目录

7.22 练习题

1. White-Black Balanced Subtrees CF1676G

【问题描述】

给定一棵 n 个结点的树，根结点是 1，每个结点是黑色或白色的。
如果一棵树中黑色结点与白色结点数量相同，那么这棵树是“平衡的”。
问这棵 n 个结点的树有多少棵“平衡的”子树。

【输入格式】

多组输入，第一行为组数 t，后续分别给出每组树的节点个数、每个节点的父节点以及每个节点的颜色，

【输出格式】

输出一个整数，表示平衡子树的个数。

【样例输入】

```
3
7
1 1 2 3 3 5
WBBWWBW
2
1
BW
8
1 2 3 4 5 6 7
BWBWBWBW
```

【样例输出】

```
2
1
4
```

题目链接：https://www.luogu.com.cn/problem/CF1676G

【题目分析】

树形 dp，统计子树染色情况，令以 x 为根点子树中黑白结点的个数分别为 dp[i][0/1]。
则 dp[x][0] = Σdp[v][0] + (x is black)，dp[x][1] = Σdp[v][1] + (x is white) ，其中 v 是 x 的子节点。
如果 dp[x][0] = dp[x][1]，那么以 x 为根的子树是平衡的。

【参考代码】

```
#include<iostream>
#include<vector>
#include<string>
#include<cstring>
#include<algorithm>

using namespace std;
typedef long long LL;

const int MAXN = 4e3 + 5;
vector<int> G[MAXN];    // 存储图的邻接表
int dp[2][MAXN];        // dp 数组，用于记录某个子树中"B"和"W"的个数
int n;                  // 节点数
string str;             // 字符串
int sum;                // 统计结果
```

```
void dfs(int r) {
    if (str[r - 1] == 'B')
        dp[0][r]++;
    else
        dp[1][r]++;
    for (int i = 0; i < G[r].size(); i++) {
        dfs(G[r][i]);
        dp[0][r] += dp[0][G[r][i]];
        dp[1][r] += dp[1][G[r][i]];
    }
    if (dp[0][r] == dp[1][r])
        sum++;
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        sum = 0;
        scanf("%d", &n);
        for (int i = 2, t; i <= n; i++) {
            scanf("%d", &t);
            G[t].push_back(i);    // 构建图的邻接表
        }
        cin >> str;               // 读取字符串
        dfs(1);                   // 从根节点开始进行 DFS
        printf("%d\n", sum);      // 输出统计结果
        memset(dp[1], 0, sizeof(dp[1]));   // 清空 dp 数组
        memset(dp[0], 0, sizeof(dp[0]));
        for (int i = 0; i <= n; i++)
            G[i].clear();
        str.clear();
    }
    return 0;
}
```

2. Cthulhu CF103B

【问题描述】

给定一个 n 个点 m 条边的无向图（不一定连通），判断其是否可以表示成至少三棵有根树，且所有根通过一个简单环连接。保证给定的图无自环、无重边。

【输入格式】

n 个点 m 条边，以及其构成。

【输出格式】

判断是否满足题目条件。

【样例输入】

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

```
2 5
1 4
5 4
```

【样例输出】

```
FHTAGN!
```

题目链接：https://www.luogu.com.cn/problem/CF103B

【题目分析】

题意可以简化为，求给定图中是否有且仅有一个环，可以用 dfs 和 bfs 搜索来做，这里重点讲并查集的思路。

如果 n > m 时，无法成环；n < m 时，无论怎么连，都会形成不止一个环，所以 n!=m 时，环无法形成。

当 n=m 时用并查集维护一个连通块，如果所有的点都在一个连通块中，那么就能复合有且只有一个环。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int n, m, x, y, fa[100005];
int find(int x) {
    if(x == fa[x])//根节点
        return x;
    return fa[x] = find(fa[x]);//路径压缩
}
void unionn(int x, int y) {//合并
    int fx = find(x), fy = find(y);//各自所在的集合
    if(fx != fy)
        fa[fx] = fy;//不在同一集合就合并
    return;
}
int main() {
    cin >> n >> m;
    if(n != m) {
        cout << "NO";
        return 0;
    }
    //无法形成环
    for(int i = 1; i <= n; i++)
        fa[i] = i;
    //初始化
    for(int i = 1; i <= m; i++)
        cin >> x >> y, unionn(x, y);//合并图中的点
    for(int i = 2; i <= n; i++)
        if(find(i) != find(i - 1)) {//不在同一集合就一定不对
            cout << "NO";
            return 0;
    }
    cout << "FHTAGN!";//只有一个环
    return 0;
}
```

3. Valiant's New Map CF1731D

【问题描述】

给定一个带权值的 n×m（1<=n<=m,1<=n*m<=1e6）网格，你可以选取一块边长为 1 的正方形区域当且仅当该区域的所有权值都大于等于 1，问可以选取的最大正方形区域的边长。

【输入格式】

多组输入，给定 n 和 m 以及网格权值(1<=ai,j<=1e6)。

【输出格式】

可以选取的最大正方形区域的边长。

【样例输入】

```
4
2 2
2 3
4 5
1 3
1 2 3
2 3
4 4 3
2 1 4
5 6
1 9 4 6 5 8
10 9 5 8 11 6
24 42 32 8 11 1
23 1 9 69 13 3
13 22 60 12 14 17
```

【样例输出】

```
2
1
1
3
```

题目链接：https://www.luogu.com.cn/problem/CF1731D

【题目分析】

如果存在边长为mid 的符合条件的正方形，则一定存在边长 <mid 的符合条件的正方形，可见，此题具有单调性，考虑二分答案。

由于只知道 n×m<=1e6，所以开二维数组会 MLE，此时 vector 代替二维数组，一种思路是利用二维前缀和进行二分答案的查询，对于每次二分设一个 b 数组，若 a 数组中这个数小于 mid，b 数组对应设置为 1，反之为 0。

然后对 b 数组进行二维前缀和，枚举左上角，对 mid×mid 的各个区间进行判断是否全为 0，如果是，l=mid+1，否则 r=mid-1.

另一种思路是利用 dp 维护最大的正方形，转化为经典问题“给定一个只包含 0 和 1 的矩阵，选择一个正方形，使得正方形内只含有 1，求最大边长。”将这道题的判断条件 aij = 1 改为 aij >= mid 即可。

【参考代码】

```
#include <bits/stdc++.h>
#include <bits/stdc++.h>
using namespace std;
const int N=1e6+5;
int T,n,m,i,j,l,r,mid,ans;
vector <int> a[N],b[N];
bool check(int mid){
    int i,j;
    for (i=1;i<=n;i++)
        for (j=1;j<=m;j++)
            b[i][j]=(a[i][j]<mid);
    for (i=1;i<=n;i++)
        for (j=1;j<=m;j++)
            b[i][j]=b[i][j]+b[i-1][j]+b[i][j-1]-b[i-1][j-1];
```

```
    for (i=1;i+mid-1<=n;i++)
        for (j=1;j+mid-1<=m;j++)
            if (b[i+mid-1][j+mid-1]-b[i-1][j+mid-1]-b[i+mid-1][j-1]+b[i-1][j-1]==0)
                return 1;
    return 0;
}
int main(){
    cin>>T;
    while (T--) {
        cin>>n>>m;
        b[0].clear();
        for (j=0;j<=m;j++) b[0].push_back(0);
        for (i=1;i<=n;i++){
            a[i].clear(),a[i].push_back(0),b[i].clear(),b[i].push_back(0);
            for (j=1;j<=m;j++)
                cin>>l,a[i].push_back(l),b[i].push_back(0);
        }
        l=1,r=n;
        while (l<=r){
            mid=(l+r)>>1;
            if (check(mid)) l=(ans=mid)+1;
                else r=mid-1;
        }
        cout<<ans<<'\n';
    }
    return 0;
}
```

4. Mike and gcd problem CF798C

【问题描述】

有一个数列 a1,a2,...,an，每次操作可以将相邻的两个数 x,y 变为 x-y,x+y，求最少的操作数使得 gcd(a1,a2,...,an)>1。gcd(a1,...,an)表示最大的非负整数使得所有 ai 都能被 gcd(a1,...,an)整除。

【输入格式】

第一行一个整数 n(2<=n<=1e5)，表示序列 A 的长度。

之后一行，n 个被空格隔开的整数 a1,a2…an（1<=ai<=1e9），表示 A 中的元素

【输出格式】

如果可以使序列变为优美的，第一行输出 YES，然后第二行输出最小操作次数。

如果不可能使得序列变为优美的，在第一行输出 NO。

【样例输入】

```
2
1 1
```

【样例输出】

```
Yes
1
```

题目链接：https://www.luogu.com.cn/problem/CF798C

【题目分析】

显然有解，若原序列 gcd > 1，则不需要操作

否则操作的最小代价 -> 所有元素变为偶数

考虑如果变换相邻两数 a b
a b -> a-b a+b -> -2b 2a
因此变换两次可以把任意相邻 a b 变为偶数
而若 a b 都是奇数，只需变换一次
所以每次先找出相邻是奇数的情况 ans++
然后找相邻是奇偶的情况 ans+=2

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;

const int maxn = 100100;
int n, a[maxn];

int main() {
    while(~scanf("%d", &n)) {
        int g = 0;
        for(int i = 1; i <= n; i++) {
            scanf("%d", &a[i]);
            g = __gcd(g, a[i]);
            if(a[i] & 1) a[i] = 1;
            else a[i] = 0;
        }
        if(g != 1) {
            puts("YES\n0");
            continue;
        }
        int ret = 0;
        for(int i = 1; i < n; i++) {
            if(a[i] && a[i+1]) {
                a[i] = a[i+1] = 0;
                ret++;
            }
        }
        for(int i = 1; i <= n; i++) {
            if(a[i]) ret += 2;
        }
        printf("YES\n%d\n", ret);
    }
}
```

5. KRUMPIRKO

【问题描述】

Mr. Potato 开了两家新店卖土豆。他买了 N 袋土豆，其中第 i 袋价值为 cic，袋里有 ai 个土豆。他打算把这 N 袋土豆整袋整袋地分在两个店里。在每家店中，土豆的平均价格等于这家店里所有袋的土豆的总价比上土豆的个数。（注意是个数而不是袋数！

设 P1 为第一家店的土豆平均价格，P2 为第二家店的土豆平均价格。Mr. Potato 希望在至少有一家店里土豆袋数正好等于 L 袋的情况下，最小化 P1×P2 的值。

【输入格式】

第一行包含两个整数 N 和 L。
第二行包含 N 个整数 ai。

第三行包含 N 个整数 ci。

【输出格式】

第一行输出一个浮点数，为 P1×P2 的最小值，保留小数点后三位。

【样例输入】

```
3 1
3 2 1
1 2 3
```

【样例输出】

```
0.556
```

题目链接：https://www.luogu.com.cn/problem/P7801

【题目分析】

首先题目要求 min（P1×P2），设一家店土豆总价值为 x，总个数为 y，
可以将 P1×P2 转化为：

$$P1\times P2 = x/y \times (\Sigma ci - x)/(\Sigma ai - y) = (x \times \Sigma ci - x^2)/(y \times \Sigma ai - y^2)$$

其中，n 和 ai 都较小，所以可以开三维 dp 数组 dp[i][j][k]表示前 i 袋土豆，选择 j 袋，选取土豆总数为 k 袋情况。其中 min（x × Σci - x^2）无法直接维护。
观察式子可知这个式子为二次项系数为负，开口朝下的二次函数，所以最小值在 x 取极值时取到，因此可以同时维护式子的最大值和最小值，然后维护最优解即可。
数据会溢出 int，需要开 long long
优化：利用滚动数组维护第一维 i。

【参考代码】

```
#include<bits/stdc++.h>
#define N 105
using namespace std;
int n,l;
int f[2][N][505],g[2][N][505];
int a[N],c[N],suma,sumc;
int main()
{
    scanf("%d%d",&n,&l);
    l=min(l,n-l);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]),suma+=a[i];
    for(int i=1;i<=n;i++) scanf("%d",&c[i]),sumc+=c[i];
    memset(f,0x3f,sizeof(f)),memset(g,-0x3f,sizeof(g));
    f[0][0][0]=g[0][0][0]=0;
    for(int i=1;i<=n;i++)
    {
        for(int j=0;j<=min(i,l);j++)
        {
            for(int k=0;k<=suma;k++)
            {
                f[i&1][j][k]=f[i-1&1][j][k],g[i&1][j][k]=g[i-1&1][j][k];
                if(k>=a[i]&&j)
                {
                    f[i&1][j][k]=min(f[i&1][j][k],f[i-1&1][j-1][k-a[i]]+c[i]);
                    g[i&1][j][k]=max(g[i&1][j][k],g[i-1&1][j-1][k-a[i]]+c[i]);
                }
            }
```

```
        }
    }
    double ans=1e16;
    for(int i=1;i<=suma;i++)
    {
        if(f[n&1][l][i]<1e9)
ans=min(ans,1ll*f[n&1][l][i]*(sumc-f[n&1][l][i])*1.0/(1ll*i*(suma-i)));
        if(g[n&1][l][i]>-1e9)
ans=min(ans,1ll*g[n&1][l][i]*(sumc-g[n&1][l][i])*1.0/(1ll*i*(suma-i)));
    }
    printf("%.3lf\n",ans);
}
```

6. IOI 计数

【问题描述】

给定一个长度为 n 的字符串 S，同时进行 m 次操作：

操作 1:1 x c 表示将第 x 个字符改为 c（c 只会为 I 或 O）。

操作 2:2 l r 询问字符串 S 中有多少对三元组（i，j，k）满足：

Si = I，Sj = O，Sk = I，并且 l<=i<j<k<=r。

【输入格式】

输入第一行为两个正整数 n 和 m。

接下来一行是长度为 n 的字符串 s，接下来是 m 行操作。

【输出格式】

输出若干行：对于所有操作 2，输出查询的答案，要求每个答案之间换行。

【样例输入】

```
4 3
IOOI
2 1 4
1 1 O
2 1 2
```

【样例输出】

```
2
0
```

题目链接：https://www.luogu.com.cn/problem/P6373

【题目分析】

单点修改，区间查询，自然可以利用线段树进行维护。

设 I,O,IO,OI,IOI 为它们在这个区间内出现的次数，则：

I=left->I + right->I

O=left->O + right->O

IO=left->IO + right->IO + left->I × right->O

OI=left->OI + right->OI + left->O × right->I

IOI=left->IOI + right->IOI + left->I × right->OI +left->IO×right->I

数据会溢出 int，需要开 long long

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=5e5+5;
struct seg
```

```
{
    ll O,I,OI,IO,IOI;
    // 0 1 01 10 101
}tr[maxn<<2];
int n,m;
char s[maxn];
#define lson now<<1
#define rson now<<1|1
void pushup(int now)
{
    tr[now].O=tr[lson].O+tr[rson].O;
    tr[now].I=tr[lson].I+tr[rson].I;
    tr[now].OI=tr[lson].OI+tr[rson].OI+tr[lson].O*tr[rson].I;
    tr[now].IO=tr[lson].IO+tr[rson].IO+tr[lson].I*tr[rson].O;
    tr[now].IOI=tr[lson].IOI+tr[rson].IOI+tr[lson].IO*tr[rson].I+tr[lson].I*tr[rson].OI;
}
void build(int now,int l,int r)
{
    if(l==r)
    {
        if(s[l]=='I') tr[now].I=1;
        else tr[now].O=1;
        return;
    }
    int mid=l+r>>1;
    build(lson,l,mid); build(rson,mid+1,r);
    pushup(now);
}
void modify(int now,int l,int r,int pos,int val)
{
    if(l==r)
    {
        if(val) tr[now].I=1,tr[now].O=0;
        else tr[now].O=1,tr[now].I=0;
        return;
    }
    int mid=l+r>>1;
    if(pos<=mid) modify(lson,l,mid,pos,val);
    else modify(rson,mid+1,r,pos,val);
    pushup(now);
}
seg merge(seg x,seg y)
{
    seg res;
    res.O=x.O+y.O; res.I=x.I+y.I;
    res.OI=x.OI+y.OI+x.O*y.I;
    res.IO=x.IO+y.IO+x.I*y.O;
    res.IOI=x.IOI+y.IOI+x.I*y.OI+x.IO*y.I;
    return res;
}
seg query(int now,int l,int r,int L,int R)
```

```
{
    if(l>=L && r<=R) return tr[now];
    int mid=l+r>>1;
    if(R<=mid) return query(lson,l,mid,L,R);
    if(mid<L) return query(rson,mid+1,r,L,R);
    return merge(query(lson,l,mid,L,R),query(rson,mid+1,r,L,R));
}
int main()
{
    // freopen("a.in","r",stdin);
    // freopen("a.out","w",stdout);
    scanf("%d%d",&n,&m);
    scanf("%s",s+1);
    build(1,1,n);
    for(int i=1;i<=m;i++)
    {
        int op,x,y;
        char cc[5];
        scanf("%d%d",&op,&x);
        if(op==1)
        {
            scanf("%s",cc);
            if(cc[0]=='I') y=1;
            else y=0;
            modify(1,1,n,x,y);
        }
        else
        {
            scanf("%d",&y);
            printf("%lld\n",query(1,1,n,x,y).IOI);
        }
    }
    return 0;
}
```