信息学联赛（NOIP2011）提高组题解

太原成成中学 张浩千

提高组 Day 1

1. Carpet 铺地毯

本题为送分题。

算法 1：

直接输出“-1”。

复杂度：O(1) 期望得分：10 分

算法 2：

对于 50%的数据有 a,b,g,k 小于 100，可以开一个二维数组 map[i][j]表示当前(I,j)，坐标上最上方的地毯编号。之后按照顺序处理每一个地毯，更新 map[i][j]即可。

复杂度 O(gkN) 期望得分 40-50 分

算法 3：

顺序处理每一个地毯，并判断当前的点是否在地毯覆盖的长方形内，并记录尽可能靠后的地毯编号，输出即可。

复杂度 O(N) 期望得分 100 分 具体请参考程序 1

注意：很容易将 g,k 的含义看成地毯的右上角坐标，仔细读题。

2. Hotel 选择客栈

按照题目要求的进行模拟，每次选择两个进行判断，若满足以下条件：

（1）两个客栈颜色相同

（2）两个客栈中间（包含两个客栈）中存在一家咖啡店最低消费小于等于 p 的

即算作一个满足条件的解，题目要求求出解的个数。

算法 1：

枚举两个客栈，判断颜色是否相同，并顺序枚举求两个客栈的最小值，看是否小于等于 p。

复杂度 O（N^3） 期望得分 30-40 分

算法 2：

对于求两个客栈的最小值问题，为经典的 RMQ 问题，可以用线段树维护一段区间的最小值。

复杂度 O（NlogN+N^2logN） 期望得分 50 分

算法 3：

对于思路 2，可以用 ST 算法 O(1)时间求出两个区间的最小值。

复杂度 O（NlogN+N^2） 期望得分 50-60 分 具体请参考程序 2

算法 4：

我们发现这个 RMQ 问题有很强的特殊性，由于我们要枚举所有的区间，按照双重循

的特点，我们每一次枚举，相比上次的区间仅仅多了一个客栈，则当前区间的最小值=min(上一个区间的最小值，当前客栈的最低消费)。

复杂度 O（N^2） 期望得分 50-60 分

算法 5：

对于每一个客栈，我们可以只考虑和当前点前面的客栈进行配对。预处理出 Pre[i]表示 i 点（包含 i 点）前面第一个<=p 的客栈。对于每一个颜色，求出 sum[i]表示前 i 个点有多少个点的颜色等于当前枚举的颜色。依次枚举每一个客栈，若当前客栈最低消费<=p，当前客栈可以和前面任意一个客栈进行匹配，并且 pre[i]一定等于 i，所以答案可以累加 sum[pre[i]]-1；对于当前客栈最低消费>p，则当前客栈只能和 pre[i]以及之前的客栈进行匹配，故答案可以累加 sum[pre[i]]。

复杂度 O（NK） 期望得分 100 分 具体请参考程序 3

算法 6：

基于分治的算法。用 ST 算法求出一个区间的最小值后找到这个客栈的编号 i，若 i 客栈的最低消费<=p，则 i 客栈可以将全部客栈分成两部分。之后依次枚举每一个颜色，答案累加 sum(l,i-1)*sum(i+1,r)，若当前客栈与枚举颜色相同，则答案再累加 sum(l,i-1)+sum(i+1,r) 注：sum(l,r)表示 i 到 j 之间有多少客栈颜色与当前枚举颜色相同，可以通过预处理前缀和 O(1)求出。之后递归计算被分成的两部分即可。

复杂度（NlogN+NK） 期望得分 90-100 分

3. Mayan 游戏

本题是一个搜索题。

算法 1：

直接输出“-1” 性价比较高。

复杂度 O（1） 期望得分 0-10 分

算法 2：

对于 30%的数据全部都在一行上，我们可以对一行进行搜索。可以采用 DFS 依次枚举当前每一个格子向左移动或者向右移动。由于向右移动总比向左移动字典序小，所以可以让向右移动尽可能替代向左移动，仅有一种情况无法替代例如：10122 中第三个位置上的方块“1”可以采用向左移动到第二个位置。递归层数限制为 N，若在 N 层恰好消除完，则输出方案。对于判重问题可以将当前状态转换成 11 进制一一映射到一个 hash 数组中。由于状态非常少，所以您不考虑上述的优化也可以拿到相应的分数。

复杂度 O（?） 期望分数 30 分

算法 3：

对于全部数据，我们依然可以采用 dfs 的搜索策略。对于思路 2 的字典序结论依然成立，这样对于任何一个状态，我们最多可以扩展 4*7=28 个状态。全部数据需要考虑方块下落以及消除方块的问题。设方块下落的处理函数为 down(),消除方块的处理函数为 clear()，对于 down()，我们可以从下到上依次判断每一个方块下方是否为 0，若为 0 则执行下落；对于 clear()，我们依次枚举每一个方块然后进行向下和向右扩展，若存在同颜色且个数>=3 时，将这些方块标记，最后统一消除。由于每一次执行完 clear()操作后，都有可能引发新的方块需要掉落，所以需要重复调用这两个函数。

复杂度 O（?） 期望分数 100 分 具体请参考程序 4

剪枝：

1. 若两种相同颜色的方块进行交换，显然没有意义。
2. 可行性剪枝：若某一种颜色的个数=1 或=2，则它显然无法消除。
3. 最优性剪枝：若某一行中存在某一种颜色，它只有 1 个或 2 个方块，则他必须借助其他行来帮助他消除，设离当前行最近且含有当前颜色的行与当前行的距离是 K，求出对于所有方块最大的 maxK，则 maxK 就是消除全部方块的下界，如果比 n-当前深度大，则可以剪枝。
4. 判重：可以将 3*5 的数组映射到一个数，建立 hash 表，对于冲突问题可以采用建链表的方式解决。

提高组 Day 2

1. Factor 计算系数

本题是一道数学题。

算法 1：

对于 30%的数据来说，可以手工推算 n<=10 的情况。

复杂度 O(1) 期望分数 30 分

算法 2：

根据二项式定理,(x+y)^k 展开式中 x^n*y^m 的系数为 C(k,n),则对于 50%的数据，题目转变成了求 C(k,n)mod 10007 的问题。关于组合数取模是一个经典问题，鉴于 N<=1000，可以采用递推的方式求解。

F[i][j]表示 C(I,j) mod 10007 的结果

F[0][0]=1 边界条件

F[i][j]=(f[i-1][j]+f[i-1][j-1]) mod 10007

结果就是 f[k][n]

复杂度 O（n^2） 期望分数 50 分

算法 3：

对于 100%的数据，则是求(ax+by)^k，根据算法 2 的结论，整个结果为 c(k,n)*a^n*b^m，因 n,m<=1000，完全没有必要使用快速幂求解。

复杂度 O（n^2） 期望分数 100 分 具体请参考程序 5

注意：

1. A,b 的范围是<1,000,000,在算幂的时候很有可能超过数据范围,所以一定要先让 A 和 B mod 10007。
2. K=0 的情况，结果应该是 1。
3. 注意当 A=0 或 B=0 的特殊情况。

2. Qc 聪明的质检员

算法 1：

对于 10%的算法，依次尝试每一个 W（范围可以是从所有矿石的最小值到最大值）

复杂度：O（10^6*nm） 期望分数 10 分

算法 2：

观察算法 1 可以发现，如果我们对 w[i]进行升序排序，在 w[i]到 w[i+1]-1 之间的计算结果都会相同。故我们可以依次枚举每一个 w[i]作为参数。

复杂度：O(n^2m) 期望分数 30 分

算法 3：

对于算法 1，我们需要依次枚举每一个区间，后枚举每一个区间内的所有矿石，由于区间可能重复的非常多，所以我们做了很多“无用功”。对于当前参数 w，我们可以预处理出一个前缀和 cnt[i]表示前 i 个矿石 w[i]>=w 的个数,sum[i]表示前 i 个矿石 w[i]>=w 的价值和，则递推式为：

边界条件：cnt[0]=0 sum[0]=0

若 w[i]>=w 则

Cnt[i]=cnt[i-1]+1

Sum[i]=sum[i-1]+v[i]

否则

Cnt[i]=cnt[i-1]

Sum[i]=sum[i-1]

则询问一个区间的校验值就为(cnt[r]-cnt[l-1])*(sum[r]-sum[l-1])。对于每一个参数 W，我们就可以只用 O(N+M) 的时间求可以求出所有区间的校验值和。

复杂度：O（10^6*(N+M)） 期望分数 30 分

算法 4：

结合算法 2 和算法 3

复杂度：O（NM） 期望分数 50 分

算法 5：

我们可以发现，随着 W 的增长，对于所有区间的校验值单调上升，这里就提示我们 abs(s-y)一定是一个单峰函数，我们的目标就是求这个单峰函数的最小值，对于单峰函数求最值可以用三分法。这里有一个更容易实现的方法，我们二分参数 w，求得小于等于 s，且最大的值，设为 x，则我们只需要检测参数 x 和 x+1 就可以知道最值了。另外一个小细节就是：由于最大的校验值可能会超过 int64(long long)类型，但是只要校验值大于 2S，那么就完全没有必要再去求它了，因为他还没有参数 W=0 优秀。

复杂度：O（NlogK） 期望分数 100 分 注：K 为二分的上界 具体请参考程序 6

3．Bus 观光公交

算法 1：

对于 10%的数据 K=0,即没有加速器的情况。设 time[i]表示公交到 i 景点的时间，num[i]表示 i 为起点的所有乘客达到 i 景点的最晚时间。则

Time[i]=max(time[i-1],num[i-1])+d[i-1]

则答案为∑time[b[i]]-t[i] i∈[1,m]

复杂度：O(n) 期望分数 10 分

算法 2：

对于 20%的数据 K=1，即只能使用一个加速器，由于 N<=1000，于是我们可以依次尝试减少每一个 d[i]，然后继续执行算法 1。

复杂度：O(n^2) 期望分数 20 分 若同时结合算法 1 可以得到 30 分

算法 3：

观察∑time[b[i]]-t[i] i∈[1,m]等价于∑time[b[i]]-∑t[i] i∈[1,m]，由于∑t[i]是定值，所以我们需要让∑time[b[i]]尽可能小。

观察 Time[i]=max(time[i-1],num[i-1])+d[i-1]，若我们使 d[i-1]减少 1，则

time[i]一定会减少 1，之后景点的达到时间是否会减少是不确定的，这取决于time[i]与num[i]的大小关系。若time[i]>num[i]，则在max过程中time[i]起到主导作用，由于time[i]减少 1，则time[i+1]也一定会减少 1。相反若time[i]<=num[i]，则num[i]起到主导作用，减少time[i]不会引起max函数值的变化，所以time[i+1]不会减少。从这里可以看出，若使d[i]-1则引起达到时间减少的一定是一个从i+1开始的连续的区间。

设g[i]表示若使d[i]-1，则可以减少达到时间的景点区间是[i+1,g[i]]，g[i]可以通过递推的方式求出。

如果time[i+1]>num[i+1]，则g[i]=g[i+1]

如果time[i+1]<=num[i+1]，则g[i]=i+1

设sum[i]表示前i个景点已经达到的人数。若当前减少的区间是[l,r]，则答案可以减少sum[r]-sum[l-1]。之后根据贪心的原则，每一次选择当前可以让时间减少最多的一个区间，之后由于g[i]和time[i]发生变化，需要重新进行维护。由于之后的决策都没有当前决策优，将后面的决策替换当前决策，则一定不优，所以贪心的方法是正确的。

复杂度：O(nk) 期望分数：80-100分　具体请参考程序7

参考程序

1. 参考程序1 Carpet 铺地毯 算法3

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
using namespace std;
const int maxn=200010;
int a[maxn],b[maxn],lenx[maxn],leny[maxn];
int i,j,n,m,x,y;

int main()
{
    freopen("carpet.in","r",stdin);
    freopen("carpet.out","w",stdout);
    scanf("%d",&n);
    for (i=1;i<=n;i++)
        scanf("%d%d%d%d",&a[i],&b[i],&lenx[i],&leny[i]);
    scanf("%d%d",&x,&y);
    int k=-1;
    for (i=1;i<=n;i++)
        if ((a[i]<=x)&&(x<=a[i]+lenx[i])&&(b[i]<=y)&&(y<=b[i]+leny[i]))
           k=i;
    printf("%d\n",k);
    return 0;
}
```

2. 参考程序 2 Hotel 选择客栈 算法 3

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<cmath>
using namespace std;
const int maxn=300000;
const int maxm=30;
int f[maxn][maxm],col[maxn],a[maxn];
int i,j,n,k,p;

int min(int a,int b)
{
    if (a>b) return b;else return a;
}

int find(int l,int r)
{
    int k=int(log(r-l+1)/log(2));
    return min(f[l][k],f[r-(1<<k)+1][k]);
}

int main()
{
    freopen("hotel.in","r",stdin);
    freopen("hotel.out","w",stdout);
    scanf("%d%d%d",&n,&k,&p);
    for (i=1;i<=n;i++)
    {
        scanf("%d%d",&col[i],&a[i]);
        f[i][0]=a[i];
    }
    for (j=1;j<=int(log(n)/log(2));j++)
        for (i=1;i<=n-(1<<j)+1;i++)
            f[i][j]=min(f[i][j-1],f[i+(1<<(j-1))][j-1]);
    int ans=0;
    for (i=1;i<=n;i++)
        for (j=i+1;j<=n;j++)
            if (col[i]==col[j])
               if (find(i,j)<=p)
                  ans++;
    printf("%d\n",ans);
    return 0;
}
```

3. 参考程序 3 Hotel 选择客栈 算法 5

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<iostream>
#define ll long long
using namespace std;
const int maxn=200010;
int col[maxn],a[maxn],sum[maxn],pre[maxn];
int i,j,n,k,p;
ll ans;

int get(int l,int r)
{
    return sum[r]-sum[l-1];
}

int main()
{
    freopen("hotel.in","r",stdin);
    freopen("hotel.out","w",stdout);
    scanf("%d%d%d",&n,&k,&p);
    for (i=1;i<=n;i++)
        scanf("%d%d",&col[i],&a[i]);
    for (j=0;j<k;j++)  //for color
    {
        memset(sum,0,sizeof(sum));
        for (i=1;i<=n;i++)
            if (col[i]==j) sum[i]=sum[i-1]+1;else sum[i]=sum[i-1];
        memset(pre,0,sizeof(pre));
        int last=0;
        for (i=1;i<=n;i++)
        {
            if (a[i]<=p) last=i;
            pre[i]=last;
        }
        for (i=1;i<=n;i++)
            if (col[i]==j)
            {
               if (a[i]<=p) ans+=get(1,pre[i]-1);
               else ans+=get(1,pre[i]);
            }
    }
    cout<<ans<<endl;
```

```
    return 0;
}
```

4. 参考程序 4 mayan 游戏 算法 3

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
using namespace std;
const int maxh=5;
const int maxs=7;
const int maxn=500000;
const int maxcol=20;
int i,j,n,m,top;
int a[maxh][maxs];
int stack[maxn][maxh][maxs];
int x[maxn],y[maxn],c[maxn],father[maxn],tot[maxcol];
bool check,bj[maxh][maxs];

void output(int p)
{
    if (p==1) return;
    output(father[p]);
    printf("%d %d %d\n",x[p],y[p],c[p]);
}

bool empty(int a[maxh][maxs])
{
    int i,j;
    for (i=0;i<5;i++)
        for (j=0;j<7;j++)
            if (a[i][j]!=0) return false;
    return true;
}

void swap(int x1,int y1,int x2,int y2)
{
    int t=a[x1][y1];
    a[x1][y1]=a[x2][y2];
    a[x2][y2]=t;
}

bool clear(int a[maxh][maxs])
{
    int i,j,k,tot;
```

```
    bool change=false;
    memset(bj,0,sizeof(bj));
    for (i=0;i<5;i++)
        for (j=0;j<7;j++)
            if (a[i][j]!=0)
            {
                int col=a[i][j];
                //横
                tot=1;
                k=i+1;
                while ((a[k][j]==col)&&(k<=maxh)) tot++,k++;
                if (tot>=3)
                {
                    change=true;
                    k=i;
                    while ((a[k][j]==col)&&(k<=maxh)) bj[k][j]=1,k++;
                }
                //竖
                tot=1;
                k=j+1;
                while ((a[i][k]==col)&&(k<=maxs)) tot++,k++;
                if (tot>=3)
                {
                    change=true;
                    k=j;
                    while ((a[i][k]==col)&&(k<=maxs)) bj[i][k]=1,k++;
                }
            }
    for (i=0;i<5;i++)
        for (j=0;j<7;j++)
            if (bj[i][j]==1)
               a[i][j]=0;
    return change;
}

void down(int a[maxh][maxs])
{
    int i,j,k;
    for (i=0;i<5;i++)
        for (j=1;j<7;j++)
            if (a[i][j]!=0)
            {
                k=j-1;
                while ((k>=0)&&(a[i][k]==0)) swap(i,k,i,k+1),k--;
```

```
            }
}

void work(int a[maxh][maxs])
{
    down(a);
    while (clear(a)) down(a);
}

bool workable(int a[maxh][maxs])
{
    int i,j;
    memset(tot,0,sizeof(tot));
    for (i=0;i<5;i++)
        for (j=0;j<7;j++)
            if (a[i][j]!=0)
                tot[a[i][j]]++;
    for (i=1;i<=10;i++)
        if ((tot[i]==1)||(tot[i]==2))
            return false;
    return true;
}

void dfs(int dep,int fa)
{
    if (check) return ;
    if (dep>n)
    {
        if (empty(stack[fa]))
        {
            check=true;
            output(fa);
        }
        return;
    }
    int i,j;
    for (i=0;i<5;i++)
        for (j=0;j<7;j++)
            if (stack[fa][i][j]!=0)
            {
                //1 direction
                memcpy(a,stack[fa],sizeof(a));
                if (i!=4)
                {
```

```
                    swap(i,j,i+1,j);
                    work(a);  //进行消方块操作
                    if ((((dep==n)||(empty(a)==false)))&&(workable(a)))
                    {
                       memcpy(stack[++top],a,sizeof(a));
                       x[top]=i;
                       y[top]=j;
                       c[top]=1;
                       father[top]=fa;
                       //if (dep==1)
                       //outputs(stack[top]);
                       dfs(dep+1,top);
                       top--;
                    }
                }
                //-1 direction
                memcpy(a,stack[fa],sizeof(a));
                if ((i!=0)&&(a[i-1][j]==0))
                {
                    swap(i,j,i-1,j);
                    work(a);  //进行消方块操作
                    if ((((dep==n)||(empty(a)==false)))&&(workable(a)))
                    {
                       memcpy(stack[++top],a,sizeof(a));
                       x[top]=i;
                       y[top]=j;
                       c[top]=-1;
                       father[top]=fa;
                       dfs(dep+1,top);
                       top--;
                    }
                }
            }
}

int main()
{
    freopen("mayan.in","r",stdin);
    freopen("mayan.out","w",stdout);
    scanf("%d",&n);
    for (i=0;i<5;i++)
    {
        j=-1;
        while (1)
```

```
        {
            scanf("%d",&a[i][++j]);
            if (a[i][j]==0) break;
        }
    }
    top=1;
    memcpy(stack[top],a,sizeof(a));
    dfs(1,1);   //深度是 1 ,fa 是编号为 1 的 stack
    if (check==false) printf("-1\n");
    return 0;
}
```

5. 参考程序 5 factor 计算系数 算法 3

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
using namespace std;
const int maxn=2010;
const int prime=10007;
int i,j,a,b,p,q,n;
int f[maxn][maxn];

int mi(int a,int b)
{
    int t=1;
    for (int i=1;i<=b;i++)
        t=(t*a)%prime;
    return t;
}

int main()
{
    freopen("factor.in","r",stdin);
    freopen("factor.out","w",stdout);
    scanf("%d%d%d%d%d",&a,&b,&n,&q,&p);
    a%=prime;
    b%=prime;
    for (i=0;i<=n;i++)
        f[i][0]=1;
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++)
            f[i][j]=(f[i-1][j]+f[i-1][j-1])%prime;
    int ans=(f[n][q]*mi(a,q))%prime*mi(b,p)%prime;
    if (n==0) ans=1;
```

```
    printf("%d\n",(ans+prime)%prime);
    return 0;
}
```

6. 参考程序 6 qc 聪明的质检员 算法 5

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#include<iostream>
#define ll long long
using namespace std;
const ll maxn=210000;
const ll oo=2011101215;
ll w[maxn],v[maxn],l[maxn],r[maxn];
ll sum[maxn],tot[maxn];
ll i,j,n,m,up,down;
ll s,ans;

ll find(ll k)
{
    ll i;
    ll t=0;
    tot[0]=0;
    sum[0]=0;
    for (i=1;i<=n;i++)
        if (w[i]>k)
        {
            tot[i]=tot[i-1]+1;
            sum[i]=sum[i-1]+v[i];
        }
        else
        {
            tot[i]=tot[i-1];
            sum[i]=sum[i-1];
        }
    for (i=1;i<=m;i++)
    {
        ll totnum=tot[r[i]]-tot[l[i]-1];
        ll sumnum=sum[r[i]]-sum[l[i]-1];
        t+=totnum*sumnum;
    }
    return t;
}
```

```
int main()
{
    freopen("qc.in","r",stdin);
    freopen("qc.out","w",stdout);
    scanf("%I64d%I64d%I64d",&n,&m,&s);
    up=-oo;
    down=oo;
    for (i=1;i<=n;i++)
    {
        scanf("%I64d%I64d",&w[i],&v[i]);
        up=max(up,w[i]);
        down=min(down,w[i]);
    }
    for (i=1;i<=m;i++)
        scanf("%I64d%I64d",&l[i],&r[i]);
    ans=s;
    ll left=down;
    ll right=up;
    while (right-left>1)
    {
         ll mid=(left+right)/2;
         ll t=find(mid);
         if (s>t) right=mid;else left=mid;
    }
    ans=min(abs(s-find(left)),abs(s-find(right)));
    cout<<ans<<endl;
    return 0;
}
```

7. 参考程序 7 bus 观光公交 算法 3

```
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
using namespace std;
const int maxn=100100;
const int oo=2011101215;
int i,j,n,m,k,ans;
int d[maxn],t[maxn],a[maxn],b[maxn];
int tim[maxn],num[maxn],g[maxn],sum[maxn];

void find()
{
```

```
    int i;
    for (i=1;i<=m;i++)
        num[a[i]]=max(num[a[i]],t[i]);
    for (i=1;i<=n;i++)
        tim[i]=max(tim[i-1],num[i-1])+d[i-1];
}

int first()
{
    find();
    int tot=0;
    for (i=1;i<=m;i++)
        tot+=tim[b[i]]-t[i];
    for (i=1;i<=m;i++)
        sum[b[i]]++;
    for (i=1;i<=n;i++)
        sum[i]+=sum[i-1];
    g[n-1]=n;
    for (i=n-2;i;i--)
        if (tim[i+1]<=num[i+1])
            g[i]=i+1;
        else
            g[i]=g[i+1];
    return tot;
}

int main()
{
    freopen("bus.in","r",stdin);
    freopen("bus.out","w",stdout);
    scanf("%d%d%d",&n,&m,&k);
    for (i=1;i<n;i++)
        scanf("%d",&d[i]);
    for (i=1;i<=m;i++)
        scanf("%d%d%d",&t[i],&a[i],&b[i]);
    int ans=first();
    for (j=1;j<=k;j++)
    {
        //若调整 i，则影响的区间是[i+1,g[i]]
        int maxnum=0,p=0;
        for (i=1;i<n;i++)
            if ((sum[g[i]]-sum[i]>maxnum)&&(d[i]>0))
            {
                maxnum=sum[g[i]]-sum[i];
```

```
                p=i;
            }
        d[p]--;
        ans-=maxnum;
        //更新 tim[i+1,g[i]]
        int pos=g[p];
        int l=p,r=min(n-2,pos);
        for (i=l;i<=r;i++)
            tim[i]=max(tim[i-1],num[i-1])+d[i-1];
        for (i=r;i>=l;i--)
            if (tim[i+1]<=num[i+1])
                g[i]=i+1;
            else
                g[i]=g[i+1];
    }
    printf("%d\n",ans);
    return 0;
}
```