

# 牛客多校第八场

想起高中时看的一本大黑砖：

《我怎样爆零》How Do I Get Zero

所以预计本周周报将陷入无题可写的困境。

I

比赛时以为是二分图的匹配用匈牙利算法做了，搞了半天搞出来了但是复杂度太高了过不去。

参考题解的答案是：总点数-# $k$ 个点 $k-1$ 条边的连通分量)

```
#include<stdio.h>
#include<string.h>

#include<vector>
#include<algorithm>

using namespace std;

bool vis[200020];
int F[200020];

void init()
{
    memset(F,-1,sizeof(F));
    memset(vis,0,sizeof(vis));
}

int find(int x)
{
    if(F[x]==-1)
    {
        return x;
    }
    return F[x]=find(F[x]); //改了根
}

void bing(int x,int y)
{
    int t1=find(x); //找到x的根
    int t2=find(y); //找到y的根
    if(t1==t2)
    {
        vis[t1]=1; //同一个根，该根变1
        return;
    }
}
```

```
}
F[t1]=t2;//如果x和y的根不同,修改t1的根t2
if(vis[t1])
{
    vis[t2]=1;
}
}

int main()
{
    int t;
    scanf("%d",&t);
    int k=0;
    while(t--)
    {
        k++;
        init();
        int n;
        scanf("%d",&n);
        vector<int> a(n),b(n);
        vector<int> val;
        int i;
        for(i=0;i<n;i++)
        {
            scanf("%d%d",&a[i],&b[i]);
            val.push_back(a[i]);
            val.push_back(b[i]);
        }
        sort(val.begin(),val.end());
        int tot=unique(val.begin(),val.end())-val.begin();
        val.resize(tot);
        for(i=0;i<n;i++)
        {
            a[i]=lower_bound(val.begin(),val.end(),a[i])-val.begin();
            b[i]=lower_bound(val.begin(),val.end(),b[i])-val.begin();
            bing(a[i],b[i]);
        }
        int ans=tot;
        for(i=0;i<tot;i++)//遍历c中不同的点
        {
            if(F[i]==-1&&!vis[i])
            {
                ans--;
            }
        }
        printf("Case #%d: %d\n",k,ans);
    }
}
```

# G

看代码觉得这个题明明没有描述的那么难啊.....

```
#include<stdio.h>
#include<string.h>

int a[300][5];
char s[100];

char check(int x,int y,int z)
{
    if((x==y&&x==z&&y==z) || (x!=y&&x!=z&&y!=z) || x== -1 || y== -1 || z== -1)
    {
        return 1;
    }
    return 0;
}

int main()
{
    int t,n,top;
    scanf("%d",&t);
    int kk=0;
    while(t--)
    {
        kk++;
        scanf("%d",&n);
        int i;
        for(i=1;i<=n;i++)
        {
            scanf("%s",s);
            top=0;
            int j;
            for(j=0;j<strlen(s);j++)
            {
                if(s[j]=='[')
                {
                    if(s[j+1]=='*')
                    {
                        a[i][top++] = -1;
                    }
                    else
                    {
                        a[i][top++] = s[j+2] - 'a';
                    }
                }
            }
        }
    }
}
```

```
char flag = 0;
printf("Case #%d: ", kk);
for(i=1; i<=n; i++)
{
    if(flag)
    {
        break;
    }
    int j;
    for(j=i+1; j<=n; j++)
    {
        if(flag)
        {
            break;
        }
        int k;
        for(k=j+1; k<=n; k++)
        {
            if(flag)
            {
                break;
            }
            char hv=1;
            int d;
            for(d=0; d<4; d++)
            {
                if(!check(a[i][d], a[j][d], a[k][d]))
                {
                    hv=0;
                    break;
                }
            }
            if(hv)
            {
                printf("%d %d %d\n", i, j, k);
                flag=1;
                break;
            }
        }
    }
}
if(!flag)
{
    puts("-1");
}
return 0;
}
```

## K

本来在赛场上我已经几乎把这个题分析完了，只是觉得维护太麻烦，可能会用到线段树之类（不会），就没写。

然后看到这份代码，觉得这就是个暴力.....

```
#include<stdio.h>

int T;
long long Max[101000];
long long a[101000];
int b[101000];

int main()
{
    scanf("%d",&T);
    int tt;
    for(tt=1;tt<=T;tt++)
    {
        int n;
        scanf("%d",&n);
        long long now = 0;
        Max[0] = -1e10;
        int i;
        for(i = 1; i <= n; i++)
        {
            scanf("%lld", &a[i]);
            now += a[i]; // 前缀和
            Max[i] = Max[i-1] > now ? Max[i-1] : now; // 求当前最大利润
        }
        unsigned long long ans1 = 0, ans2 = 0; // 开两个的原因是unsigned long
        long不能出现负数，为了解决用两个unsigned long long来存值，一个存负值，一个存正值
        int w;
        for(i = 1; i <= n + 1; i++)
        {
            if(i <= n)
            {
                scanf("%d", &b[i]);
            }
            else
            {
                b[i] = 0;
            }
            if(i == 1)
            {
                w = b[1];
            }
            if(b[i] < w)
```

```
{
    if(Max[i - 1] < 0)
    {
        ans2 += (unsigned long long)(-Max[i - 1]) * (w - b[i]);
    }
    else
    {
        ans1 += (unsigned long long)Max[i - 1] * (w - b[i]);
    }
    w = b[i]; //用了w-b[i]个, 剩下了b[i]个, 最后有个b[n+1]=0收尾, 不重不漏
}
}
if(ans1 < ans2)
{
    printf("Case #%d: %d -%llu\n", tt, b[1], ans2 - ans1);
}
else
{
    printf("Case #%d: %d %llu\n", tt, b[1], ans1 - ans2);
}
}
```

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
<https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:%E7%89%9B%E5%AE%A2%E5%A4%9A%E6%A0%A1%E7%AC%AC%E5%85%AB%E5%9C%BA>

Last update: 2020/08/08 20:55