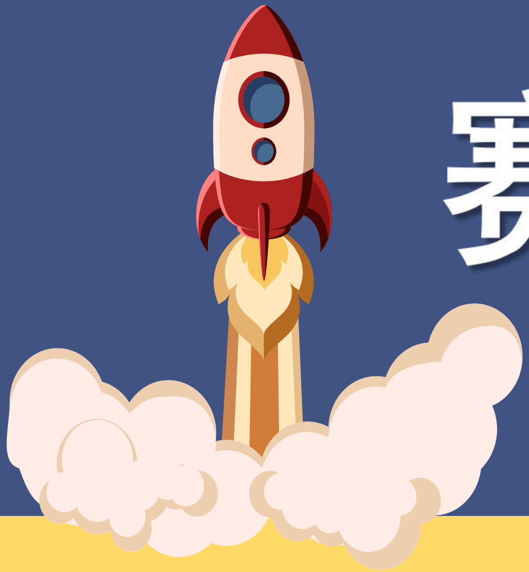


赛题解答与讨论

helang

问题A：组队

假期中，N名南开信竞队的队员打算组队参加编程比赛。
赛会方要求，每队至少3名队员。
我校最多能派出多少只队伍？

$1<=n<=1000$

```
int main()
{
        int N;
        cin>>N;
        cout<<N/3;
}
```

问题B：整除100

给你一个整数N和一个整数D，
请你找出**恰好**能被100整除D次的数字中，第N小的一个。
0<=D<=200
1<=N<=100

考察点：观察，分析规律

除一次100相当与抹掉两个0，整除D次就是抹掉D*2个零。

N是多少就是排名第几小的。直接在N后添加D*2个零就行。
但要注意“恰好”：
当N==100时，排名100的就是101了。

```c
int main()
{
    int D,N;
    scanf("%d%d",&D,&N);
    if(N==100) printf("101");
    else printf("%d",N);
    for(int i=1;i<=D;i++)
        printf("00");
}
```

问题C：吃金币

何老板在玩一款小游戏，游戏虽然简单，他仍乐此不疲。
游戏中，沿着一条笔直公路，分布有N块金币。我们可以把公路当作数轴，把每块金币都看作数轴上的点。其中第i块金币的坐标为 x_i

何老板操控一只青蛙去吃金币，青蛙跳跃运动，若落点处有金币，青蛙就会吃掉它。开始时青蛙位于X位置。游戏玩家可以给青蛙设定一个弹跳值D，每次跳跃的跨度为D。也就是说，若当前青蛙位于Y位置，那么下一个位置可以是Y+D或Y-D。

何老板打算操控青蛙吃掉所有的金币，他想知道D值最大可以设定成多少?

$1 \le N \le 10^5$
$1 \le X \le 10^9$
$1 \le x_i \le 10^9$
所有金币的坐标都不相同

考察点：最大公约数

求出起点与每块金币的距离的最大公约数即可。
时间复杂度O(nlogn)

```
main()
{
    int N,D,Dis,G;
    cin>>N>>D;
    for(int i = 1; i <= N; i++) cin>>a[i]; //a[i]记录第i块金币的坐标
    int G = abs(X - a[1]);                 //G记录公约数
    for(int i = 2; i <= N; i++)
    {
        Dis = abs(X - a[i]);
        G = Gcd(G, Dis);                   //Gcd()用于求最大公约数
    }
    printf("%d", G);
}
```

问题D：奥特曼打怪

何老板最近在玩一款打怪兽的游戏，游戏虽然简单，他仍乐此不疲。
游戏中，何老板操控奥特曼与N只怪兽作战。每只怪兽都有一定的生命值，第i只怪兽的生命值为 h_i
奥特曼可以发射光弹，何老板可以控制奥特曼往一只指定的怪兽发射光弹。怪兽被光弹击中后会损失A单位的生命值。该怪兽被击中的同时，光弹会发生爆炸，使得其他所有怪兽都损失B的生命值。
何老板想知道，奥特曼最少发射几次光弹，就能消灭所有怪兽(怪兽的生命值小于等于0，就被消灭了)。

$1 \le N \le 10^5$
$1 \le h_i \le 10^9$
$1 \le B < A \le 10^9$

考察点：二分答案

二分发射次数Mid

Mid次发射相当于每个怪兽集体减去Mid∗ B的生命值
然后分配Mid个A−B给还未减到0的怪兽

问题D：奥特曼打

```cpp
typedef long long ll;

bool cheak(ll k) {
    ll A = a - b;
    ll Cnt = 0;
    for(int i = 1 ; i <= n ; i++) {
        ll Harm = Monster[i] - b * k;
        if(Harm <= 0) continue;
        else {
            Cnt += Harm / A;
            Cnt += (Harm % A != 0);
            if(Cnt > k) return false;
        }
    }
    if(Cnt > k) return false;
    return true;
}
int main() {
    scanf("%lld%lld%lld" , &n , &a , &b);
    for(int i = 1 ; i <= n ; i++)
        scanf("%lld" , &Monster[i]);
    ll l = 0 , r = 1e9;
    ll ans = r;
    while(l <= r) {
        ll mid = (l + r) >> 1;
        if(cheak(mid)) r = mid - 1 , ans = min(mid , ans);
        else l = mid + 1;
    }
    printf("%lld\n" , ans);
}
```

问题E：基因中有AC

人类基因由碱基'A','C','G','T'构成。
何老板检查了一下自己的DNA，拿到了检查报告。
检测报告里有一段长度为N的由字母'A','C','G','T'构成的表示基因片段的字符串。
何老板发现，自己的基因中带了很多“AC”，众所周知他的口号是“Life is short,AC more!”，于是他想知道对于该基因片段中的任意一段区间[L,R]中，出现了多少次“AC”，请你快速帮他做出回答。

$$1 \le N \le 2*10^5$$
$$1 \le Q \le 10^5$$
$$1 \le L \le R \le N$$

考察点：前缀和

Sum[i]记录字符串1到i位置出现的“AC”的总个数
对于询问[L,R]

Sum[R]-Sum[L]即为答案
时间复杂度O(N)

问题E：基因中有AC

```
int main()
{
    scanf("%s",s+1);
    for(int i = 1; i <= n; ++i) {
        sum[i] = sum[i - 1];
        if(s[i]=='C'&&s[i-1]=='A') {
            ++sum[i];
        }
    }
    while(q--){
        int scanf("%d%d",&l,&r);
        outn(sum[r]-sum[l]);
    }
}
```

问题F：砍树

何老板家门前有一排树，共 3 * N 棵，每棵树都有一定的高度，第i棵树高度为 h_i

何老板需要添置家具，他雇你砍掉其中N棵树，剩下2*N棵。你可以砍掉其中任意N棵树，但何老板有个奇怪的要求，他希望剩下的树中，**前N棵树的高度总和与后N棵树的高度总和之差，尽可能大。**

请你计算出这个最大差值

对于60%的数据 $1 \le N \le 10^3$

对于100%的数据 $1 \le N \le 10^5$

$1 \le h_i \le 10^9$

考察点：堆(优先队列)

显然：前N棵树和后N棵树的分界点一定位于编号[N,2*N]这段区间中。

设该分界位置为i

我们只需找出i左侧最高的N棵树和i右侧最低的N棵树即可。

记i左侧最高N棵树的高度总和为Left[i],i右侧最低N棵树的高度总和为Right[i]

以i为界的答案即是Left[i]-Right[i]

求Left[i]:

1.将1到N号树的树高依次存入小根堆。

2.N+1到i逐个讨论没棵树，对于第i棵树，若高度比堆顶元素大，则删除堆顶元素，将第i棵树树高入堆。

求Right[i]同理，采用大根堆，将2*N+1到3*N号树的树高进堆，从2*N-1到i逐个讨论没棵树，若i的高度低于堆顶，则删除堆顶元素，第i棵树的树高入堆。

Ans=mas{ Left[i]-Right[i] }　　时间复杂度O(nlogn)

问题F：砍树

```
#define LL long long
LL a[maxn],Left[maxn],Right[maxn];
priority_queue<LL, vector<LL>, greater<LL> >Q1;
priority_queue<LL> Q2;

int main(){
        int n;
        LL sum1 =0,sum2=0;
        scanf("%d",&n);

        for(int i=1;i<=3*n;i++)
        {
            scanf("%d",&a[i]);
            if(i<=n) sum1+=a[i], Q1.push(a[i]);
            else if(i>2*n) sum2+=a[i], Q2.push(a[i]);
        }
        Left[n]=sum1;
        Right[2*n+1]=sum2;
```

```
    for(int i=n+1;i<=2*n;i++){
        if(Q1.top()<a[i]){
            sum1+= a[i]-Q1.top();
            Q1.pop();
            Q1.push(a[i]);
        }
        Left[i]=sum1;
    }
    Right[2*n]=sum2;
    for(int i=2*n-1;i>=n;i--){
        if(Q2.top()>a[i+1]){
            sum2=sum2 - Q2.top()+a[i+1];
            Q2.pop();
            Q2.push(a[i+1]);
        }
        Right[i]=sum2;
    }
    LL ans = Left[n]-Right[n];
    for(int i=n+1;i<=2*n;i++){
        ans = max(ans,Left[i]-Right[i]);
    }

    printf("%lld\n",ans);
}
```

问题G：作业积分

何老板给信竞队队员布置了N道习题，队员们可以任选其中k道题作为假期作业。每道题都有两个标签:题目类型和难度系数。第i题的类型为 t_i 难度系数为 d_i

何老板想鼓励队员们刷多种类型的题，刷难题，所以他指定了以下积分策略
作业最终得分为：k道题难度系数总和 + (k道题中的题目类型数) ^{2}

何老板想知道，同学们最高能得到多少积分

$$1 \le K \le N \le 10^5$$
$$1 \le t_i \le N$$
$$1 \le d_i \le 10^9$$

考察点：贪心、栈

1.贪心：优先选择难度系数大的题目
将题目按难度系数由大到小排序，选出前k道。

2.将选中的k道题依次放入栈中，标记选中的的题目种类，记录选中的题目的总分ans。
讨论剩下的k+1到n号题，对于其中的第i题：
1) 若i所属类型的题已出现在栈中了，跳过i;
2) 若i所属类型的题没在栈中，将栈顶元素删除。将i设为选中，此时若选中题目总分超过ans,则更新ans。
对于栈中的某题x，如果该类型是多次出现在栈中，才有可能被换出去。只出现一次的是无法被换掉的。

问题G：作业积分

```
stack<int>s;
int main()
{                                                   //Cnt记录已选题目类型数
    long long ans=0,Sum=0,Cnt=0;//Sum记录已选题目难度系数总和
    scanf("%d%d",&n,&k);
    for(int i=1;i<=n;i++)scanf("%d%d",&a[i].t,&a[i].d);
    sort(a+1,a+1+n);                            //按难度系数由大到小排序
    for(int i=1;i<=n;i++)
    {
        if(i<=k)
        {
            if(!Mark[a[i].t])   //一种新类型的题入栈
            {
                Mark[a[i].t]=true;  //Mark[i]标记i号题型已出现在栈中
                Cnt++;
            }
            else s.push(a[i].d);
            Sum+=a[i].d;
            ans=max(ans,Sum+Cnt*Cnt);
        }
        else
        {
            if(s.empty())break;
            if(Mark[a[i].t]) continue; //该题型栈中已有
            Mark[a[i].t]=true;
            Cnt++;
            Sum+=a[i].d;
            Sum-=s.top();
            s.pop();
            ans=max(ans,Sum+Cnt*Cnt);
        }
    }
```

问题G：取石块游戏

何老板和果老师在玩一款有趣的游戏，游戏虽然简单，他俩仍旧乐此不疲。

游戏开始时有两堆石块，分别有A块和B块石块。两位玩家按照以下操作规则拿取石块：
任选一堆石块，从中拿取 $2*X$ 块石头。然后，扔掉X块，把剩下的X块放到另一堆。注意：取走的 $2*X$ 块石块不能超过该堆石块的总数，玩家可以自由选择整数X（1≤X）。

何老板先手，两人交替进行操作，谁无法进行操作，谁就输掉游戏。

考察点：博弈 观察与分析

甲乙两堆石块数分别是A和B，设A>B ,A-B=K　　若K<=1那么后手必赢
初步考虑：
　先手 从甲堆拿走2*X块后，得到甲乙两堆石块分别是 A-2*X 和 B+X
　后手 从乙堆拿走2*X块后，得到甲乙两堆石块分别是 A-X 和 B-X
经过一轮操作后，甲乙两堆的石块数之差为(A-X)-(B-X)=A-B=K
也就是后手可以保持两堆石头的数量差恒定，即只要先手能操作，后手就能操作。
操作到最后，甲乙两堆石头数就是 K 和 0了。
若K的值为1或0，先手就无法操作，输掉游戏。

问题E：取石块游戏

结论：如果 $|x - y| <= 1$ 那么后手必赢，反之先手必赢。
证明：

假设现在有$|x - y| <=1$，我们不妨设$x > y$,那么$y = x - k$.所以如果我们从x中拿出2i个石子，那么x,y将变为：

$x' = x - 2i$　　　$y' = x - k + i$

这时我们可以发现$y' - x' = -k + 3i$，因为i是非负的，而k因为$|x - y| <=1$,所以$k <=1$,所以3i的大小至少为3，而k最大为1，所以$y' - x'$至少为2，那么这个时候后手一定可以移动。

现在来考虑后手怎么动：

首先我们可以推出一个性质，当一开始的石子数为x， y时，先手取了一步变成了$x - 2i$和$y + i$,这个时候后手只需要从y中拿同样的2i个，就可以使得石子数变为$x - i$和$y - i$,而我们知道，两个数同时减去同一个数，它们的差是不会变化的，所以这个时候的x'和y'依然相差小于等于1，这就转化成了第一种情况

```
int main()
{
    long long x,y;
    cin>>x>>y;
    if(abs(x-y)<=1)cout<<"guo";
    else cout<<"he";
}
```

问题H：平均身高

NK信竞队有N名队员。
因要参加一年一度的全校自编操比赛，何老板组织队员们进行排练。
N名队员站成一排，从左往右身高分别是 $a_1, a_2, \ldots, a_N$

何老板设计了一个炫酷的加分动作。需要选相邻的若干同学出来(连续一段同学)出来表演。但要求选出的这些同学的平均身高需要大于等于K。

何老板想知道，他总共有多少种选择方案。

概括：给出一个整数序列，需要从中选出一个连续子序列，使得该序列的平均值不小于k，求方案数。

$1 \le N \le 2 * 10^5$
$1 \le K \le 10^9$
$1 \le a_i \le 10^9$

考察点：前缀和 离散化 逆序对

1.将a数组每个数都减去k,得到新的数组b
2.对b求前缀和，得到前缀和数组Sum[ ]
3.若a数组区间[x+1,y]的平均值>=k，则Sum[y]-Sum[x]>=0
4.对于Sum数组，从左往右讨论每一个位置i,查找[i+1,n]区间有多少个>=Sum[i]的数
5.要在查找i右边有多少个比它大的数，树状数组处理逆序对
6.本题数值规模较大，需要离散化

问题H：平均身高

```
LL a[maxn], c[maxn*4];
map<LL, int> M;
vector<LL> V;
void Modify(int x, int s){
    for(; x <= n; x += x&(-x)) c[x] += s;
}
LL Query(int y){
    if(y <= 0) return 0;
    LL ans = 0;
    for(int x = y; x; x -= x&(-x)) ans += c[x];
    return ans;
}
```

```
int main()
{
    int x;
    scanf("%d %d", &n, &k);
    for(int i = 1; i <= n; i++) scanf("%d", &x), a[i] = x;
    LL sum = 0;
    for(int i = 1; i <= n; i++) {
        sum += a[i];
        sum -= k;
        V.push_back(sum);
    }
    sort(V.begin(), V.end());
    for(int i = 0; i < V.size(); i++) M[V[i]] = i+1;
    sum = 0;
    LL ans = 0;
    for(int i = 1; i <= n; i++){
        sum += a[i];
        sum -= k;
        ans += Query(M[sum]);
        if(sum >= 0) ans++;
        Modify(M[sum], 1);
    }
    cout<<ans<<endl;
}
```

问题H：*红绿生成树

何老板给你一个有N个点和M条边构成的无向图。
每条边都有一定的长度。一开始所有边都没有被涂颜色，何老板准备了红绿两种颜料，你可以将边涂成红色或绿色。
何老板要你将图中每条边都涂上颜色。但要求该**图中存在既有红边又有绿边的生成树**。且这些生成树中，边长总和最小的一棵恰好为X。
何老板问，有多少种满足上述要求的涂色方案？mod 10^9+7后再输出。

$1 \le N \le 1000$
$1 \le M \le 2000$
$1 \le W \le 10^9$
$1 \le X \le 10^{12}$
给出的图是连通的，且不含重边和自环

考察点：最小生成树 快速幂 组合数学

问题H：*红绿生成树

首先，我们求一棵原图的最小生成树，设它的边长总和为 t

1.t>x　显然x大于t是无解的！

2.t==x

当t==x的时候，只要某条边在任意一棵最小生成树上，我们可以随意染色，只要不是同一种颜色即可。而对于不在任意一棵最小生成树上的边，是可以随便选的。所以设在最小生成树上的边有a条，不在的有b条，那么对于t==x的情况，总方案数就为**$(2^a-2)*(2^b)$**

（在最小生成树上的随便选 - 全部涂红和全部涂绿两种方案)×(不在最小生成树上的随便选）

3.t<x

讨论3种类型的边：

类型1：对于任意一条边，包含它的最小生成树总长<x

我们都要把它们染成同一种颜色。因为如果它们中某一条颜色与另一条颜色不同了，这些边就能组合成一棵生成树了，且该生成树总长<x，与题设不符。

类型2：对于任意一条边，包含它的最小生成树总长==x

对于这些边，可以随便选颜色，但是类型2和 类型1边的不能全部涂同一种颜色，因为类型1肯定和类型2组合成最小生成树，

类型3：对于任意一条边，包含它的最小生成树总长>x

随便染色，反正也选不到它们

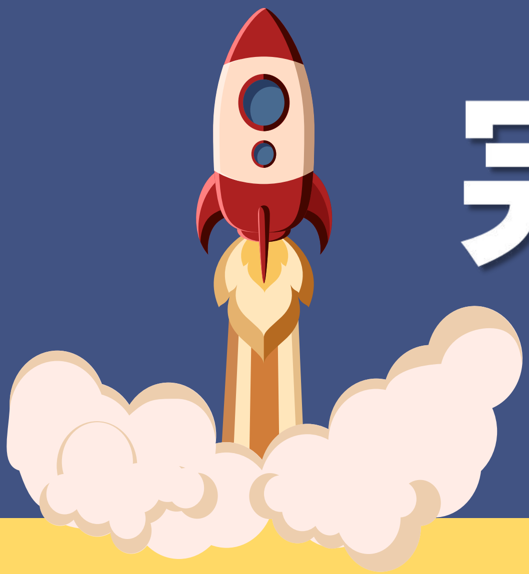
所以设在最小生成树上的边有a条，在大于x的最小生成树上的有b条，总方案数就为:**$2*(2^a-1)*2^b$**

(类型1全涂红或全涂绿共2种方案) × (类型2随意染-全部涂成与类型1同色的方案) ×（类型3颜色随便选）

问题H：*红绿生成树

若t<X。那么生成树中的边一定都是同色的。考虑加入一条边i，换边求生成树法,边i换原来位于生成树中的边j,
t-a[j]+a[i]==X时，至少有一条边要和之前的同色块不同色，一定是i号边。

怎样算出包含边i的最小生成树的长度？
首先.将i号边加入生成树
然后.根据kruskal原理，在生成树中加入n-2条边即可。



完！

以梦为码 心之所往