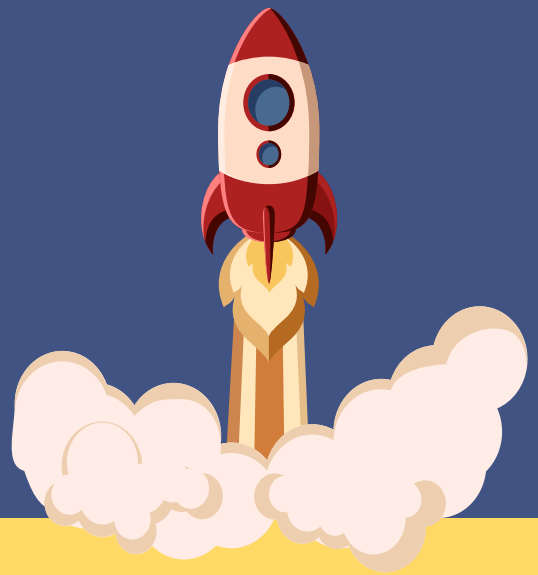


DFS序 基础技巧

NKOJ4263 1248 3886

一. 什么是DFS序？

DFS序的概念

”所谓DFS序，就是DFS整棵树依次访问到的结点组成的序列。”

”DFS序有一个很强的性质：一颗子树的所有节点在DFS序内是连续的一段，利用这个性质我们可以解决很多问题。“

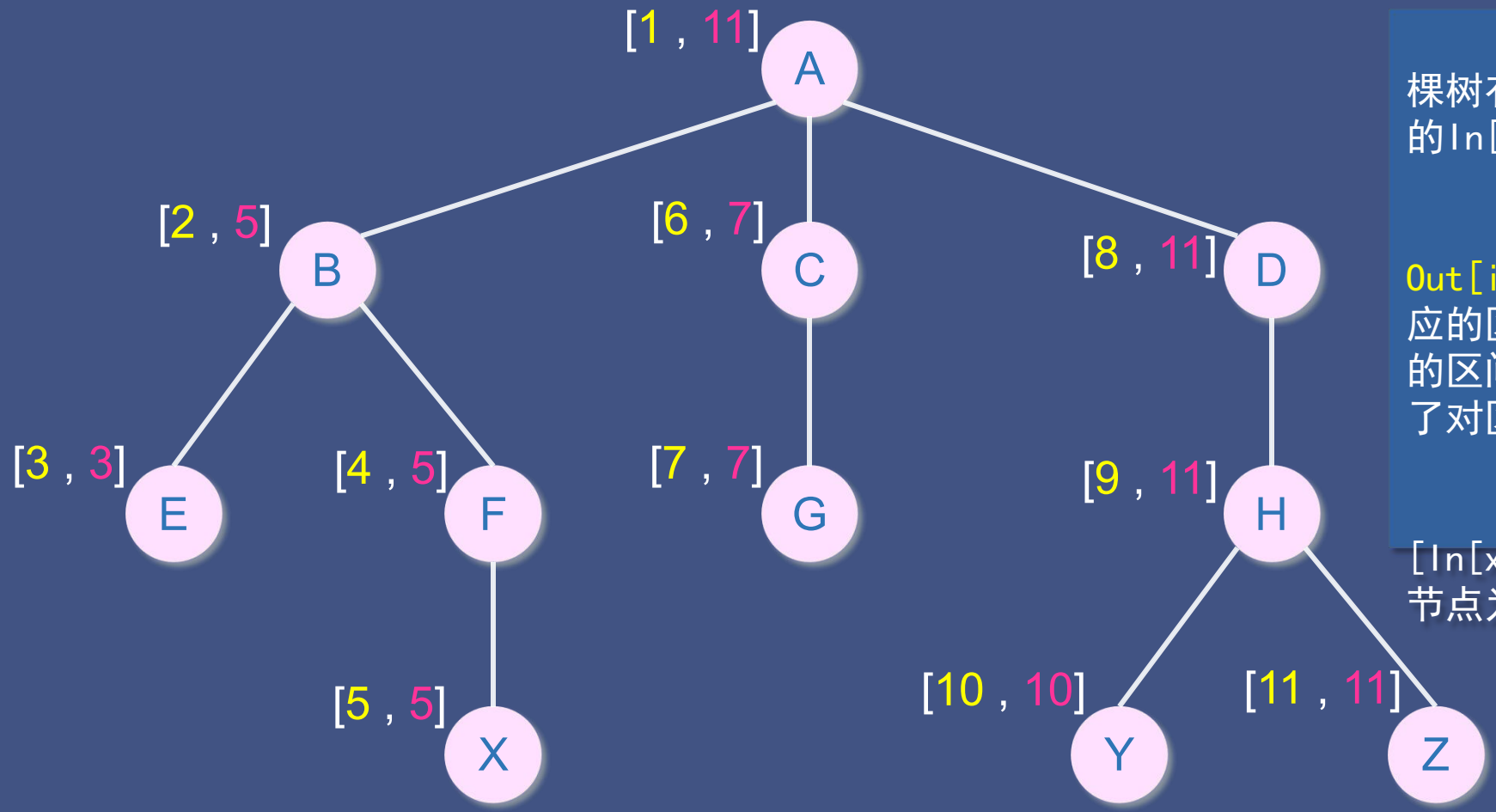
——徐昊然

所谓DFS序是指在一棵树上进行DFS，每一个节点记录两个时间戳In[i]和Out[i]，一个是dfs前序遍历进入该节点的时间戳In[i]，另一个是dfs离开该节点的时间Out[i]。

假设现在有一个计数器Cnt记录当前已经访问了的节点个数，In[i]表示的就是第一次访问到这个节点的时候计数器的值，Out[i]表示以这个节点为根的子树访问完之后计数器的值。

DFS序的概念

下图[ ]中的数字就是这棵树的DFS序，黄色数字代表In[ ]，红色数字代表Out[ ]。



我们观察，DFS序的In[ ]值使得这棵树有了区间性质，一棵子树的所有节点的In[ ]在DFS序内是连续的一段。

如果我们把节点i的[In[i] , Out[i]]看做一个区间，那么每个节点对应的区间正好“管辖”了它子树所有节点的区间，那么对点或子树的操作就转化为了对区间的操作。

比如若In[y]位于区间[In[x],Out[x]]，那么y号节点一定在x号节点为根的子树中。

DFS序的概念

```
void Dfs(int x,int Fa)
{
        In[x]=++Cnt;
        Dep[x]=Dep[Fa]+1;
        for(int i=Last[x];  i; i=Next[i])
        {
                int y=End[i];
                if(y!=Fa) Dfs(y,x);
        }
        Out[x]=Cnt;
}
```

二. DFS序的应用

NKOJ4263员工等级

何老板的公司有n名员工，编号1到n。除了何老板以外，每个员工都有且仅有一名直接上司。每个员工可能有0个或多个直接下属。

公司采取等级制度，等级越大的员工工资越低。何老板的工资最高，他的等级为1。每个员工的等级数都比他的直接上司的等级数大1。

何老板向你提问，i号员工管辖的部门中，等级为k的有多少个人？

i号员工管辖的部门包括i和i所有的直接或间接下属。

何老板向你提了m次问题，你要快速回答出所有的提问。

1<=n<=50000　　1<=m<=30000

NKOJ4263员工等级

1. 从树的根节点进行DFS，对于每个节点记录两个信息，一个是DFS进入该节点的时间戳In[i]，另一个是dfs离开该节点的时间戳Ou[i]。

2. 在DFS过程中，将树的所有节点按深度保存起来，每个深度的节点都单独用一个线性结构保存(比如vector)，每个深度的节点相对顺序要和DFS的前序遍历一致。

3. 对于每次查询，求节点p在深度k的所有子节点，只需将深度为k的线性表中，In[ ]值在In[p]和Ou[p]之间的所有节点都求出来即可。由于对于存储深度k的线性表中的所有节点，相对顺序和前序遍历的顺序一致，那么表中他们的In[ ]值也是递增的，于是可以通过二分搜索求解。

NKOJ4263员工等级

```
vector<int>P[100005];                    //P[i]记录i号点的所有儿子编号
vector<int>DepCnt[100005];               //DepCnt[i]记录深度为i的所有节点的In[ ]值
```

```
for(i=1;i<n;i++)                         //读图，建树
{
      scanf("%d%d",&x,&y);
      P[x].push_back(y);
}
```

```
void Dfs(int x,int Fa)                          //深搜，记录DFS序
{
      In[x]=++Cnt;
      Dep[x]=Dep[Fa]+1;
      DepCnt[Dep[x]].push_back(In[x]);    //记录深度为Dep[x]的节点的In[]值
      for(int i=0;i<P[x].size();i++)Dfs(P[x][i],x);
      Ou[x]=Cnt;
}
```

NKOJ4263员工等级

```
vector<int>::iterator Beg,End;
for(i=1;i<=m;i++)
{
	scanf("%d%d",&x,&y);
	Beg=lower_bound(DepCnt[y].begin(),DepCnt[y].end(),In[x]);
	End=upper_bound(DepCnt[y].begin(),DepCnt[y].end(),Ou[x]);
	ans=End-Beg;
	printf("%d\n",ans);
}
//Beg 在记录深度为y的表中，二分查询深度In[]值>=In[x]的起始位置
//End 在记录深度为y的表中，二分查询深度In[]值<=Ou[x]的结束位置
//End-Beg就是深度为y的表中，In[]值在[In[x],Ou[x]]区间中节点的数量
```

NKOJ1248 慢慢走

每天Farmer John的N头奶牛(1编号1…N)从粮仓走向他的自己的牧场。牧场构成了一棵树，粮仓在1号牧场。恰好有N-1条道路直接连接着牧场，使得牧场之间都恰好有一条路径相连。第i条路连接着Ai，Bi。

奶牛们每人有一个私人牧场Pi。粮仓的门每次只能让一只奶牛离开。耐心的奶牛们会等到他们的前面的朋友们到达了自己的私人牧场后才离开。首先奶牛1离开，前往P1；然后是奶牛2，以此类推。

当奶牛i走向牧场Pi时候，他可能会经过正在吃草的同伴旁。当路过已经有奶牛的牧场时，奶牛i会放慢自己的速度，防止打扰他的朋友。

Farmer John想知道，奶牛们总共要放慢多少次脚步。

1 <= N <= 100000

NKOJ1248 慢慢走

一. 预处理

1. 按前序遍历的顺序求出DFS序，即求出每个节点的In[i]和Ou[i];

2. 对于数组In[ ]，实际是数字1到n由小到大的构成的一个数列。在这个数列中，树中i号点管辖的范围是[In[i],Ou[i]]，即In[ ]值在这个区间中的点都属于i为根的子树。

3. 我们建立一棵线段树来维护上面的数列，设线段树中节点p代表区间[L,R]，我们维护一个域Cover，记录p号点表示的区间被完全覆盖的次数;

二. 操作

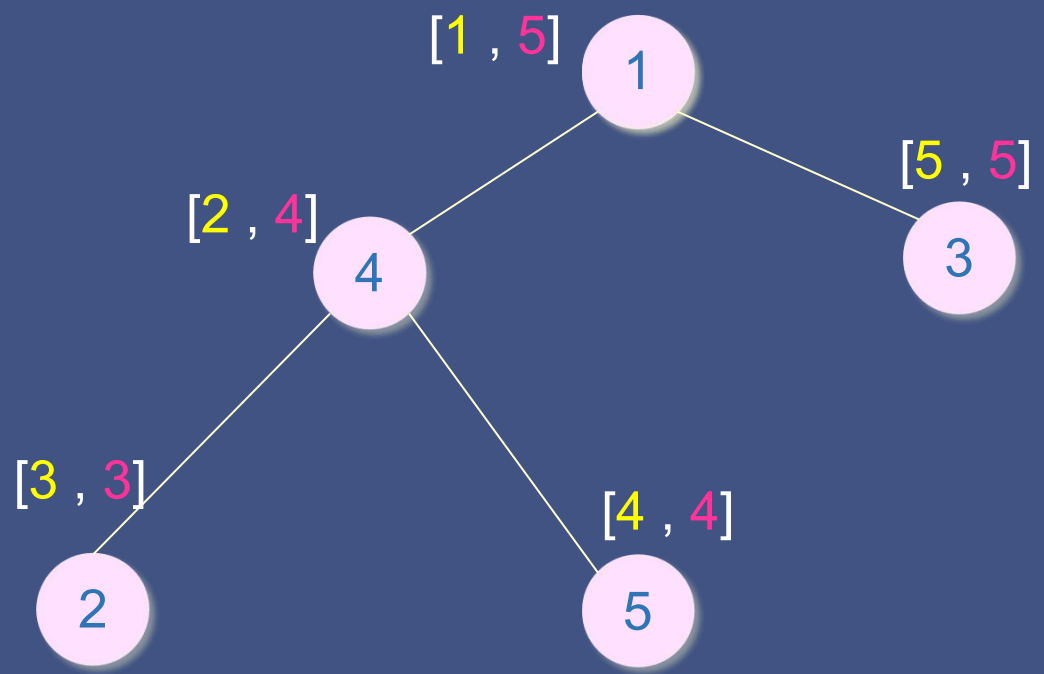
1. 对于第i号牛，设它要去的点是j号点。而j号点管辖的范围是[In[j],Ou[j]]。也就是说，后面出场的牛中，只要有牛要去的点在这个区间以内，都会打扰到i号牛吃草。

于是我们把线段树中被区间[In[j],Ou[j]]覆盖的点的Cover值+1;

2. 查询i号牛放慢的次数：i号牛要去j号点，对应In[j]，我们在线段树中查询包含In[j]的区间被Cover的总次数即可;

NKOJ1248 慢慢走

下图[ ]中的数字就是这棵树的DFS序，黄色数字代表In[ ]，红色数字代表Out[ ]。



奶牛和对应目的地

1	4
2	2
3	1
4	5
5	3

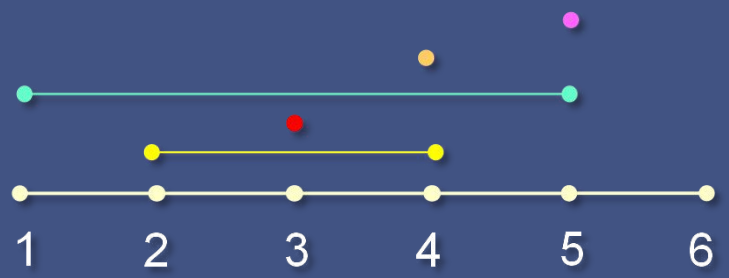
目标In[]=2　答案0　覆盖[2, 4]

目标In[]=3　答案1　覆盖[3, 3]

目标In[]=1　答案0　覆盖[1, 5]

目标In[]=4　答案2　覆盖[4, 4]

目标In[]=5　答案1　覆盖[5, 5]



NKOJ3886 大都市

有编号1..n的n个村庄，某些村庄之间有一些双向的土路。从每个村庄都恰好有一条路径到达村庄1（即比特堡）。对于每个村庄，它到比特堡的路径恰好只经过编号比它的编号小的村庄。

对于所有道路而言，它们都不在除村庄以外的其他地点相遇。随着时间的推移，越来越多的土路被改造成了公路。

Blue Mary想起了在土路改造期间她送信的经历。她从比特堡出发，需要去某个村庄，并且在两次送信经历的间隔期间,有某些土路被改造成了公路.

现在Blue Mary需要你的帮助：计算出每次送信她需要走过的土路数目。

输入格式:

第一行是一个数n(1 < = n < = 250000).

以下n-1行，每行两个整数a，b 表示a,b村庄有路相连。

以下一行包含一个整数m（1 < = m < = 250000），表示Blue Mary曾经在改造期间送过m次信。

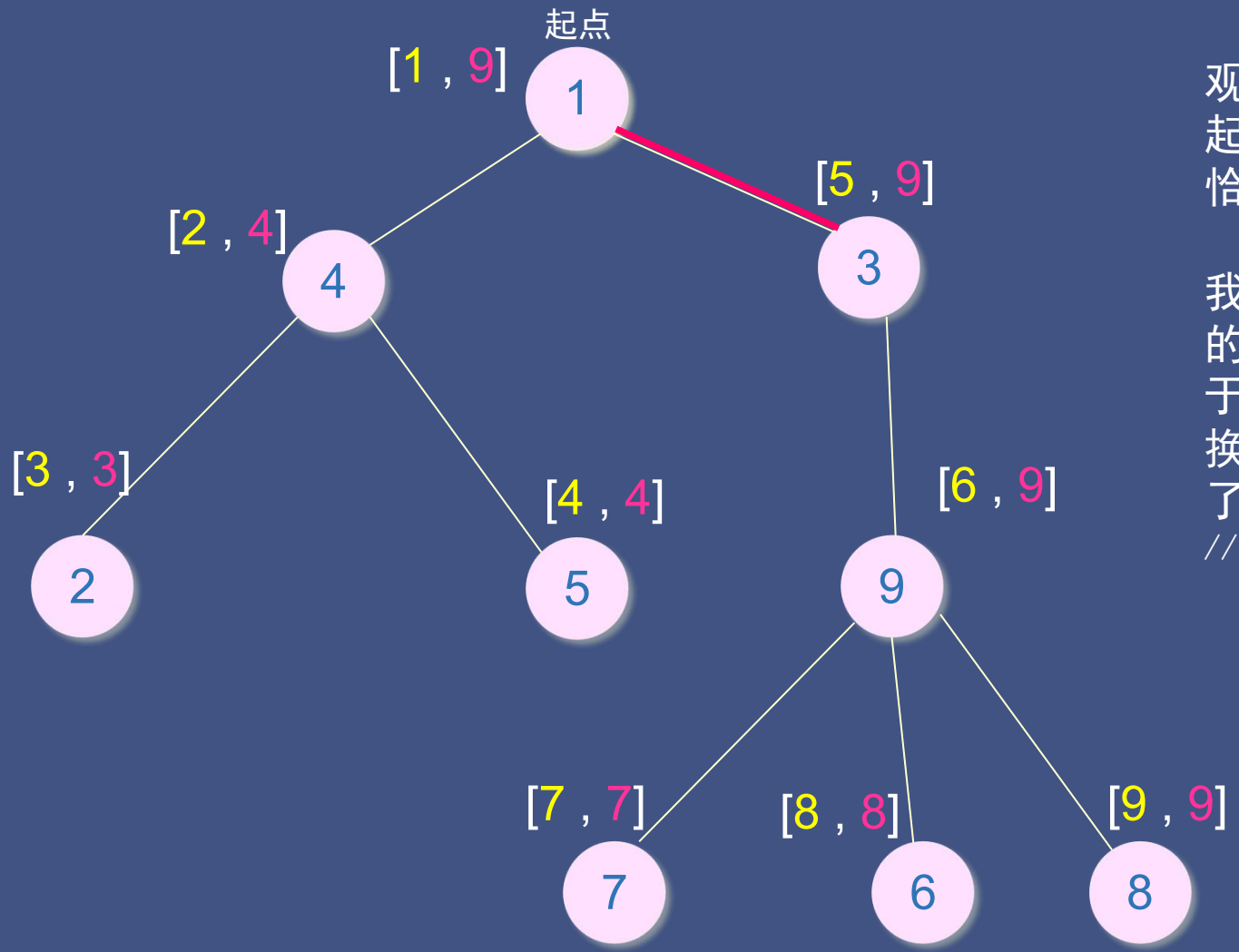
以下n+m-1行，每行有两种格式的若干信息，表示按时间先后发生过的n+m-1次事件:

若这行为 A a b，表示道路a b被改成了公路。

若这行为 W a，则表示Blue Mary曾经从比特堡送信到村庄a。

NKOJ3886 大都市

下图[ ]中的数字就是这棵树的DFS序，黄色数字代表In[ ]，红色数字代表Out[ ]。



观察：当我们把左图中1到3这条边改成公路以后，起点到哪些点经过的土路条数会发生变化呢？
恰好是In[ ]值为[5,9]的点。

我们相当于有条线段将区间[5,9]覆盖，这段区间的整体权值-1。
于是我们相当于把树形结构通过DFS序线性化，转换成用线段树或差分数组来维护的区间覆盖问题了。
//树状数组+差分的知识请查阅讲义<差分数组与树状数组>

NKOJ3886 大都市

树状数组+差分：

将边权下放到点，边(x,y)的权值记录到点y上，y是x的儿子。

对于节点权值，用树状数组来维护它的差分前缀和。

比如将边(x,y)修改成公路。则区间[In(y),Out[y]]中每个点到根经过的公路数量都增加1，则相当于区间[In(y),Out[y]] 整体增加1：

```
Modify( In(y) , 1 );
Modify( Out(y)+1 , -1 );
```

我们要询问点x到根路径上经过的土路数量：

```
cout<<Dep[x]-getSum(In(x))-1;
```

Dep[x]表示点x的深度，getSum(In(x))统计根到x路径上公路的数量

```
int getSum(int x)
{
      int Sum= 0;
      for ( int k = x; k > 0; k -= lowbit(k) )  Sum += C[k];
      return Sum;
}
```

```
void modify(int x,int d)
{
    for( int i = x; i<=n; i+=lowbit(i) )C[i] += d;
}
```

NKOJ4331 树上操作

有一棵点数为 N 的树，以点 1 为根，且树点有边权。然后有 M 个操作，分为三种：
操作 1 ：把某个节点 x 的点权增加 a 。
操作 2 ：把某个节点 x 为根的子树中所有点的点权都增加 a 。
操作 3 ：询问某个节点 x 到根的路径中所有点的点权和。

N,M<=100000 ，且所有输入数据的绝对值都不会超过 10^6

树状数组 差分

三. DFS序的7个故事

见阅读材料

三. DFS序的习题

BZOJ 4034, NKOJ5658

DFS序入门，完！

以梦为码 心之所往