中北大学ACM-ICPC程序设计
创新实验室

中北大學
NORTH UNIVERSITY OF CHINA

第二届山西省大学生程序设计大赛
暨第十二届中北大学程序设计大赛

题解

2017 年 5 月 20 日

Problem A	2、3、5 木头人
Problem B	Jack and Rose
Problem C	郭姐的数学
Problem D	魔力手环
Problem E	回收大佬之气
Problem F	郭姐相亲
Problem G	跳跃数
Problem H	郭姐的老婆
Problem I	旺大神&郭姐
Problem J	无聊的一天
Problem K	玩数字

Problem A　2、3、5木头人

题解：既然求的是第一个大于等于它的数，那么二分的高效性就可以体现了。把只有2,3,5质因子的数预处理打表排序，大致推一下就知道满足的数不会超过[log2(1e18)*log3(1e18)*log5(1e18)]个，每次询问二分即可。复杂度T*log2(n)+排序复杂度

标程：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#include<string>
using namespace std;
typedef long long ll;
const ll INF=1e18+100;
const double eps=1e-9;
ll qpow(ll a,ll b)
{
    ll ans=1;
    while(b)
    {
        if(b&1)ans=(ans*a);
        b>>=1;a=(a*a);
    }
    return ans;
}
ll vec[1000010];
int cnt;
void init()
{
    cnt=0;
```

```
    for(int i=0;qpow(211,i)<=INF;i++)
    {
        ll s2=qpow(211,i);
        for(int j=0;qpow(311,j)<=INF/s2;j++)
        {
            ll s3=qpow(311,j);
            for(int k=0;qpow(511,k)<=INF/s2/s3;k++)
            {
                vec[cnt++]=s2*s3*qpow(511,k);
            }
        }
    }
    sort(vec,vec+cnt);
}
int main()
{
    init();int t;scanf("%d",&t);
    while(t--)
    {
        ll n;scanf("%lld",&n);
        if(n==1)printf("2\n");
        else
        {
            ll x=lower_bound(vec,vec+cnt,n)-vec;
            printf("%lld\n",vec[x]);
        }
    }
    return 0;
}
```

Problem B Jack and Rose

题解：(1)直接每次跑最小生成树，kruscal 复杂度 q*m*logm，过于暴力，会超时，可以想办法优化一下。（2）一开始没删边的时候直接跑一遍最小生成树，然后把这些边存入 set，以后每次查询，如果删的边不在 set 里，那么就直接输出答案，否则，删边重新跑最小生成树。优化之后就能过题了。（3）当然本题还有更好的解法，把所有询问的边存起来进行离线处理。首先跑一边最小生成树，把最小生成树的边和询问中在最小生成树里的边存起来，假设总数为 num，跑出来一个不完全的生成树，然后每次询问的时候只需要在 num 条边里面删掉查询边，跑最小生成树。这样缩小下规模，性能便大大提升了。复杂度 q*（n+q）*log2(n+q)!其实还有还多解法的，比如类似上面的 nlog（2n）的做法，最小生成树+树形 dp 等。

标程：

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
#include<vector>
using namespace std;
const int maxn  = 1e6+10;
struct node{
    int a,b,val;
    bool operator < (const node &c)const {
        return this->val < c.val;
    }
};
node A[maxn];
node C[maxn];
node B[1010];
int fa[1010];
int vis[1010][1010];
int Ans[1010];
int dp[1010][1010];
vector<int>G[1001];
void Sort(int m)
```

```
{
    int j = 0;
    for(int i = 1; i <=m;i++)
        G[A[i].val].push_back(i);
    for(int i = 0; i <=1000;i++)
        for(int k = 0; k < G[i].size();k++)
            C[++j] = A[G[i][k]];
    for(int i=1;i <=m;i++)
        A[i]=C[i];
    for(int i = 0; i <=1000;i++)
        G[i].clear();
}
bool cmp(node a,node b)
{
    return (a.a < b.a)||(a.a== b.a && a.b<b.b)|| (a.a== b.a && a.b==b.b && a.val
< b.val);
}
void init()
{
    for(int i=0;i<1010;i++) fa[i]=i;
}
int fi(int a)
{
    return a ==fa[a] ? a: fa[a]=fi(fa[a]);
}
void un(int a,int b)
{
    a =fi(a);
    b = fi(b);
    fa[a]=b;
}
int work1(int n,int &m)
{
    int ans = 0;
    int num = 0;
    for(int i =1; i <= m; i++)
    {
        if(fi(A[i].a) != fi(A[i].b))
        {
            A[++num].a=A[i].a;
            A[num].b =A[i].b;
            A[num].val = A[i].val;
```

```
            if(!vis[A[i].a][A[i].b])
            {
                un(A[i].a,A[i].b);
            }
        }

    }
    return num;
}
int work(int n,int m,int a,int b)
{
    init();
    int ans = 0;
    int num = 1;
    for(int i =1; i <= m; i++)
    {
        if(fi(A[i].a) != fi(A[i].b) && (A[i].a !=a ||A[i].b != b) )
        {
            num++;
            un(A[i].a,A[i].b);
            ans = ans + A[i].val;
        }
    }
    return num == n ? ans:-1;
}
void cal(int n,int m,int q)
{
    int i,j;
    memset(dp,0,sizeof dp);
    memset(vis, 0, sizeof vis);
    for(i=1;i<=q ;i++)
        vis[B[i].a][B[i].b]=1;
    Sort(m);
    m = work1(n,m);
    for(i=1;i<=q;i++)
    {
        Ans[B[i].val] =dp[B[i].a][B[i].b]==0 ?
dp[B[i].a][B[i].b]=work(n,m,B[i].a,B[i].b) : dp[B[i].a][B[i].b];
    }
}
int main()
{
```

```
    int n,m,q;
    while(scanf("%d%d",&n,&m)!=EOF)
    {
        int i,j,a,b,c;
        init();
        if(n==m &&n ==0)
            break;

        for(i=1;i<=m;i++)
        {
            scanf("%d%d%d",&A[i].a,&A[i].b,&A[i].val);
            if(A[i].a > A[i].b)
                swap(A[i].a,A[i].b);
        }
        scanf("%d",&q);
        for(i=1;i<=q;i++)
        {
            scanf("%d%d",&B[i].a,&B[i].b);
            B[i].val = i;
            if(B[i].a > B[i].b)
            swap(B[i].a,B[i].b);
        }
        cal(n,m,q);
        for(i=1;i<=q;i++)
            printf("%d\n",Ans[i]);
    }
    return 0;
}
```

Problem C 郭姐的数学

思路：自己写一组数据，就能推出公式了 :6/(n*(n-1)*(n-2))

标程：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#include<string>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
int main()
{
    double n;
    while(scanf("%lf",&n)!=EOF)
    {
        printf("%.12f\n",6.0/n/(n-1)/(n-2));
    }
    return 0;
}
```

Problem D 魔力手环

思路：题面很简单，很容易推出递推式。但是 k 过于大，可以用矩阵快速幂优化。但是这样还是会超时，矩阵的相乘复杂度 n^3，但是这个矩阵有个很有趣的性质，从第二列开始，每一列的矩阵值都是由上一列整体向右推一位得到的。这样的话如果不细心还是过不了。再看初值矩阵，除了第一列其他都是 0，所以和初值矩阵有关的运算也可以从 n^3 优化 n^2,然后就轻松过题了。。复杂度 n^2*log2(k)；

初值矩阵就是给的序列，递推矩阵（拿四维举例，其他同理）

1 0 0 1

1 1 0 0

0 1 1 0

0 0 1 1

标程：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#include<string>
using namespace std;
typedef long long ll;
typedef pair<int,int>P;
const int INF=0x3f3f3f3f;
struct Matrix
{
    int a[205][205];
```

```
    Matrix(){memset(a,0,sizeof(a));}
};
Matrix tmp,pre;
int num[205];
int n,k;
void multi1(Matrix p1,Matrix p2)
{
    Matrix ans;
        for(int j=1;j<=n;j++)
        {
            for(int m=1;m<=n;m++)
            {
                ans.a[1][j]=(ans.a[1][j]+p1.a[1][m]*p2.a[m][j])%100;
            }
        }
    pre=ans;
}
void multi2(Matrix p1,Matrix p2)
{
    Matrix ans;
    for(int j=1;j<=n;j++)
    {
        for(int i=1;i<=n;i++)
        {
            ans.a[1][j]=(ans.a[1][j]+p1.a[i][j]*p2.a[1][i])%100;
        }
    }
    for(int i=2;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            if(j==1)ans.a[i][j]=ans.a[i-1][n];else ans.a[i][j]=ans.a[i-1][j-1];
        }
    }
    tmp=ans;
}
void qpow()
{
    while(k)
    {
        if(k&1)
            multi1(pre,tmp);
```

```
        k>>=1;
        multi2(tmp,tmp);
    }
}

int main()
{
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++){if(i==1)tmp.a[1][n]=tmp.a[1][1]=1;else
tmp.a[i][i]=tmp.a[i][i-1]=1;}
    for(int i=1;i<=n;i++){scanf("%d",&num[i]);pre.a[1][i]=num[i];}
    qpow();
    for(int i=1;i<=n;i++){printf("%d%c",pre.a[1][i],i==n?'\n':' ');}
    return 0;
}
```

Problem E 回收大佬之气

思路：图论简单题，先更新好大佬之气，完后记忆化搜索跑最短路,不断更新题意要求的最优的路径就好了。这个题的一个坑点是卡 vector，你是因为这个卡 TLE 卡到怀疑人生的么？复杂度 n+nlogn+min(n^3,100000*n),bfs+最短路+更新路径（复杂度主要在更新路径）

标程 1（记录路径最后比较的方法，更快）

```
#include<cstdio>
#include<iostream>
#include<queue>
#include<algorithm>
#include<cmath>
#include<cstring>
#include<vector>
#include<fstream>

using namespace std;
const int maxn = 900+10;
const int inf = 0x3f3f3f3f;

int p[maxn];
int head[maxn];
int vis[maxn];
int N,M,L,K,S,T;
int map[maxn][maxn];
int dis[maxn];
vector<int>path[maxn*(maxn-1)/2];

struct node {
    int t, c;
    int next;
    node() {
    }
    node (int _t, int _c, int _ne) {
        t = _t;
```

```
        c = _c;
        next = _ne;
    }
}edge[maxn*(maxn-1)/2];

int cnt = 0;
void add(int u, int v, int c) {
    edge[cnt].t = v;
    edge[cnt].c = c;
    edge[cnt].next = head[u];
    head[u] = cnt;
    cnt++;
}

typedef pair<int, int> P;
vector<int>pa[maxn*(maxn-1)/2];

void bfs(int s) {
    queue<P>q;
    q.push(make_pair(s, 0));
    double d = 1.0/(L*1.0);
    p[s] += p[s];
    vis[s] = 1;
    while (!q.empty()) {
        P now = q.front();
        q.pop();
        int u = now.first;
        int c = now.second;

        for (int i=head[u]; i!=-1; i=edge[i].next) {
            int v = edge[i].t;

            if (!vis[v] && c+1 < L)
                q.push(make_pair(v, c+1)), vis[v]=1,
p[v]+=ceil(p[v]*(1-d*(c+1)));

        }
    }
}

void Dijkstra(int n, int v, int *dist, vector<int> *prev, int c[maxn][maxn])
{
```

```
    bool s[maxn];    // 判断是否已存入该点到S集合中
    for(int i=0; i<n; ++i)
    {
        dist[i] = c[v][i];
        s[i] = 0;     // 初始都未用过该点
        if(dist[i] < inf)
            prev[i].push_back(v);
    }
    dist[v] = 0;
    s[v] = 1;
    for(int i=2; i<=n; ++i)
    {
        int tmp = inf;
        int u = v;
        // 找出当前未使用的点j的dist[j]最小值
        for(int j=0; j<n; ++j)
            if((!s[j]) && dist[j]<tmp)
            {
                u = j;              // u保存当前邻接点中距离最小的点的号码
                tmp = dist[j];
            }
        s[u] = 1;    // 表示u点已存入S集合中
        // 更新dist
        for(int j=0; j<n; ++j)
            if((!s[j]) && c[u][j]<inf)
            {
                int newdist = dist[u] + c[u][j];
                if(newdist <= dist[j])
                {
                    if (newdist < dist[j]) {
                        prev[j].clear();
                        dist[j] = newdist;
                    }
                    prev[j].push_back(u);
                }
            }
    }
}

int pcnt = 0;
void searchPath(vector<int> *prev, int v, int u, int sta[], int len) {
    if (u == v) {
```

```
        pcnt++;
        pa[pcnt].push_back(S);
        return ;
    }
    sta[len] = u;
    for (int i = 0 ; i < prev[u].size(); ++i ) {
        if (i > 0) {
            for (int j = len - 1  ; j >= 0 ; --j) {
                pa[pcnt].push_back(sta[j]);
            }
        }
        searchPath(prev, v, prev[u][i], sta, len + 1);
        pa[pcnt].push_back(u);
    }
}

int main() {
    cnt = 0;
    memset(head, -1, sizeof(head));
    for (int i=0; i<N; i++)
        path[i].clear(), pa[i].clear();
    memset(map, inf, sizeof(map));
    memset(vis, 0, sizeof(vis));
    scanf("%d%d%d%d%d%d", &N, &M, &L, &K, &S, &T);

    for (int i=0; i<N; i++)
        scanf("%d", &p[i]);

    for (int i=0; i<M; i++) {
        int u, v, c;
        scanf("%d%d%d", &u, &v, &c);
        if (map[u][v] != inf)
            cout<<"map"<<endl;
        add(u, v, c);
        add(v, u, c);
        map[u][v]=map[v][u]=c;
    }
    bfs(S);

    Dijkstra(N, S, dis, path, map);
    int sta[maxn];
    pcnt = 0;
```

```
    pa[pcnt].push_back(S);

    searchPath(path, S, T, sta, 0);
    int ans = 0, res=inf, pos=-1;
    for (int i=1; i<=pcnt; i++){
        int asum = 0, rsum = 0;
        for (int j=0; j<pa[i].size(); j++)
            asum += p[pa[i][j]], asum %= K;
        int nn = 0;
        if (pa[i].size()%2 == 0)
            nn = (int)pa[i].size()/2;
        else
            nn = (int)pa[i].size()/2+1;
        for (int j=nn; j<pa[i].size(); j++)
            rsum += p[pa[i][j]];

        if (asum > ans) {
            ans = asum;
            res = rsum;
            pos = i;
        }
        else if (asum == ans && rsum < res){
            ans = asum;
            res = rsum;
            pos = i;
        }
    }

    if (dis[T] >= inf)
        puts("-1");
    else {
        printf("%d %d %d %d\n", pcnt, dis[T], ans, res);
        for (int i=0; i<pa[pos].size(); i++)
            if (i == 0)
                printf("%d", pa[pos][i]);
            else
                printf("->%d", pa[pos][i]);
        printf("\n");
    }

    return 0;
}
```

例程 2：（直接记忆化搜索来更新路径，慢一点，但可以过题）

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#include<string>
using namespace std;
typedef long long ll;
typedef pair<int,int>P;
const int INF=0x3f3f3f3f;
int n,m,l,k,s,t,a,b,c;
double per;
struct node{int b,c,next;}edge[250010];
int v1[500],v2[500];
int cnt1=0,cnt2=0;
int gas[500],dis[500],vis[500];
int head[500];
int Sum,Gas,Gasbehind,Dis,cnt;
void add(int a,int b,int c)
{
    edge[cnt].c=c;
    edge[cnt].b=b;
    edge[cnt].next=head[a];
    head[a]=cnt++;
}
void init()
{
    scanf("%d%d%d%d%d%d",&n,&m,&l,&k,&s,&t);
    cnt=0;vis[s]=1;memset(head,-1,sizeof(head));
    for(int i=0;i<n;i++)scanf("%d",&gas[i]);
    while(m--){scanf("%d%d%d",&a,&b,&c);add(a,b,c);add(b,a,c);};
}
void min_road()
{
    memset(dis,INF,sizeof(dis));dis[s]=0;
```

```
    priority_queue<P,vector<P>,greater<P> >q;q.push(P(0,s));
    while(!q.empty())
    {
        P p=q.top();q.pop();int v=p.second;if(dis[v]<p.first)continue;for(int
i=head[v];i!=-1;i=edge[i].next)
        {node
e=edge[i];if(dis[e.b]>dis[v]+e.c){dis[e.b]=dis[v]+e.c;q.push(P(dis[e.b],e.b));}
}
    }
}
void bfs()
{
    queue<P>q;q.push(P(s,1));
    while(!q.empty())
    {
        P p=q.front();q.pop();if(l<=p.second)break;int v=p.first;
        for(int i=head[v];i!=-1;i=edge[i].next)
        {node
e=edge[i];if(!vis[e.b]){vis[e.b]=1;gas[e.b]+=ceil(gas[e.b]*(1-p.second)*1.0/1);
q.push(P(e.b,p.second+1));}}
    }
}
void dfs(int u,int sum,int gass)
{
    if(u==s)
    {
        if(gass%k>Gas){Sum++;Gas=gass%k;for(int
i=0;i<cnt2;i++){v1[i]=v2[i];}cnt1=cnt2;
            int summ=0;for(int
i=0;i<cnt2/2;i++){summ+=gas[v2[i]];}Gasbehind=summ;return;}
        else if(gass%k==Gas){Sum++;int summ=0;for(int
i=0;i<cnt2/2;i++){summ+=gas[v2[i]];}
        if(summ<Gasbehind){Gasbehind=summ;for(int
i=0;i<cnt2;i++){v1[i]=v2[i];}cnt1=cnt2;;return;}}
        else{Sum++;return;}
    }
    for(int i=head[u];i!=-1;i=edge[i].next)
    {
        node
e=edge[i];if(!vis[e.b]&&(dis[e.b]+e.c+sum==Dis)){v2[cnt2++]=e.b;vis[e.b]=1;dfs(
e.b,sum+e.c,(gass+gas[e.b])%k);vis[e.b]=0;
        cnt2--;}
```

```
    }
}
int main()
{
    init();gas[s]*=2;bfs();min_road();Dis=dis[t];
    if(Dis>=INF){printf("-1\n");return 0;}
    Gas=Sum=0;Gasbehind=INF;
    memset(vis,0,sizeof(vis));vis[t]=1;v2[cnt2++]=t;dfs(t,0,gas[t]);
    printf("%d %d %d %d\n",Sum,Dis,Gas,Gasbehind);
    for(int i=cnt1-1;i>=1;i--){printf("%d->",v1[i]);}printf("%d\n",t);
    return 0;
}
```

Problem F 郭姐相亲

思路：模拟题，注意细节和优化。

标程：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#include<string>
using namespace std;
typedef long long ll;
typedef pair<char,char>P;
const int INF=0x3f3f3f3f;
const double eps=1e-9;
char str[1001];
int vis[1001];
struct node1
{
    int id,a;
}node[101],node2[101];
bool cmp(node1 p1,node1 p2){return p1.id<p2.id;}
vector<P>vec;
int main()
{
    int k;scanf("%s",str+1);scanf("%d",&k);
    int len=strlen(str+1);int cnt1=0,cnt2=0;
    for(int i=1;i<=len;i++)
    {
        if(str[i]>='a'&&str[i]<='z')node[cnt1++].id=str[i];
        else node2[cnt2++].id=str[i];
```

```
}
sort(node,node+cnt1,cmp); sort(node2,node2+cnt2,cmp);
int len2=len,tmp1=0,tmp2=0,t=1;
int flag=0,flag2=0,flag3=0;
while(len2>1)
{
    int step=(k)%len2;
    if(step==0)step=len2;int sum=0;
    for(int i=t;step;i++)
    {
        sum++;
        if(sum>50)break;//循环节
        if(i>len)i=1;
        if(!vis[str[i]]){step--;t=i;}
    }
    vis[str[t]]=1;
    if(str[t]>='A'&&str[t]<='Z')
    {
        while(tmp1<cnt1&&vis[node[tmp1].id])tmp1++;
        if(tmp1>=cnt1)break;
        if(tmp1<cnt1)
        {
            vis[node[tmp1].id]=1;flag=1;len2-=2;
            vec.push_back(P(str[t],node[tmp1++].id));
        }
    }
    if(str[t]>='a'&&str[t]<='z')
    {
        while(tmp2<cnt2&&vis[node2[tmp2].id])tmp2++;
        if(tmp2>=cnt2)break;
        if(tmp2<cnt2)
        {
            vis[node2[tmp2].id]=1;flag=1;len2-=2;
            vec.push_back(P(str[t],node2[tmp2++].id));
        }
    }
    sum=0;
    for(int i=t+1;;i++)
    {
        sum++;
        if(sum>50)break;//循环节
        if(i>len)i=1;
```

```
            if(!vis[str[i]]){t=i;break;}
        }
    }
    if(!flag){printf("-1\n");}
    else
    {
        for(int i=0;i<vec.size();i++)
            printf("%c%c%c",vec[i].first,vec[i].second,i+1==vec.size()?'\n':'
');
    }
    return 0;
}
```

Problem G 跳跃数

思路：数位 DP。按照题意的条件，记忆化搜索即可。

标程：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#include<string>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
const double eps=1e-9;
int a[12];
int dp[12][12];
int dfs(int pos,int pre,int flag2,int flag)
{
    if(pos<0)return 1;
    if(!flag&&dp[pos][pre]!=-1&&!flag2)return dp[pos][pre];
    int ans=0,up=flag?a[pos]:9;
    for(int i=0;i<=up;i++)
    {
        if(!flag2&&abs(i-pre)<2)continue;
        ans+=dfs(pos-1,i,flag2&&i==0,flag&&i==up);
    }
    if(!flag&&!flag2)dp[pos][pre]=ans;
    return ans;
}
int solve(int x)
{
    if(x<0)return 0;
    int pos=0;
```

```
    while(x>0)
    {
        a[pos++]=x%10;x/=10;
    }
    return dfs(pos-1,0,1,1);
}
int main()
{
    int l,r;memset(dp,-1,sizeof(dp));
    while(scanf("%d%d",&l,&r)!=EOF)
    {
        printf("%d\n",solve(r)-solve(l-1));
    }
    return 0;
}
```

Problem H　郭姐的老婆

思路：简单推一推公式 n^2-2,注意开 long long;

标程：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
#include<string>
using namespace std;
typedef long long ll;
const int INF=0x3f3f3f3f;
int main()
{
    ll x;
    while(scanf("%lld",&x)!=EOF)
    {
        printf("%lld\n",x*x-2);
    }
    return 0;
}
```

Problem I 旺大神&郭姐

思路：贪心+优先队列。思想就是到了一个地方如果还有兴趣，那就把这个点存入堆，否则，就取堆顶元素，直到兴趣为正为止。复杂度 n*log2(n)

标程：

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<queue>
#include<algorithm>
using namespace std;
const int maxn = 1e5+100;
int n,l;
struct node
{
    int a,b;
    bool operator <(const node &a)const {
        return this->a< a.a;
    }
};
node A[maxn];
priority_queue<int>q;
int main()
{
    while(scanf("%d%d",&n,&l)!=EOF)
    {
        int i,j,k,sum;
        scanf("%d",&sum);
        for(i=1;i<=n;i++)scanf("%d%d",&A[i].a,&A[i].b);
        sort(A+1,A+1+n);
        int last=0,ans = 0;
        bool flag = true;
        for(i=1;i<=n;i++)
        {
            if(A[i].a==0) q.push(A[i].b);
            else break;
        }
        for(;i<=n; i=j)
```

```
        {
            sum = sum-(A[i].a - last);
            while(sum<0&& q.size() > 0)
            {
                sum+=q.top();q.pop();ans++;
            }
            if(sum<0 )
            {
                flag = false;break;
            }
            last = A[i].a;
            for(j=i;j<=n&&A[j].a==A[i].a;j++)q.push(A[j].b);
        }
        sum-= 1 - last;
        while(sum <0 && q.size() >0)
        {
            sum+=q.top();q.pop();ans++;
        }
        if(sum<0||!flag)printf("Wang dashen, I give up!\n");
        else printf("%d\nWang dashen, Let's have a long talk!\n",ans);
        while(!q.empty()) q.pop();
    }
    return 0;
}
```

Problem J 无聊的一天

思路：模拟建树，记录每个点的父亲，儿子和深度即可，只要深度相同，就说明他们是兄弟。

复杂度 m*log2(n)；

标程：

```
#include<cstdio>
#include<iostream>
#include<cmath>
#include<vector>
#include<cstring>
#include<set>
#include<algorithm>
#include<map>
#include<queue>
using namespace std;
const int maxn = 1000000+10;
struct node {
    int p,l,r,c,h;
    node(){p = -1;l = -1;r = -1;c = 0;h = 0;}
}p[maxn];
int A[maxn];
void add(int x, int root, int &N) {
    int q = 0, pre=-1;
    while (q != -1) {
        pre = q;
        if (x > p[q].c)
            q = p[q].r;
        else if (x < p[q].c)
            q = p[q].l;
    }
    p[N].c = x;
    p[N].h = p[pre].h+1;
    p[N].l = -1;
    p[N].r = -1;
    p[N].p = pre;
    if (x > p[pre].c)p[pre].r = N;
```

```
    else p[pre].l = N;
    N++;
}

int main() {
    int n, m;
        scanf("%d%d", &n, &m);
        int N = 1;
        for (int i=0; i<n; i++) {
            int x;
            scanf("%d", &x);
            if (i == 0)
                p[0].c = x, A[x] = 0;
            else
                add(x, 0, N), A[x] = N-1;
        }
        for (int i=0; i<m; i++) {
            int x, y;
            scanf("%d%d", &x, &y);
            x = A[x], y=A[y];
            if (p[x].h == p[y].h)
                puts("brother");
            else if (p[x].p == y)
                puts("son");
            else if (p[y].p == x)
                puts("father");
            else
                puts("no connection");
        }
    return 0;
}
```

Problem K 玩数字

思路：DP 一定会超时，需要贪心。贪心的策略就是先选一个最小值，然后把这个最小值缩点。怎么缩呢？举个例子，-4 -5 -4；三个数，先取-5，然后把这个点变成-4+（-4）-（-5），然后把这个点-3 放进去，再一次如果取到了-3，就相当于取了-8，即-4+（-4），相当于就是取了两边没取中间，是不是恍然大悟，感到巧妙至极啊？然后注意处理边界就好了。如果是边界上的点，直接把旁边的数也删了就好了，想想为什么！还有就是删数的时候注意细节，这道题就解决了！

标程：

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
#include<stack>
#include<queue>
#include<vector>
#include<set>
#include<map>
using namespace std;
typedef long long ll;
typedef pair<ll,int> P;
const int INF=0x3f3f3f3f;
const int mod=1e9+7;
struct node1
{
    ll a;
    int l,r;
}node[100010];
set<P>s;
void Delete(int now)
{
    int l=node[now].l;
    int r=node[now].r;
```

```
    node[l].r=r;
    node[r].l=l;
    s.erase(s.find(P(node[now].a,now)));
}

int main()
{
    int n,k;
    while(scanf("%d",&n)!=EOF)
    {
        s.clear();ll ans=0,x;
        for(int i=1;i<=n;i++)
        {
            scanf("%lld",&x);
            s.insert(P(x,i));
            node[i].a=x;node[i].l=i-1;node[i].r=i+1;
        }
        scanf("%d",&k);
        while(k)
        {
            int id=s.begin()->second;
            ans+=s.begin()->first;k--;s.erase(s.begin());
            if(node[id].l<1)
            {
                int tmp=node[id].r;
                if(tmp<=n)
                {
                    s.erase(s.find(P(node[tmp].a,tmp)));
                    if(node[tmp].r<=n)
                        node[node[tmp].r].l=-1;
                }
            }
            else if(node[id].r>n)
            {
                int tmp=node[id].l;
                if(tmp>0)
                {
                    s.erase(s.find(P(node[tmp].a,tmp)));
                    if(node[tmp].l>0)
                        node[node[tmp].l].r=n+1;
                }
            }
```

```
            else
            {
                int tmp1=node[id].l,tmp2=node[id].r;
                Delete(tmp1),Delete(tmp2);
                node[id].a=node[tmp1].a+node[tmp2].a-node[id].a;
                s.insert(P(node[id].a,id));
            }
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```