牛客假日团队赛40（重现赛）

比赛网址

训练结果

- 时间:2020-5-24 13:00~18:00
- rank:2/30
- 完成情况：10/10/12

题解

B. Optimal Milking

题意

维护一个数列中最大独立点集合的和，即求得$max\{\sum a_{b_i}\},其中b_i单增□b_i-b_{i-1}>1$

m次操作，每次修改该数列一个点的数值，询问答案□ $$ n\leq4*10^4 \\ m\leq10^5 $$

题解

线段树维护最大独立点集合的和，维护信息为$mx[o][0/1][0/1]$表示o节点的答案，端点的数值选或者没选。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=40055;
ll n,m,mx[N<<2][2][2],a[N];
ll ans;
#define lson k<<1,l,mid
#define rson k<<1|1,mid+1,r
void pu(int k)
```

```
{
mx[k][0][0]=max(max(mx[k<<1][0][1],mx[k<<1][0][0])+mx[k<<1|1][0][0],mx[k<<1
][0][0]+mx[k<<1|1][1][0]);
mx[k][0][1]=max(max(mx[k<<1][0][1],mx[k<<1][0][0])+mx[k<<1|1][0][1],mx[k<<1
][0][0]+mx[k<<1|1][1][1]);
mx[k][1][0]=max(max(mx[k<<1][1][1],mx[k<<1][1][0])+mx[k<<1|1][0][0],mx[k<<1
][1][0]+mx[k<<1|1][1][0]);
mx[k][1][1]=max(max(mx[k<<1][1][1],mx[k<<1][1][0])+mx[k<<1|1][0][1],mx[k<<1
][1][0]+mx[k<<1|1][1][1]);
}
void build(int k,int l,int r)
{
    if(l==r) {mx[k][1][1]=a[l];return;}
    int mid=l+r>>1;
    build(lson);build(rson);
    pu(k);
}
void update(int k,int l,int r,int a,int b)
{
    if(l==r) {mx[k][1][1]=b;return;}
    int mid=l+r>>1;
    if(a<=mid) update(lson,a,b);
    else update(rson,a,b);
    pu(k);
}
int main()
{
    n=read();m=read();
    for(int i=1;i<=n;i++)
        a[i]=read();
    int x,y;
    build(1,1,n);
    for(int i=1;i<=m;i++)
    {
        x=read();y=read();
        update(1,1,n,x,y);
ans+=max(mx[1][0][0],max(mx[1][0][1],max(mx[1][1][0],mx[1][1][1])));
    }
    printf("%lld\n",ans);
    return 0;
}
```

C. Vacation Planning (gold)

题意

求一个图的多源最短路，给定的边连接的一个点保证在给定的 k 个点中q 组询问

$ 点数，边数 ^4 , k, q^4$

题解

路径一定是起始点到指定点再到终点，对k个点跑最短路即可，询问的时候枚举到哪个指定点算最小值。

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<queue>
#include<map>
#define fi first
#define se second
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=20055,M=205,inf=0x7fffff;
int n,m,k,ques,sum,dis[M][N],a[N],vis[N];
ll ans;
typedef pair<int,int> P;
priority_queue<P,vector<P>,greater<P> >q;
int tot,to[N],nextt[N],head[N],w[N];
void add(int a,int b,int c)
{
    to[++tot]=b;
    nextt[tot]=head[a];
    head[a]=tot;
    w[tot]=c;
}
void dij(int x)
{
    for(int i=1;i<=n;i++)
        dis[x][i]=inf;
    dis[x][a[x]]=0;
    q.push(P(0,a[x]));
    while(!q.empty())
    {
        P y=q.top();q.pop();
        int u=y.se;
        if(dis[x][u]<y.fi) continue;
        for(int i=head[u];i;i=nextt[i])
        {
            if(dis[x][to[i]]>dis[x][u]+w[i])
dis[x][to[i]]=dis[x][u]+w[i],q.push(P(dis[x][to[i]],to[i]));
```

```
        }
    }
}
int main()
{
    n=read();m=read();k=read();ques=read();
    int x,y,z;
    for(int i=1;i<=m;i++)
        x=read(),y=read(),z=read(),add(x,y,z);
    for(int i=1;i<=k;i++)
        a[i]=read(),vis[a[i]]=i;
    for(int i=1;i<=k;i++)
        dij(i);
    for(int i=1;i<=ques;i++)
    {
        x=read();y=read();
        if(vis[x])
        {
            if(dis[vis[x]][y]!=inf)
                sum++,ans+=dis[vis[x]][y];
        }
        else
        {
            int mn=inf;
            for(int j=head[x];j;j=nextt[j])
                if(dis[vis[to[j]]][y]!=inf)
                    mn=min(mn,w[j]+dis[vis[to[j]]][y]);
            if(mn!=inf)
                sum++,ans+=mn;
        }
    }
    cout<<sum<<"\n"<<ans<<endl;
    return 0;
}
```

F. Combination Lock

循环签到题，略

G.Pogo-Cow

题意

给定n头奶牛的坐标x_i和收益a_i,你从一个方向(正反都行)开始顺次选若干个奶牛，要求选的奶牛坐标差要单调不下降，求最大收益$n\leq1000$

题解

$dp[i][j]$表示从i转移j的最佳收益，正常DP的话还需要枚举一个k□复杂度是$O(n^3)$□根据转移方程$dp[i][j]=max\{dp[k][i]\}+a[j],x[i]-x[k]\leq x[j]-x[i]$□发现$dp[k][i]$可以用单调队列维护最大值，所以复杂度降为$O(n^2)$

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
#define mp make_pair
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=1010;
int n,A,B;
PII a[maxn];
int dp[maxn][maxn],ans;
int main()
{
    n=read();
    for(int i=1;i<=n;i++)A=read(),B=read(),a[i]=mp(A,B);
    sort(a+1,a+n+1);
    for(int i=1;i<n;i++)
    {
        int k=i-1,val=a[i].Y;
        for(int j=i+1;j<=n;j++)
        {
            while(k && a[j].X-a[i].X>=a[i].X-a[k].X)
            {
                val=max(val,dp[k][i]);
                k--;
            }
            dp[i][j]=val+a[j].Y;
            ans=max(ans,dp[i][j]);
        }
    }
    for(int i=1;i<=n/2;i++)swap(a[i],a[n-i+1]);
    mem(dp,0);
    for(int i=1;i<n;i++)
    {
        int k=i-1,val=a[i].Y;
        for(int j=i+1;j<=n;j++)
```

```
        {
            while(k && a[i].X-a[j].X>=a[k].X-a[i].X)
            {
                val=max(val,dp[k][i]);
                k--;
            }
            dp[i][j]=val+a[j].Y;
            ans=max(ans,dp[i][j]);
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

H. Crowded Cows

题意

给定n头奶牛的坐标x_i和高度h_i,问有多少头奶牛满足在$x_i+D\geq x_{now}$的奶牛中至少满足一个$2h_{i}\geq h_{now}$并且$x_i-D\leq x_{now}$的奶牛中至少满足一个$2h_{i}\geq h_{now}$

$n\leq5*10^4$

题解

讲所有奶牛按坐标排序，然后预处理一下RMQ维护一下区间奶牛的最大值，在统计时直接查询两端区间最大值是否满足大于二倍关系即可。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
#define mp make_pair
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=50010,inf=2e9;
int n,d,ans,A,B,Max[20][maxn],len[maxn];
```

```
PII a[maxn];
int query(int L,int R)
{
    if(L>R)return inf;
    int bin=len[R-L+1];
    return max(Max[bin][L],Max[bin][R-(1<<bin)+1]);
}
int main()
{
    n=read();d=read();
    for(int i=1;i<=n;i++)A=read(),B=read(),a[i]=mp(A,B);
    sort(a+1,a+n+1);
    for(int i=1;i<=n;i++)Max[0][i]=a[i].Y;
    for(int i=1;(1<<i)<=n;i++)
        for(int j=1;j+(1<<(i-1))<=n;j++)
            Max[i][j]=max(Max[i-1][j],Max[i-1][j+(1<<(i-1))]);
    for(int i=2;i<=n;i++)len[i]=len[i>>1]+1;
    for(int i=1;i<=n;i++)
    {
        int l=lower_bound(a+1,a+n,mp(a[i].X-d,0))-
a,r=lower_bound(a+1,a+n,mp(a[i].X+d,inf))-a;
        if(query(l,i)>=2*a[i].Y && query(i,r)>=2*a[i].Y)ans++;
    }
    printf("%d\n",ans);
    return 0;
}
```

J. No Change

题意

有 k 个面值给定的硬币，你需要按顺序买 n 个东西，一个硬币可以买多个，但不找零，求买完所有东西最多能剩下多少钱。

题解

状压dp□$f[S]$ 表示用集合 S 最多能买到的位置，转移的时候枚举用哪个硬币，在前缀和序列上二分即可

```
// luogu-judger-enable-o2
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;
const int N=100055;
inline int read()
```

```
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return f*k;
}
int sum[N],n,m,dp[N],c[50],ans=-1;
int main()
{
    m=read();n=read();
    for(int i=0;i<m;i++) c[i]=read();
    for(int i=1;i<=n;i++) sum[i]=read()+sum[i-1];
    sum[n+1]=0x3f3f3f3f;
    for(int i=0;i<(1<<m);i++)
    {
        if(dp[i]>=n) continue;
        for(int j=0;j<m;j++)
        if(!((i>>j)&1)) {int
k=(upper_bound(sum+dp[i],sum+2+n,sum[dp[i]]+c[j])-
sum-1);dp[i|(1<<j)]=max(dp[i|(1<<j)],k);}
    }
    for(int i=1;i<(1<<m);i++)
    if(dp[i]>=n)
    {
        int sum=0;
        for(int j=0;j<m;j++)
        if(!((i>>j)&1)) sum+=c[j];
        ans=max(ans,sum);
    }
    cout<<ans<<endl;return 0;
}
```

K.Line of Sight

题意

平面若干个点，有一个圆心在原点的圆，问有多少对点连线不与圆相交 $N\leq5*10^5$

题解

想了半天发现一个性质，就是对于每一个点过圆做两条切线，对应出一段弧，两个点满足条件当且仅当所对应的两段弧有交。这样就变成了圆弧求交个数问题，（）但是最后wa了，不知道为啥

L. Empty Stalls

题意

一个圆形的牛圈，每头牛有会从指定的位置开始往后走，直到找到一个空位置进去，问最后没有牛的编号最小的位置是哪个。

题解

模拟，直接从头开始走，转两圈就可以了

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
const int N=3000055;
int n,m,sum[N],vis[N];
int main()
{
    n=read();m=read();
    for(int i=1;i<=m;i++)
    {
        int X,Y,A,B;
        X=read();Y=read();A=read();B=read();
        for(int j=1;j<=Y;j++)
            sum[(1ll*j*A+B)%n]+=X;
    }
    int now=0;
    for(int i=0;i<n;i++)
    {
        now+=sum[i];
        if(now&&!vis[i])
            vis[i]=1,now--;
    }
    if(now>0)
    {
        for(int i=0;i<n;i++)
            if(!vis[i])
            {
                if(now)
```

```
                    now--;
                else
                {
                    printf("%d\n",i);break;
                }
            }
    }
    else
    {
        for(int i=0;i<n;i++)
            if(!vis[i])
            {
                printf("%d\n",i);break;
            }
    }
    return 0;
}
```

训练实况

0~1h 开局fyh读**B** ,hxm读**A** ,fyh误解题意□1wa **B**,hxm觉得**A**有难度，开始读**D**

fyh和wxg想出**B** wxg开写**B**

hxm看**E** fyh看**H** 得出算法

wxg WA **B** 开始肉眼调题□hxm写**E** 并通过 wxg发现线段树写错并通过**B**

fyh开写**H**

1~2h fyh通过**H** hxm开写**D** wxg,fyh开始读其他题 wxg发现**J**原题，秒通过

hxm通过**D** 双倍经验通过**I**

2~3h wxg写并通过**C** ,**F** ,**L**□fyh开写**G**,hxmwxg口糊**A** **K**

3h fyh通过**G** hxm开写**A**,fyh交替写**K**,直至比赛结束也没有过。