.

第二种拓扑排序算法

(1) 在有向图中选一个没有直接前驱的结点, 并输出之;

(2) 从图中删去该结点, 同时删去所有它发出的有向边;

(3) 重复以上2、3 步, 直到:

全部顶点均已输出, 拓扑有序序列形成, 拓扑排序完成;

或图中还有未输出的顶点, 但已跳出处理循环. 这说明图中还剩下一些顶点, 它们都有直接前驱, 再也找不到没有前驱的顶点了. 这时有向图中必定存在有向环.

问题2.3

将n个捣蛋的小孩面朝前排成一队. 输入m条信息“i恨j”. 如果i恨j, 则i不能排在j后面的某个位置, 否则i会向j扔东西. 请设计一个算法在$O(m+n)$时间内将小孩排成一队, 或者判断不存在符合条件的排队方法.

强连通分量定义

定义

(1) 称一个有向图中的两个结点是**强连通**的，如果它们是互相可达.

(2) 称一个有向图是**强连通**的，如果其任意两个结点之间互相可达.

(3) 一个极大强连通子图称为有向图的**强连通分量**(*strongly connected component*, ***SCC***).

注

(1) *A directed graph G is strongly connected if and only if G has exactly one strongly connected component.*

(2) *G is a DAG if and only if every strongly connected component of G consists of a single vertex.*

强连通分量的基本性质

引理

(1) 结点之间的强连通关系是等价关系.

(2) 设C, C'为有向图G的两个不同的强连通分量, 且设$u, v \in C$, $u', v' \in C'$. 如果图中包含一条从结点u到结点u'的路径, 那么图不可能包含一条从结点v'到结点v的路径.

分量图

设图有强连通分量$C_1, C_2, \ldots, C_k$, 图G的分量图

$$G^{SCC}=(V^{SCC}, E^{SCC})$$

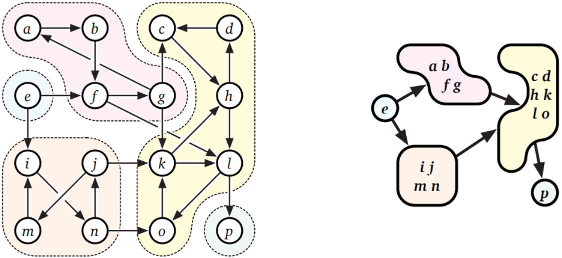
定义如下:

- $V^{SCC}=\{v_1, v_2, \ldots, v_k\}$. 对于图$G$的每个强连通分量$C_i$来说, V^{SCC} 包含代表该分量的结点v_i.
- 如果$x \in C_i$, $y \in C_j$, 且$(x, y) \in E$, 则$(v_i, v_j) \in E^{SCC}$.

引理

G^{SCC}是一个有向无环图.

分量图：实例



图的转置

设$G=(V, E)$, 定义$G^T = (V, E^T)$, 其中$E^T=\{(v, u) : (u, v) \in E\}$.
则:

(1) 给点图G的邻接链表, 创建G^T的时间为$O(|V| + |E|)$.

(2) 图G和图G^T的强连通分量完全相同.

强连通分量算法

算法2.4 **Strongly-Connected-Components**(G)

1 call **DFS**(G) to compute finishing time $u.f$ for each vertex u

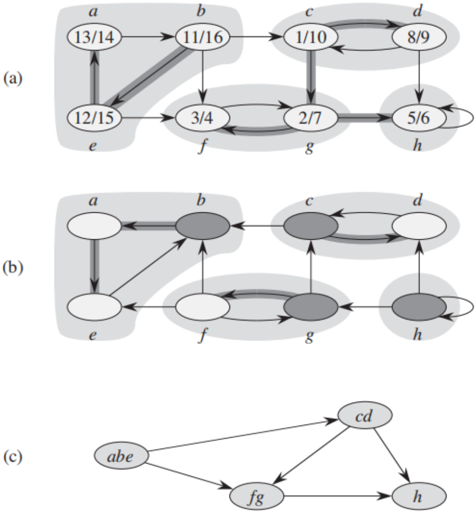
2 compute G^T

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

算法的时间复杂度: $O(|V|+|E|)$.

强连通分量：实例


(a)
a
b
c
d
13/14
11/16
1/10
8/9
12/15
3/4
2/7
5/6
e
f
g
h
(b)
a
b
c
d
e
f
g
h
(c)
abe
cd
fg
h

强连通分量算法的正确性(1/3)

给定结点集合$U\subseteq V$, 定义

$$d(U)=\min_{u\in U} u.d,\ f(U)=\max_{u\in U} u.f.$$

引理

设C和C'为有向图G的两个不同的强连通分量. 如果存在一条边$(u, v)\in E$, 这里$u\in C$, $v\in C'$, 则$f(C)>f(C')$.

证明 根据**DFS算法**中最早发现的结点在哪个强连通分量里面分为两种情形进行考虑.

(1) $d(C)<d(C')$.

(2) $d(C)>d(C')$.

强连通分量算法的正确性(2/3)

定理

算法***Strongly-Connected-Components***正确计算出有向图G的强连通分量.

证明 以算法第3行对图G^T进行深度优先搜索时所发现的深度优先树的棵数来进行归纳.

归纳假设是:

算法第3行所生成的前面k棵树都是强连通分量.

归纳证明的**初始情况**是$k=0$, 归纳假设显然成立.

在**归纳步**, 假设算法第3行所生成的前k棵树都是强连通分量, 现在考虑第$(k+1)$棵树. 设该树的根结点为u, 结点u处于强连通分量C中.

根据算法第3行选择深度优先搜索根结点的方式, 对于任意除C以外, 且尚未被访问的强连通分量C'来说, 有

$$u.f=f(C)>f(C').$$

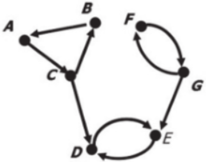
强连通分量算法的正确性(3/3)

根据**归纳假设**, 在搜索算法访问结点u的时刻, C中的所有结点都是白色. 根据**白色路径定理**, C中的其他所有结点都是结点u在深度优先树中的后代. 另外, 除C以外的强连通分量中的结点不可能在对G^T进行深度优先搜索成为结点u的后代.

问题2.4

(1) 如果在图G中加入一条新的边, G中的SCC的数量会发生什么么的变化?

(2) 有一群牛, 总数为n. 其中奶牛A认为奶牛B备受注目, 而奶牛B也可能认为奶牛C备受注目. 奶牛们的这种"认为"是单向可传递的, 就是说如果奶牛A认为奶牛B备受注目, 但奶牛B 不一定会认为奶牛A备受注目. 而当奶牛A认为奶牛B备受注目, 且奶牛B认为C备受注目时, A一定会认为C备受注目. 现在给出m对这样的"认为...备受注目"的关系对, 问有多少只奶牛被除其本身以外的所有奶牛关注.



作业

编程实现课件中的DFS算法和拓扑排序算法.

Thank you!