陶陶玩红警 解题报告

江苏苏州中学 宋文杰

1，题目描述

【问题描述】

陶陶最近开始玩红警了，他玩的是自己 MOD 的一个红警版本。这个版本是这样的：你只能造两样东西，战车工厂和坦克。最初你有一个战车工厂，然后在接下来的每一秒内你可以有两种选择(假设当前有 k 个战车工厂)：1，建造一个战车工厂；2，建造 k 辆坦克。注意，战车工厂和坦克是不能同时建造的。

陶陶在玩了一个月红警后，认为自己红警水平很厉害了。于是他向 curimit 发了一份挑战书，他还嚣张地说他会对 curimit 发动 N 次攻击。第 i 次攻击在第 time[i]秒末，陶陶会使用 num[i]辆坦克来进攻，消灭掉 curimit 家里同等数量的坦克。如果此时 curimit 家里没有这么多的坦克的话，那么 curimit 就死翘翘了。

curimit 接到战书后看到看到陶陶如此强大的攻势，被吓得不轻。他想请你帮帮忙，帮他制定一份作战计划（什么时候造战车工厂，什么时候造坦克）：curimit 希望在抵挡了陶陶的 N 轮进攻之后，在第 final 秒末发动最终的总攻击，一举歼灭陶陶的老家。他希望你的作战计划能够在第 final 秒末造出最多数量的坦克。

【输入文件】

输出文件为 tank.in。

第 1 行为两个整数 N，final，含义如题目中所述。

接下来 N 行，第 i 行有两个数 time[i]和 num[i]，含义如题目中所述。

【输出文件】

输出文件为 tank.out。

输出文件仅包含一行，如果 curimit 无法抵挡陶陶的进攻，那么输出“No Answer!”；否则输出第 final 秒末 curimit 最多能拥有多少辆坦克。

【输入样例】

```
3 10
5 3
7 13
9 4
```

【输出样例】

8

【样例解释】

第 1 秒：造战车工厂。
第 2 秒：造战车工厂。
第 3 秒：造战车工厂。
第 4 秒：造坦克，当前坦克数：4。
第 5 秒：造坦克，当前坦克数：8。
第 5 秒：陶陶带了 3 辆坦克冲了进来，当前坦克数：5。
第 6 秒：造坦克，当前坦克数：9。
第 7 秒：造坦克，当前坦克数：13。
第 7 秒：陶陶带了 13 辆坦克冲了进来，当前坦克数：0。
第 8 秒：造坦克，当前坦克数：4。
第 9 秒：造坦克，当前坦克数：8。
第 9 秒：陶陶带了 4 辆坦克冲了进来，当前坦克数：4。
第 10 秒：造坦克，当前坦克数：8。
在第 10 秒末，curimit 家里最多有 8 辆坦克。

【数据规模和约定】

10%的数据中 N≤5，final≤10；
30%的数据中 N≤1000，final≤1000；
100%的数据中 N≤100000，final≤10^9，0≤num[i]≤10^{18}；
100%的数据中 1≤time[1]<time[2]<……<time[n-1]<time[n] ≤final。

2，数学模型

我们把题目稍微变形一下，令

$$T_i = time[i]$$

$$P_i = \sum_{k=1}^{i} num[i]$$

$$m = final$$

然后，我们添加一个限制条件：T_{n+1}=m，P_{n+1}=0

现在的问题就是，给出 n 个限制条件，第 i 个限制条件为：第 T_i 秒末至少造出了 P_i 辆坦克。要求最大化第 T_{n+1}=m 秒末的坦克数。

3，初步分析

如果没有任何限制条件，即只要求最大化第 m 秒末的坦克数，那么怎么做呢？这个问题非常容易，我们很容易知道我们的方案一定是先造若干个战车工厂，再造坦克的。

设我们先花 x 秒来造战车工厂，再花 m-x 秒来造坦克，那么

$ans=(x+1)(m-x)=-x^2+(m-1)x+m$

这是一个开口向下的二次函数，易知它在 $x=\frac{m-1}{2}$ 处取得最大值。

小结：从这个简单的情况中，我们可以发现一点：**战车工厂越早造越好**。

4，深入分析

回到原题，我们假设原问题有解，先忽略最大化第 T_{n+1} 秒末的坦克数这个条件，我们来看如何得到一个满足如下条件的方案：

1，这个方案是所有合法方案中保留战车工厂数目最多的。
2，这个方案是所有满足条件 1 的方案中坦克数目最多的。

在这里，我们选取的贪心算法是：

能造战车工厂就尽量造战车工厂，如果方案不合法了，就减少战车工厂的数目。

我们考虑逐个处理限制条件并调整方案。

假设我们当前已经得到了前 n-1 个时间段的方案。现在加入了第 n 个限制条件。这里就有两种情况：

1，如果当前坦克数≥P_n，那么第 n 个时间段全部用来造战车工厂。
2，否则，我们还是先让第 n 个时间段全部用来造战车工厂。然后我们通过逐个减少战车工厂来调整方案使之满足第 n 个限制条件。

假设当前有 s 个战车工厂，如果我们要去除其中一个战车工厂，那么应该去除哪个呢？

当前的最后一个！

为什么去掉最后一个最优？

我们从两方面来考虑：

1，方案的合法性：如果去除最后一个战车工厂将会导致方案不合法，那么去除之前的任何一个战车工厂还是会导致方案不合法。
2，坦克的数目：去除之前任何一个战车工厂所得到的坦克数目一定没有去除最后一个战车工厂得到的坦克数目多。

因此，去除最后一个战车工厂是最优的。

于是我们得到基本算法：

```
让第 n 个时间段全部用来造坦克。
while  当前坦克数<Pn do
       去除最后一个战车工厂
```

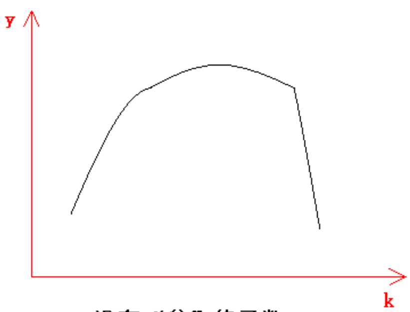
我们设 f(k)=保留前 k 个战车工厂能得到的坦克数。
那么现在的问题就是：找到一个最大的 k 使得 $f(k) \geq P_n$。
这里有两个小问题：
1， 如何计算 f(k)?
2， 如何找满足 $f(k) \geq P_n$ 的最大的 k?

首先来看第一个问题，如何计算 f(k)：
为了能快速地计算 f(k)，我们需要维护两个前缀和：
1， A[i]=前 i 个时间段内的战车工厂数
2， B[i]=前 i 个时间段内的坦克数
我们设第 k 个战车工厂是在第 p 个时间段内第 q 个被建造出来的，那么
$f(k)=B[p-1]+(1+q+A[i])(T_n-T_{p-1}-q)$
计算 f(k)的时间复杂度为 O(1)。

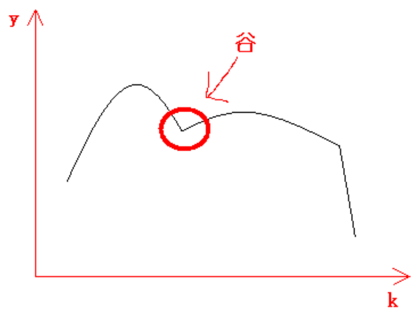
接下来我们来看第二个问题，如何找到满足 $f(k) \geq P_n$ 的最大的 k：
一个很自然的想法就是，我们能不能二分？
不行！因为函数 y=f(k)它不是一个单调函数，反例就是 n=0。
我们注意到 n=0 的时候，这个函数它是一个单峰函数。
于是我们猜想，在这里 y=f(k)它是不是也是一个单峰函数呢？
答案是肯定的。
下面我们就来证明一下这个定理。

【引理】函数 y=f(k)是由 n 段开口向下的抛物线拼接而成的。
证明：我们只需写出函数 y=f(k)的表达式即可。
设第 k 个战车工厂是在第 p 个时间段内被建造出来的。
设前 p-1 个时间段内共造了 c_0 个坦克，v_0 个战车工厂。
那么，$y=f(k)=c_0+k*(T_n-T_{p-1}-(k-v_0))$。
这是一个开口向下的二次函数。
证毕。

【定理】函数 y=f(k)是一个单峰函数。
证明：我们采用反证法。
对于一个多峰连续函数，它必定存在“谷”，如下图所示：



没有“谷”的函数



多峰连续函数，必定存在“谷”

首先，“谷”不可能存在于某段抛物线内，因为抛物线都是开口向下的。所以“谷”只可能存在于某两段抛物线的拼接处。

设“谷”位于第 i 个时间段内的二次函数和第 i+1 个时间段内的二次函数的拼接处。
设 $x_i=a$，$x_{i+1}=b$，前 i-1 个时间段内共造了 c_0 个坦克，v_0 个战车工厂。
当 $v_0+1\le k\le v_0+a$ 时：
$y=c_0+k*(T_n-T_{i-1}-(k-v_0))$
$\ \ =-k^2+(T_n-T_{i-1}+v_0)k+c_0$

对称轴为 $\dfrac{T_n-T_{i-1}+v_0}{2}$

当 $v_0+a\le k\le v_0+b$ 时：
$y=c_0+(v_0+a)*(T_i-T_{i-1}-a)+k*(T_n-T_i-(k-v_0-a))$
$\ \ =-k^2+(T_n-T_i+v_0+a)k+c_0+(v_0+a)*(T_i-T_{i-1}-a)$

对称轴为 $\dfrac{T_n-T_i+v_0+a}{2}$

由“谷”的性质：
在“谷”的左侧附近必定是减函数。
在“谷”的右侧附近必定是增函数。
我们得到关于两段函数对称轴的两个不等式：

$$\begin{cases}\dfrac{T_n-T_{i-1}+v_0}{2}<v_0+a\\[2ex]\dfrac{T_n-T_i+v_0+a}{2}>v_0+a\end{cases}$$

由此，我们得到

$$\frac{T_n-T_i+v_0+a}{2}>\frac{T_n-T_{i-1}+v_0}{2}$$

即：$a>T_i-T_{i-1}$
这显然是不可能的，因为 $a\le x_i\le T_i-T_{i-1}$
矛盾，故原命题成立。

有了以上这个定理后，“寻找满足 $f(k)\ge P_n$ 的最大的 k”，这个问题就非常容易了，伪代码如下：

1，利用三分法找到函数 f(k)的极值点 k_0。
2，若 $f(k_0)<P_n$，那么输出无解，退出。
3，在 $k\ge k_0$ 上，利用二分法找到满足 $f(k)\ge P_n$ 的最大的 k。
4，修改当前方案。

以上算法处理一个限制条件的时间复杂度为 O(logC)。
处理 n 个限制条件的时间复杂度就是 O(nlogC)。

当前我们已经得到了一个满足如下条件的方案：
1， 这个方案是所有合法方案中保留战车工厂数目最多的。
2， 这个方案是所有满足条件 1 的方案中坦克数目最多的。

这是建立在原问题有解的前提之下的，如果原问题无解呢？
那么，当前这个方案就是一个不合法方案！

只需判断一下当前方案是否合法就能知道原问题是否有解了。

现在，我们来看如何求出第 T_n 秒末最多的坦克数。
可以发现，这一步与前面调整方案时的三分法类似，不过这里首先需要做的是找到使得方案合法的最小的 k_0。具体做法很简单，见如下的伪代码：

> 1， 利用二分法求出使得方案合法的最小的 k_0。
> 2， 在 $k \ge k_0$ 上，利用三分法求出 f(k)的最大值，即为答案。

第一步的时间复杂度为 O(nlogC)。
第二步的时间复杂度为 O(logC)。

整个算法的伪代码如下：

> 1， n=n+1 T_n=m P_n=0
> 2， for i=1...n
>   1) 让第 i 个时间段全部用来建造战车工厂。
>   2) 利用三分法求出 y=f(k)的极值点 k_0。
>   3) 如果 $f(k_0)<P_i$，输出无解，退出。
>   4) 在 $k \geqslant k_0$ 上利用二分法找到最大的 k 使得 $f(k) \geqslant P_i$。
>   5) 修改当前方案。
> 3， 如果当前方案不合法，输出无解，退出。
> 4， 利用二分法找到使得方案合法的最小的 k_0。
> 5， 在 $k \geqslant k_0$ 上，利用三分法求出答案。

整个算法的时间复杂度为 O(nlogC)。

5，总结

初看本题，我们很容易想到战车工厂越先造越好，但是由于我们需要最大化第 final 秒末的坦克数，所以战车工厂也不能造太多，在这里我们陷入了一个难点。事实上我们可以先求出一个造出战车工厂数目最多的方案。

本题的另一个难点在于证明函数 y=f(k)是单峰函数，这一点需要选手具有敏锐的洞察力发现这一个性质，还需要选手具有较强的数学分析能力，运用数学工具来证明这个单峰性。