T1 魔法照片

这个题就送分用的(~~这就是你格式错误的理由？~~)也不卡常，两遍排序即可

注意这玩意要求稳定排序，自己写个比较函数丢$sort$就行了

Code

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
using namespace std;
int n,k,ex[20];
struct PEO
{
        int id,val;
}peo[100005];
bool cmp(PEO a,PEO b)
{
        if (a.val!=b.val) return a.val>b.val;
        return a.id<b.id;
}
signed main()
{
        cin>>n>>k;
        for (int i=1;i<=10;i++) cin>>ex[i];
        for (int i=1;i<=n;i++) cin>>peo[i].val,peo[i].id=i;
        sort(peo+1,peo+1+n,cmp);
        for (int i=1;i<=n;i++)
                peo[i].val+=ex[((i-1)%10)+1];
        sort(peo+1,peo+1+n,cmp);
        for (int i=1;i<=k;i++) cout<<peo[i].id<<' ';
        return 0;
}
```

End

T2 魔法药水

题意

有一堆药水，你要合成 0 号药水，给了你一些药方（指A药水+B药水=C药水这种）和所有药水的成本

求 合成 0 号药水的最少成本和此时的方案数

Solve

考场思路

看见题目有物品与物品之间的对应关系，求最小成本，便觉得是图论题

但是在构造了 $20min$ 图论模型未果后果断放弃，然后就写了下面这个奇葩解法

先不考虑方案数问题，搞定最小成本再说

因为感觉这药方关系过于混乱，方案数不太好搞就先搞最低成本

因为要求最小成本，所以配置完成之后一定没有多余的药水剩下

我想着既然先不考虑方案，那自然不用关心中间过程，那么

一个药方$\Longleftrightarrow$得到一种药水的新成本

如果更划算，那么直接改这个药水的成本

只要还有药方能降低某种药药水的成本就不断改下去

因为 0 号药水 也是药水，只能直接买或者用药方配，所以这么干一定能得到最低成本

就是这个

```
bool flag=true;
while (1)
{
        flag=true;
        for (int i=1;i<=cnt;i++)
                if (med[i].delta>0)
                {
                        flag=false;
                        price[med[i].get]-=med[i].delta;
                        for (int j=i+1;j<=cnt;j++)
                                med[j].delta=
                                price[med[j].get]-
                                 (price[med[j].a]+price[med[j].b]);
                }
        for (int j=1;j<=cnt;j++)
                med[j].delta=price[med[j].get]-price[med[j].a]-price[med[j].b];
        if (minans>price[0]) minans=price[0];
        if (flag) break;
}
```

那方案数怎么求

根据我们上面的更新方法，这么干完之后留下的就是每一种药水的最低价格

开一个 $vector<int>temp[n]$，把能得到每种药水的每一个药方的存进去

我们想一下，如果这个药方的俩原材料(a,b)的成本加起来等于成果(c)的成本，那么就可以说明这是一个可以采用的药方

这药方的原材料也可能有药方可以合成，如此下去，再想一下，这玩意貌似就是一棵树哇

根据数学知识，$plan_{[c]}=plan_{[a]}\times plan_{[b]}$

有个小点要注意一下，如果这个药水的最低成本和他的原始成本一样，那么这个$plan$还得$+1$

Code

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#include<vector>
using namespace std;
int n,k,x,y,z;
int price[1100],P[1100];
struct Med
{
        int a,b;
        int get;
        int delta;
}med[100000];
int cnt;
int minans,ans;
vector<int>temp[1100];
int dfs(int pos)
{
        int res=0;
        if (price[pos]==P[pos]) res++;//注意这个点
        bool flag=true;
        for (int i=0,to1,to2,temp1,temp2;i<temp[pos].size();i++)
        {
                to1=med[temp[pos][i]].a;
                to2=med[temp[pos][i]].b;
                if (price[to1]+price[to2]>price[pos]) continue;
                flag=false;
                temp1=dfs(to1);temp2=dfs(to2);
                res+=temp1*temp2;
        }
        if (flag) res=1;//注意这个边界条件
        return res;
}
signed main()
{
        cin>>n;
        for (int i=0;i<n;i++) cin>>price[i],P[i]=price[i];
        while (cin>>x>>y>>z)
        {
                med[++cnt].a=x;med[cnt].b=y;med[cnt].get=z;
                med[cnt].delta=price[z]-(price[x]+price[y]);
                temp[z].push_back(cnt);
        }
        minans=price[0];
        bool flag=true;
        while (1)
        {
                flag=true;
                for (int i=1;i<=cnt;i++)
                        if (med[i].delta>0)
                        {
                                flag=false;
```

```
                                price[med[i].get]-=med[i].delta;
                                for (int j=i+1;j<=cnt;j++)
                                        med[j].delta=
                                        price[med[j].get]-
                                        (price[med[j].a]+price[med[j].b])
                                        ;
                        }
                for (int j=1;j<=cnt;j++)
                        med[j].delta=price[med[j].get]-price[med[j].a]-price[med[j].b];
                if (minans>price[0]) minans=price[0];
                if (flag) break;
        }
        ans+=dfs(0);
        cout<<price[0]<<' '<<ans;
        return 0;
}
```

正解

本题正解是$dijkstra$

Dijkstra的思想(贪心选择用已知的最优解来更新未知最优解)。

我们用一个vis数组来标记一种药水的最小花费是否确定，如果$vis[i]$为$true$，则表示i号药水的最小花费已经确定，否则反之。

同时，用$cost[i]$和$ans[i]$记录当前i号药水的最小花费和满足最小花费的方案个数

$edge[i][j]$ 就是存的药方 $i+j=edge[i][j]$

每次选择一个没去过的最小的$cost[pos]$

参考$dijkstra$是怎么写的

若 $j+pos=i$,且这个药方能参与更新

如果$cost[j]+cost[pos]<cost[f[j][pos]]$，

则将$cost[edge[j][pos]]$更新为$cost[j]+cost[pos]$，并将ans[i]重置为1；

如果$cost[j]+cost[pos]=cost[edge[j][pos]]$

则将$ans[edge[j][pos]]$加上$ans[j]*ans[pos]$

最后答案就是$cost[0]$和$ans[0]$

Code

```
#include<cmath>
#include<iostream>
#include<vector>
#include<cstring>
using namespace std;
#define int long long
```

```
int n;
int price[1100],edge[1100][1100],ans[1100];
bool vis[1100];
int a,b,c,pos,maxx=0x3f3f3f3f;
signed main()
{
        memset(edge,-1,sizeof(edge));
        cin>>n;
        for (int i=0;i<n;i++) cin>>price[i],ans[i]=1;
        while (cin>>a>>b>>c) {edge[a][b]=c;edge[b][a]=c;}
        for (int i=0;i<n;i++)
        {
                maxx=0x3f3f3f3f;
                for (int j=0;j<n;j++)
                        if (!vis[j]&&price[j]<maxx) pos=j,maxx=price[j];
                vis[pos]=true;
                for (int j=0,Next;j<n;j++)
                        if (vis[j]&&edge[pos][j]!=-1)
                        {
                                Next=edge[pos][j];
                                if (price[pos]+price[j]<price[Next])
                                {
                                        price[Next]=price[pos]+price[j];
                                        ans[Next]=ans[pos]*ans[j];
                                }
                                else if (price[pos]+price[j]==price[Next])
                                        ans[Next]+=ans[pos]*ans[j];
                        }
        }
        cout<<price[0]<<' '<<ans[0];
        return 0;
}
```

End

考试因为第二题出了非常弱智的错误导致这后面俩题没时间写
（指调试输出看成结果以为自己程序大错特错然后对着完全正确的程序改了近 $2h$

T3 魔杖

题意

给你一段序列，选出一些没有互相包含的子序列
使他们的和最大

Solve

当时拿到这个题就感觉是个DP，像这种在区间上面乱搞的题要么贪要么DP，但是想了下发现这玩意贪心不好搞，貌似可以举出很多反例，所以就是DP无疑

考场思路

原本是写标准的区间DP，然后发现不能出现区间包含区间的情况，直接搞好像转移比较困难，写不出来

然后就有了第二个想法

因为数据范围只有 $n \leq 1000$ 所以完全可以枚举出所有的子区间（记为seg）

把区间都记录下来，然后假设 $dp[i]$ 是从 0 到 i (而且 终点为 i ，意思是 i 是某条线段的终点) 的魔力和最大值,看起来就好转移多了，因为不包含这个条件就比较好搞了

则

```
dp[seg[i].y]=max(dp[seg[i].y],dp[1~seg[i].y-1]+seg[i].val)
```

因为起点的范围已经被限定，而且进行了一次对线段的预处理（见那个 $sort$），所以就没有重复的问题

Code

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
using namespace std;
#define int long long
int n,low,hi,len[1100],magic[1100];
int prelen[1100],premagic[1100];
struct RANGE
{
        int x,y;
        int val;
        bool operator <(const RANGE&b) const
        {
                if(x==b.x) return y>b.y;
                return x<b.x;
        }
}seg[1100*1100];
int total,ans,dp[1100];
signed main()
{
        cin>>n>>low>>hi;
        for(int i=1;i<=n;i++) cin>>len[i],prelen[i]=prelen[i-1]+len[i];
        for(int i=1;i<=n;i++) cin>>magic[i],premagic[i]=premagic[i-1]+magic[i];
        for(int l=1;l<=n;l++)
                for(int r=l;r<=n;r++)
                        if(low<=prelen[r]-prelen[l-1]&&prelen[r]-prelen[l-1]<=hi)
                        {
                                seg[++total].x=l;
                                seg[total].y=r;
                                seg[total].val=premagic[r]-premagic[l-1];
                        }
        sort(seg+1,seg+1+total);
    //这个sort很重要，没了他正确性就没了 先按照 x 的升序，x相同再 y 的降序排列
    //正确性证明如下
    //1.如果seg[i].x>seg[i-1].x 此时显然不会有冲突
    //2.如果seg[i].x==seg[i-1].x
    //  如果dp没有被更新，显然没有问题
```

```
    //  因为seg[i].y<seg[i-1].y
    //  所以我们这条线段的更新更不到dp[seg[i-1].y]那边去，也不会有冲突
    //故正确性可以保证
        for(int i=1;i<=total;i++)
                for(int j=1;j<seg[i].y;j++)
                        dp[seg[i].y]=max(dp[seg[i].y],dp[j]+seg[i].val);
        if(total==1) {cout<<seg[1].val; return 0;}//这里是防止一下特殊情况
        for(int i=1;i<=n;i++) ans=max(ans,dp[i]);
        cout<<ans;
        return 0;
}
```

正解

我们设一个数组 $f[i][j]$ 代表以 j 为终点， i 为起点的最大值

所以我们对于每一个终点 j ，只需要枚举起点 i 即可

枚举对每一个符合条件的子序列，我们都可以进行比较

Code

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
using namespace std;
#define int long long
int n,low,hi,len[1100],magic[1100];
int dp[1100][1100],prelen[1100],prem[1100];
inline int Max(int a,int b) {return a>b?a:b;}
signed main()
{
        cin>>n>>low>>hi;
        for (int i=1;i<=n;i++) cin>>len[i],prelen[i]=prelen[i-1]+len[i];
        for (int i=1;i<=n;i++) cin>>magic[i],prem[i]=prem[i-1]+magic[i];
        for (int r=1;r<=n;r++)
                for (int l=1;l<=r;l++)
                        if (low<=prelen[r]-prelen[l-1]&&prelen[r]-prelen[l-1]<=hi)
                                dp[l][r]=Max(
                                        dp[l][r-1],
                                Max(
                                        dp[l-1][r],
                                        dp[l-1][r-1]+prem[r]-prem[l-1]
                                        ));
                        else dp[l][r]=Max(dp[l][r-1],dp[l-1][r]);
        cout<<dp[n][n];
        return 0;
}
```

End

T4 魔法阵

题意

魔法阵是一个$n \times m$的格子（高n，宽m），$n \times m$为偶数

Smart手中有n×m个宝石（以1~$n \times m$编号）

Smart从最右上角的格子开始走，从一个格子可以走到上、下、左、右4个相邻的格子，但不能走出边界。每个格子必须且仅能到过1次，这样Smart一共走了n×m个格子停止（随便停哪里）。Smart每进入一个格子，就在该格子里放入一颗宝石。他是按顺序放的，也就是说——第i个进入的格子放入i号宝石。

如果两颗宝石的编号对 $\frac{n \times m}{2}$ 取模的值相同，则认为这两颗宝石相互之间有微妙的影响

也就是说，我们按照宝石的编号对$\frac{n \times m}{2}$取模的值，将宝石分成$\frac{n \times m}{2}$对，其中每对都恰有两颗宝石。对于每一对宝石，设第一颗宝石在第a行第b列，另一颗宝石在第c行第d列，那么定义这2个宝石的魔力影响值为$k1 \times |a-c| + k2 \times |b-d|$

求，在所有合乎题意的宝石摆放方案中，所有成对的宝石间的最大魔力影响值的最小值为多少

Solve

这题没看，所以没有考场思路

这个题要求最大值最下，然而不能二分，因为这个玩意怎么看都没单调性

所以只能爆搜了

但是这题要是直接爆搜能过就有鬼了

根据一笔画的经验，考虑一个剪枝

假设已经搜到x, y

```
if(vis[x+1][y]&&vis[x-1][y]&&!vis[x][y+1]&&!vis[x][y-1]) return ;
if(!vis[x+1][y]&&!vis[x-1][y]&&vis[x][y+1]&&vis[x][y-1]) return ;
```

画一下图就能搞明白这是什么意思了

Code

```
#include<cmath>
#include<iostream>
#include<vector>
#include<cstring>
using namespace std;
int n,m,k1,k2,ans=0x7f7f7f7f,prex[100],prey[100];
bool vis[50][50];
int dx[4]={0,0,1,-1};
int dy[4]={1,-1,0,0};
int qval(int x1,int y1,int x2,int y2) {return k1*abs(x1-x2)+k2*abs(y1-y2);}
void dfs(int x,int y,int s,int val)
```

```
{
        if(vis[x+1][y]&&vis[x-1][y]&&!vis[x][y+1]&&!vis[x][y-1]) return ;
        if(!vis[x+1][y]&&!vis[x-1][y]&&vis[x][y+1]&&vis[x][y-1]) return ;
    int res=0;
    if (s<=n*m/2) prex[s]=x,prey[s]=y;
    else res=qval(x,y,prex[s-n*m/2],prey[s-n*m/2]);
        if (val>=ans) return ;
    if (s==n*m) {ans=min(ans,max(val,res));return ;}
    for (int i=0,nx,ny;i<4;i++)
        {
                nx=x+dx[i];ny=y+dy[i];
                if (nx<1||ny<1||nx>n||ny>m) continue;
        if (!vis[nx][ny])
                {
            vis[nx][ny]=true;
            dfs(nx,ny,s+1,max(val,res));
            vis[nx][ny]=false;
        }
    }
}
int main()
{
    cin>>n>>m>>k1>>k2;
        memset(vis,true,sizeof(vis));
        for(int i=1;i<=n;i++)
                for(int j=1;j<=m;j++)
                        vis[i][j]=false;
    vis[1][1]=true;
    dfs(1,1,1,0);
    cout<<ans;
    return 0;
}
```

End