

Codeforces Round #666 (Div. 1)

[比赛链接](#)

B. Stoned Game

题意

给定 n 堆石头和两个玩家 A, B A 先手。

每次轮到某个玩家时，该玩家需要选择一个非空且上一轮没有被另一个玩家拿过的堆，拿走一个石头。

如果该玩家无法操作，则该玩家败北。问谁有必胜策略。

题解

设最大的堆有 v 个石头，总共有 s 个石头。

首先不难得出结论：如果 $v > \frac{n+1}{2}$ 则 A 有必胜策略。

否则如果 s 为偶数，设 $s=2k$ 构造石头序列，使得所有属于同一个堆的石头相邻。

例如，设现有 3 个堆，分别有 3, 3, 4 个石头，则得到序列 1, 1, 1, 2, 2, 2, 3, 3, 3, 3。

于是可以保证第 i 个石头与第 $i+n$ 个石头不属于同一个堆，于是将其两两配对。每次 A 选取石头后 B 选择对应石头即可。

所以上述情况 B 有必胜策略。

如果 s 为奇数，设 $s=2k+1$ 根据假设 $v \leq \lfloor \frac{n+1}{2} \rfloor = k$ 于是 A 任意取一个石头后 $s=2k$ 仍然满足 $v \leq \frac{n+1}{2}$

于是该情况转化为 s 为偶数的情况，此时 A 转变为后手，于是 A 有必胜策略。

C. Monster Invaders

题意

现有 n 层楼，每层 a_i 个 1 点 Hp 的怪物和一个 2 点 Hp 的 boos

给定 3 把枪：

1. 手枪，花费 r_1 对一个单体敌人造成 1 点伤害，当该楼层有其他怪物存在时不能攻击 boos
2. 镭射枪，花费 r_2 对该楼层全体敌人造成 1 点伤害
3. AWP ，花费 r_3 对一个单体敌人造成 2 点伤害，当该楼层有其他怪物存在时不能攻击

$\text{\text{\text{boos}}}$

如果对当且楼层 $\text{\text{\text{boos}}}$ 造成伤害但未将 $\text{\text{\text{boos}}}$ 杀死则立刻强制转移到其他楼层。每次转移楼层(不管是主动还是强制转移)均花费 d

问杀死所有怪物(包括 $\text{\text{\text{boos}}}$)的最小费用。(保证 $r_1 \leq r_2 \leq r_3$)

题解

假如某时刻在第 i 层, 第 $i-1$ 层的 $\text{\text{\text{boos}}}$ 还有一点 $\text{\text{Hp}}$ 但 $j \leq i-1$ 的怪物已经全部消灭。

如果将第 $i-1$ 层作为终点, 则必须从第 i 层前往第 $i+2$ 层, 将第 $i+2$ 层及以上的怪物全部消灭。

然后再从第 $i+1$ 层回到第 i 层, 再回到第 $i-1$ 层。所以从此刻起一共需要在第 $i-1$ 到第 $i+1$ 层间转移 3 次。

事实上, 先从第 i 层回到第 $i-1$ 层消灭 $\text{\text{\text{boos}}}$ 再回到第 i 层, 最后前往 $i+1$ 层也只需要转移 3 次。

于是可以用第二种方案替代第一种方案。

另外, 由于第二种方案保证经过第 i 层两次, 于是可以保证杀死第 i 层所有怪物, 于是有 $j \leq i+1$ 的怪物已经全部消灭。

于是只要第 $i+1$ 层存在, 第 $i-1$ 层一定可以不是终点, 且可以认为之前必须先杀死 $j \leq i-1$ 层的所有怪物。

于是只需要考虑将第 n 层和第 $n-1$ 层作为终点的情况。设 $\text{\text{dp}}(i, 0/1)$ 表示已经到达 $i+1$ 层, 第 i 层 $\text{\text{\text{boos}}}$ 还剩余 $0/1 \text{\text{Hp}}$ 的情况。

时间复杂度 $O(n)$

```
const int MAXN=1e6+5;
int a[MAXN];
LL dp[MAXN][2];
int main()
{
    LL n=read_int(), r1=read_int(), r2=read_int(), r3=read_int(), d=read_int();
    _for(i, 0, n) a[i]=read_int();
    _rep(i, 1, n) dp[i][0]=dp[i][1]=1e18;
    dp[1][0]=r1*a[0]+r3, dp[1][1]=min((a[0]+1)*r1, r2);
    _for(i, 1, n){
        dp[i+1][0]=min(dp[i+1][0], d+dp[i][0]+a[i]*r1+r3);
        dp[i+1][0]=min(dp[i+1][0], 3*d+dp[i][1]+r1+min(r2+r1, r1*a[i]+min(2*r1, r3)));
        dp[i+1][1]=min(dp[i+1][1], min(d+dp[i][0], 3*d+r1+dp[i][1])+min(r2, (a[i]+1)*r1));
    }
    enter(min(dp[n][0], 2*d+dp[n-1][1]+r1+a[n-1]*r1+r3));
    return 0;
}
```

```
}
```

From:

<https://wiki.cvbbacm.com/> - **CVBB ACM Team**

Permanent link:

https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:cf_666_div_1 

Last update: **2020/09/04 15:34**