待打印的

HDU 6035 Colorful Tree（虚树）

原创 2017年07月26日 10:11:18 标签：ACM / 图论 / 多校联合训练赛 / dfs / 虚树 664

Colorful Tree

Time Limit: 6000/3000 MS (Java/Others) Memory Limit: 131072/131072 K (Java/Others)
Total Submission(s): 443 Accepted Submission(s): 155

Problem Description

There is a tree with n nodes, each of which has a type of color represented by an integer, where the color of node i is c_i.

The path between each two different nodes is unique, of which we define the value as the number of different colors appearing in it.

Calculate the sum of values of all paths on the tree that has $\frac{n(n-1)}{2}$ paths in total.

Input

The input contains multiple test cases.

For each test case, the first line contains one positive integers n, indicating the number of node. $(2 \le n \le 200000)$

Next line contains n integers where the i-th integer represents c_i, the color of node i. $(1 \le c_i \le n)$

Each of the next $n-1$ lines contains two positive integers x, y $(1 \le x, y \le n, x \ne y)$ meaning an edge between node x and node y.

It is guaranteed that these edges form a tree.

Output

For each test case, output "**Case #x: y**" in one line (without quotes), where x indicates the case number starting from 1 and y denotes the answer of corresponding case.

Sample Input

```
3
1 2 1
1 2
2 3
6
1 2 1 3 2 1
1 2
1 3
2 4
2 5
3 6
```

Sample Output

```
Case #1: 6
Case #2: 29
```

Source

2017 Multi-University Training Contest - Team 1

Recommend

liuyiding

题目大意：

给你一棵n(2<=n<=200000)个节点的每个节点有一个颜色的树，求树上所有路径经过的颜色数的和。

解题思路：

首先分开考虑每种颜色，也就是拆成n棵只存在当前枚举的颜色，其他颜色视为无色的树。那么就变成了求所有路径中经过这种颜色的路径数，这样还是不好求，再转化一下，如果我们知道所有路径中没有经过这种颜色的路径数也可以算出答案。

对于求没有进过这个颜色的路径数，我们可以把这棵树上这种颜色的点删掉，就得到了一些联通块，每个联通块中的全部路径都是要求的。于是就可以对每棵树dfs一遍，得到结果，时间复杂度是O(N*N)，显然会TLE。

继续优化，可以发现，在dfs的过程中，我们只关心和当前节点颜色相同的节点，于是我们就可以考虑使用虚树，也就是不用真的把原树拆开，我们只需要同时维护所有颜色的信息，在dfs到一个节点是只使用和更新当前颜色的信息即可。这样就可一遍dfs解决所有问题，时间复杂度就减为O(N)，十分完美。

AC代码：

```cpp
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <cmath>
#include <ctime>
#include <vector>
#include <queue>
#include <stack>
#include <deque>
#include <string>
#include <map>
#include <set>
#include <list>
using namespace std;
#define INF 0x3f3f3f3f
#define LL long long
#define fi first
#define se second
#define mem(a,b) memset((a),(b),sizeof(a))

const int MAXN=200000+3;
int N, color[MAXN];
vector<int> G[MAXN];
LL ans;
int top[MAXN];//此时颜色为i的联通块的上界
int not_in[MAXN];//以i为根节点的子树中，不在i所在的联通块中节点的个数
int not_in_root[MAXN];//以颜色i为分界线，不在根节点所在的联通块中的节点个数

void init()//初始化
{
    for(int i=0;i<=N;++i)
    {
        not_in_root[i]=0;
        G[i].clear();
    }
}

inline LL get_path_num(LL num)//得到一个大小为num的联通块中有多少条路径
{
    return num*(num-1)>>1;
}

int dfs(int u, int fa)
{
    int now_top=top[color[u]];//以u的颜色为边界的u上面的联通块的上界
```

```
    int now_top=top[color[u]];//以u的颜色为边界的上面的联通块的上界
    top[color[u]]=u;//更新当前颜色的上界
    int u_num=1;//以u为根节点的子树的节点数
    for(int i=0;i<G[u].size();++i)
    {
        int v=G[u][i];
        if(v==fa)
            continue;
        not_in[u]=0;
        int v_num=dfs(v, u);//以v为根节点的子树的节点数
        ans-=get_path_num(v_num-not_in[u]);
        u_num+=v_num;
    }
    (now_top?not_in[now_top]:not_in_root[color[u]])+=u_num;//更新不在当前联通块中的节点数
    top[color[u]]=now_top;//回溯
    return u_num;
}

int main()
{
    int cas=1;
    while(~scanf("%d", &N))
    {
        init();
        for(int i=1;i<=N;++i)
            scanf("%d",&color[i]);
        for(int i=0;i<N-1;++i)
        {
            int u, v;
            scanf("%d%d", &u, &v);
            G[u].push_back(v);
            G[v].push_back(u);
        }
        ans=get_path_num(N)*N;//假设所有路径都经过所有颜色
        dfs(1, -1);
        for(int i=1;i<=N;++i)
            ans-=get_path_num(N-not_in_root[i]);//减去根节点所在的联通块
        printf("Case #%d: %lld\n",cas++,ans);
    }

    return 0;
}
```

Problem Description

sd0061, the legend of Beihang University ACM-ICPC Team, retired last year leaving a group of noobs. Noobs have no idea how to deal with m coming contests. **sd0061** has left a set of hints for them.

There are n noobs in the team, the i-th of which has a rating ai. **sd0061** prepares one hint for each contest. The hint for the j-th contest is a number bj, which means that the noob with the (bj+1)-th lowest rating is ordained by **sd0061** for the j-th contest.

The coach asks **constroy** to make a list of contestants. **constroy** looks into these hints and finds out: bi+bj≤bk is satisfied if bi≠bj, bi<bk and bj<bk.

Now, you are in charge of making the list for **constroy**.

Input

There are multiple test cases (about 10).

For each test case:

The first line contains five integers n,m,A,B,C. (1≤n≤107,1≤m≤100)

The second line contains m integers, the i-th of which is the number bi of the i-th hint. (0≤bi<n)

The n noobs' ratings are obtained by calling following function n times, the i-th result of which is ai.

```
unsigned x = A, y = B, z = C;
unsigned rng61() {
    unsigned t;
    x ^= x << 16;
    x ^= x >> 5;
```

```
    x ^= x << 1;
    t = x;
    x = y;
    y = z;
    z = t ^ x ^ y;
    return z;
}
```

Output
For each test case, output "**Case #x: y1 y2 ⋯ ym**" in one line (without quotes), where x indicates the case number starting from 1 and yi (1≤i≤m) denotes the rating of noob for the i-th contest of corresponding case.

Sample Input
3 3 1 1 1
0 1 2
2 2 2 2 2
1 1

Sample Output
Case #1: 1 1 202755
Case #2: 405510 405510

题意：输入n,m,A,B,C 由ABC和题中提供的函数可以 得到n个数的数组a[n] ，然后输入m个数 表示 数组b[m]，现在对于每个b[i]输出a[]数组中的第b[i]+1小的数。题中给出b[]数组的限制bi+bj≤bk is satisfied if bi≠bj, bi<bk and bj<bk ;

思路：可以发现b[]数组不是很大，对于每个b[i]找a[]数组的第b[i]+1的数时，可以使用nth(a,a+p,a+n) 这个STL函数，进行一次o(n)的排序，使得a[p]之前的数均小于a[p]，a[p]之后的数均大于a[p]，所以a[p]即为我们要的数。在对每个b[]元素操作时，可以对b[]排序,从大到小进行计算，以为之前排序的右半部分不必再进行排序，这样会减少很多计算量。

代码如下：

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;
const int N=1e7+5;
unsigned x,y,z, A,B,C;
unsigned a[N];
struct Node
{
    int x;
    int id;
    unsigned y;
}tr[105];
bool cmp(const Node s1,const Node s2)
{
    return s1.x<s2.x;
}
bool cmp2(const Node s1,const Node s2)
{
    return s1.id<s2.id;
}

unsigned rng61() {
  unsigned t;
  x ^= x << 16;
  x ^= x >> 5;
  x ^= x << 1;
  t = x;
  x = y;
  y = z;
  z = t ^ x ^ y;
  return z;
}

int main()
{
    ///cout << "Hello world!" << endl;
```

```
    int n,m,Case=1;
    while(scanf("%d%d%u%u%u",&n,&m,&A,&B,&C)!=EOF)
    {
        x = A, y = B, z = C;
        for(int i=0;i<n;i++)  a[i]=rng61();
        printf("Case #%d:",Case++);
        for(int i=1;i<=m;i++) scanf("%d",&tr[i].x),tr[i].id=i;
        sort(tr+1,tr+m+1,cmp);

        int p=n;
        for(int i=m;i>=1;i--)
        {
            int x = tr[i].x;
            nth_element(a,a+x,a+p);
            p=x;
            tr[i].y=a[x];
        }
        sort(tr+1,tr+m+1,cmp2);
        for(int i=1;i<=m;i++) printf(" %u",tr[i].y);
        puts("");
    }
    return 0;
}
```

传送门：点这里~

题意：有n个区间，对于第i个区间[li,ri]有li<=i<=ri,

对于任意1<=L<=i<=R<=n，**当前仅当li<=L<=i<=R<=ri**时P[i]=min(P[L],P[L+1],...,P[R])

题解：

首先要理解题意：当前仅当li<=L<=i<=R<=ri时P[i]=min(P[L],P[L+1],...,P[R])

因此对于P[i]一定有**P[i]>P[li-1]且P[i]>P[ri+1]**，进一步说区间[li,ri](除了[1,n])一定被某个区间[lj,rj]包含,且j=li-1或j=ri+1

即**区间可分成[lj,j-1]和[j+1,rj]**

我们把n个区间按L升序R降序进行排序(**这样得到的区间LR正是前序遍历的区间,区间由大到小**)。得到的第1个区间一定要是[1,n](1比任何数都小)，否则不合法，输出0；设这个区间对应的是第i个数，因此区间可再分为[1,i-1]和[i+1,n],看是否有这2个区间，如果没有则不合法，输出0...直到区间不可再分。

现在再来考虑方法数：设f(i)为区间i内的方法数，u,v分别为左右子区间，i内一共有ri-li+1个数，除去中间一个，要从中选i-li个数放入左区间，剩下的放入右区间，因此答案为：f(i)=f(u)*f(v)*C(ri-li,i-li)

```
[cpp] view plain  copy  print  ?
01. #include<bits/stdc++.h>
02. using namespace std;
03. typedef long long LL;
04. namespace IO {
05.     const int MX = 4e7; //1e7占用内存11000kb
06.     char buf[MX]; int c, sz;
07.     void begin() {
08.         c = 0;
09.         sz = fread(buf, 1, MX, stdin);
10.     }
11.     inline bool read(int &t) {
12.         while(c < sz && buf[c] != '-' && (buf[c] < '0' || buf[c] > '9')) c++;
13.         if(c >= sz) return false;
14.         bool flag = 0; if(buf[c] == '-') flag = 1, c++;
15.         for(t = 0; c < sz && '0' <= buf[c] && buf[c] <= '9'; c++) t = t * 10 + buf[c] - '0';
16.         if(flag) t = -t;
17.         return true;
18.     }
19. }
20. const LL mod = 1e9 + 7;
21. const int MX = 1e6 + 5;
22. LL F[MX], invF[MX];
23. LL power(LL a, LL b) {
```

```
    LL ret = 1;
    while (b) {
        if (b & 1) ret = ret * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ret;
}
void init() {
    F[0] = 1;
    for (int i = 1; i < MX; i++) F[i] = F[i - 1] * i % mod;
    invF[MX - 1] = power(F[MX - 1], mod - 2);
    for (int i = MX - 2; i >= 0; i--) invF[i] = invF[i + 1] * (i + 1) % mod;
}
LL C(LL n, LL m) {
    LL ret = 1;
    while (n && m) {
        LL nn = n % mod, mm = m % mod;
        if (nn < mm) return 0;
        ret = ((ret * F[nn] % mod) * invF[mm] % mod) * invF[nn - mm] % mod;
        n /= mod, m /= mod;
    }
    return ret;
}
struct node {
    int l, r, id;
    bool operator<(const node& _A)const {
        if (l != _A.l) return l < _A.l;
        return r > _A.r;
    }
} a[MX];
int n;
int mark;  //是否有区间不合法
int rear;
LL dfs(int l, int r) {
    if (mark == 0) return 0;
    if (l > r) return 1;
    if (a[rear].l != l || a[rear].r != r) {
        mark = 0;
        return 0;
    }
    node now = a[rear++];
    LL ret = C(now.r - now.l, now.id - now.l) * dfs(now.l, now.id - 1) % mod;
    ret = (ret * dfs(now.id + 1, now.r)) % mod;
    return ret;
}
int main() {
    int cas = 0;
    init();
    //freopen("in.txt", "r", stdin);
    IO::begin();
    while (IO::read(n)) {
        for (int i = 1; i <= n; i++) IO::read(a[i].l);
        for (int i = 1; i <= n; i++) IO::read(a[i].r), a[i].id = i;
        sort(a + 1, a + n + 1);
        mark = rear = 1;
        printf("Case #%d: %lld\n", ++cas, dfs(1, n));
    }
    return 0;
}
```