

# 2020牛客暑期多校训练营（第八场）

## 比赛情况

|    |   |   |   |   |   |   |   |   |   |   |   |
|----|---|---|---|---|---|---|---|---|---|---|---|
| 题号 | A | B | C | D | E | F | G | H | I | J | K |
| 状态 | - | - | - | - | 0 | - | 0 | - | 0 | - | 0 |

0 在比赛中通过 0 赛后通过! 尝试了但是失败了- 没有尝试

比赛时间

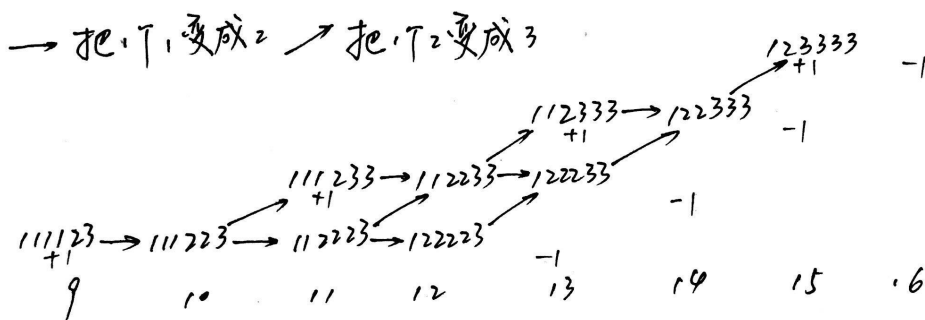
2020-08-03 12:00-17:00

## 题解

### E - Enigmatic Partition

$f(x)$  表示  $x$  由连续三个数字组成的划分有多少种，给出  $l,r$  求  $\sum_{k=l}^r f(k)$

显然只要能求出来所有  $f(x)$  就可以通过前缀和  $O(1)$  得到  $\sum_{k=l}^r f(k)$



先放一个写题解必画的图...这是当起始数字  $i=1$  长度固定为  $j=6$  时的情况，差分一次我们发现  $+1$  是从  $i+j+3$  开始往前跳两个数字一次， $-1$  是从  $(i+1)j+1$  开始往前跳一个数字一次。于是再差分一次（ $+1$  是跨度  $2$  的差分），这样就可以枚举  $i,j$  后  $O(1)$  计入贡献，最后做几次前缀和还原原数组即可。

```

#include<bits/stdc++.h>
#define ll long long
#define pii pair<int,int>
#define mp make_pair
#define fi first
#define se second
#define pb push_back
using namespace std;
const int N=1e5+10;
ll add[10*N],del[10*N],f[N];

```

```
int main()
{
    for(int i=1;i<N/3;i++)
    for(int j=3;j*i+3<N;j++)
    {
        add[i*j+3]++,add[(i+2)*j-1]--;
        del[(i+1)*j+1]++,del[(i+2)*j-1]--;
    }
    for(int i=2;i<N;i++)
    add[i]+=add[i-2],del[i]+=del[i-1],f[i]=add[i]-del[i]+f[i-1];
    for(int i=1;i<N;i++)f[i]+=f[i-1];
    int t;
    scanf("%d",&t);
    for(int ca=1;ca<=t;ca++)
    {
        int l,r;
        scanf("%d%d",&l,&r);
        printf("Case #d: %lld\n",ca,f[r]-f[l-1]);
    }
    return 0;
}
```

## G - Game SET

每张卡牌有四个维度，每个维度有三种取向（或者一个通配符代表可以视作任一种），先要选出一组三张牌，使得每个维度要么都一样，要么都不一样。

先统计每种牌的数量，一共只有  $81$  种牌，暴力枚举两种确定第三种就好（但其实是只要超过  $21$  种必有可组的牌，复杂度更小）。

```
#include<bits/stdc++.h>
#define ll long long
#define pii pair<int,int>
#define mp make_pair
#define fi first
#define se second
#define pb push_back
using namespace std;
const int N=303;
int used[N],tim;
char s[4][50];
set<int>st[4][4][4][4],A,B,C,E,F,G;
vector<int>v[4];
int read()
{
    int x=0,f=1;char c=getchar();
    while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
```

```
while(c>='0'&& c<='9'){x=x*10+c-'0';c=getchar();}
return x*f;
}
void getS(char *p)
{
    char c=getchar();int i=0;
    while(c!='[')c=getchar();
    c=getchar();
    while(c!=']')p[i++]=c,c=getchar();
    p[i]='\0';
}
int main()
{
    int t=read();
    for(int ca=1;ca<=t;ca++)
    {
        memset(used,0,sizeof(used)),tim=0;
        for(int a=1;a<=3;a++)
        for(int b=1;b<=3;b++)
        for(int c=1;c<=3;c++)
        for(int d=1;d<=3;d++)
        set<int>().swap(st[a][b][c][d]);

        int n=read();
        for(int i=1;i<=n;i++)
        {
            getS(s[0]),getS(s[1]),getS(s[2]),getS(s[3]);
            vector<int>().swap(v[0]),vector<int>().swap(v[1]);
            vector<int>().swap(v[2]),vector<int>().swap(v[3]);

            if(strcmp(s[0],"*")==0)v[0].pb(1),v[0].pb(2),v[0].pb(3);
            else if(strcmp(s[0],"one")==0)v[0].pb(1);
            else if(strcmp(s[0],"two")==0)v[0].pb(2);
            else if(strcmp(s[0],"three")==0)v[0].pb(3);

            if(strcmp(s[1],"*")==0)v[1].pb(1),v[1].pb(2),v[1].pb(3);
            else if(strcmp(s[1],"diamond")==0)v[1].pb(1);
            else if(strcmp(s[1],"squiggle")==0)v[1].pb(2);
            else if(strcmp(s[1],"oval")==0)v[1].pb(3);

            if(strcmp(s[2],"*")==0)v[2].pb(1),v[2].pb(2),v[2].pb(3);
            else if(strcmp(s[2],"solid")==0)v[2].pb(1);
            else if(strcmp(s[2],"striped")==0)v[2].pb(2);
            else if(strcmp(s[2],"open")==0)v[2].pb(3);

            if(strcmp(s[3],"*")==0)v[3].pb(1),v[3].pb(2),v[3].pb(3);
            else if(strcmp(s[3],"red")==0)v[3].pb(1);
            else if(strcmp(s[3],"green")==0)v[3].pb(2);
            else if(strcmp(s[3],"purple")==0)v[3].pb(3);
        }
    }
}
```

```
        for(int a:v[0])for(int b:v[1])
        for(int c:v[2])for(int d:v[3])
            st[a][b][c][d].insert(i);
    }
    printf("Case #d: ",ca);
    int f=0;
    for(int a=1;a<=3&&!f;a++)
    for(int b=1;b<=3&&!f;b++)
    for(int c=1;c<=3&&!f;c++)
    for(int d=1;d<=3&&!f;d++)
    {
        if(st[a][b][c][d].empty())continue;
        if(st[a][b][c][d].size()>=3)
        {
            int cnt=0;
            for(int x:st[a][b][c][d])
            {
                printf("%d ",x),cnt++;
                if(cnt==3)break;
            }
            puts(""),f=1;
        }
        else
        {
            for(int p=1;p<=3&&!f;p++)
            for(int q=1;q<=3&&!f;q++)
            for(int g=1;g<=3&&!f;g++)
            for(int h=1;h<=3&&!f;h++)
            {
                if(st[p][q][g][h].empty())continue;
                int t1,t2,t3,t4;
                if(p==a)t1=a;else t1=6-a-p;
                if(b==q)t2=b;else t2=6-b-q;
                if(c==g)t3=c;else t3=6-c-g;
                if(d==h)t4=d;else t4=6-d-h;

                if(st[t1][t2][t3][t4].empty())continue;
                tim++;int x=0,y=0,z=0;
                for(int
k:st[a][b][c][d])if(used[k]!=tim){used[k]=tim,x=k;break;}
                for(int
k:st[p][q][g][h])if(used[k]!=tim){used[k]=tim,y=k;break;}
                for(int
k:st[t1][t2][t3][t4])if(used[k]!=tim){used[k]=tim,z=k;break;}
                if(x&&y&&z)printf("%d %d %d\n",x,y,z),f=1;
            }
        }
    }
    if(!f)puts("-1");
}
```

```

return 0;
}

```

## I - Interesting Computer Game

每回合给出两个数字，你可以从中选一个，问最终最多能选得多少种数字。

把每回合的两个数字连边，每个连通块大小为  $n$  则至少能选  $n-1$  个（仅不选dfs树根节点），当仅当有环可以选  $n$  个。

```

#include<bits/stdc++.h>
#define ll long long
#define pii pair<int,int>
#define mp make_pair
#define fi first
#define se second
#define pb push_back
using namespace std;
const int N=1e5+10;
ll read()
{
    ll x=0,f=1;char c=getchar();
    while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0' && c<='9'){x=x*10+c-'0';c=getchar();}
    return x*f;
}
int a[N],b[N],num[N*2],tot,father[N*2],sz[N*2],e[N*2];
int find(int x){return x==father[x]?x:father[x]=find(father[x]);}
int main()
{
    int t=read();
    for(int ca=1;ca<=t;ca++)
    {
        tot=0;
        int n=read();
        for(int
i=1;i<=n;i++)a[i]=read(),b[i]=read(),num[++tot]=a[i],num[++tot]=b[i];
        sort(num+1,num+1+tot);
        tot=unique(num+1,num+1+tot)-num-1;
        for(int i=1;i<=tot;i++)father[i]=i,sz[i]=1,e[i]=0;
        for(int i=1;i<=n;i++)
        {
            a[i]=lower_bound(num+1,num+1+tot,a[i])-num;
            b[i]=lower_bound(num+1,num+1+tot,b[i])-num;
            int fu=find(a[i]),fv=find(b[i]);
            if(fu!=fv){father[fu]=fv,sz[fv]+=sz[fu],e[fv]+=e[fu];}
            e[fv]++;
        }
    }
}

```

```
    }  
    int res=0;  
    for(int i=1;i<=tot;i++)  
        if(father[i]==i) res+=(e[i]>sz[i]-1)?sz[i]:sz[i]-1;  
    printf("Case #d: %d\n",ca,res);  
  
}  
return 0;  
}
```

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

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:wangzai\\_milk:20200803%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95&rev=1596629405](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:wangzai_milk:20200803%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95&rev=1596629405)

Last update: 2020/08/05 20:10

