一类单调问题的求解

中山纪念中学 宋新波

例0.老徐真人秀

- 最美杭城人老徐参加一个真人秀，节目共录制N天，每一天节目组都提供同一种美味的苹果若干，老徐有个怪癖，每一天只会吃天数不超过$M\left(1 \le M \le N \le 10^6\right)$天的最美味的苹果。
- 老徐是编程达人，决定通过程序来快速选择。请老徐回答当时是怎么做到的？
- 老徐是大师，当然不会直接告诉你答案，当时随手扔过来下面这个：
- 比如，$N=5$，$M=4$，5天提供的苹果美味度分别为$(80,75,78,73,79)$，老徐的做法如下：

一.单调队列及应用

- **五大要点：**

①维护区间最值：

如例0中，设a_i表示第i天发放苹果的美味度，$f(i)$表示老徐第i天选择的苹果的美味度，

$f(i)=\max\{a_j\}(\max\{1,i-M+1\}\le j\le i)$

②区间出现平移：

如例0中，区间左边界$l_i=\max\{1,i-M+1\}$，右边界$r_i=i$。随着i的增加，右边界r_i递增，左边界l_i非递减，当$i>M$时，l_i递增。查询区间是随着i右移向右平移的。

③去除冗余状态：

如例0中，区间中两个元素a_j与a_k，如果满足$k>j$且$a_k\ge a_j$，a_k跟a_j比“既新鲜又美味”，a_j没有存在的必要，可以 直接删除。

④保持队列单调：

由③得队列中保留的元 素是单调的，如例 0中是单调减的。

⑤最优选择在队首：

如例0中维护单调减的队列， 队列中的最大值在队首 。

一.单调队列及应用

- **代码实现：**

例0中的代码如下：

$st := 1; en := 0;$

$for\ i := 1\ to\ n\ do\ begin$

$\quad while\ (en >= st) and (a[i] >= a[q[en]]) do\ dec(en);$ {删除队尾元素}

$\quad if\ \ i - q[st] = M\ \ then\ inc(st);$ {删除队首元素}

$\quad inc(en); q[en] := i;$ {插入元素i}

$\quad f[i] := a[q[st]];$ {最优答案在队首}

$end;$

- 由于每个元素最多只会进出队列各一次，所以时间复杂度为$O(N)$,优于堆或线段树等数据结构。

例1.[NOIP2010初赛]烽火传递

- **题目描述：**

烽火台又称烽燧，是重要的军事防御设施，一般建在险要或交通要道上。一旦有敌情发生，白天燃烧柴草，通过浓烟表达信息；夜晚燃烧干柴，以火光传递军情，在某两座城市之间有N个烽火台，每个烽火台发出信号都有一定代价。为了使情报准确地传递，在连续M个烽火台中至少要有一个发出信号。

请计算总共最少花费多少代价，才能使敌军来袭之时，情报能在这两座城市之间准确出传递。

- **输入格式：**

第一行：两个整数N,M。其中N表示烽火台的个数，M表示在连续M个烽火台中至少要有一个发出信号。

接下来N行，每行一个数W_i，表示第i个烽火台发出信号所需代价。

- **输出格式：**

一行，表示答案。

- **样例输入：**

5 3
1
2
5
6
2

- **样例输出：**

4

- **数据范围：**

对于50%的数据，$M \le N \le 1000$;

对于100%的数据，$M \le N \le 100{,}000, W_i \le 100$。

例1.[NOIP2010初赛]烽火传递

- **分析:**
- 动态规划：状态$f(i)$表示“在前i个烽火台传递情报且第i个烽火台一定发出信号”所需最小代价。
- 通过考虑“前一个烽火台的位置j”得到以下状态转移方程：

$$f(i)=\begin{cases} W_i & i \le M\text{时} \\ \min\{f(j)\}+w_i\ (i-M \le j \le i-1) & i > M\text{时} \end{cases}$$

- 答案$Ans=\min\{f(i)\}(\max\{N+1-M,1\}\le i \le N)$
- 上式计算$f(i)$时要计算区间最小值，而且区间是向右平移的，如果$f(k)\le f(j)$且$k>j$，可以删除$f(j)$，所以队列中的元素保持单调递增，最优决策在队首。
- 使用单调队列优化，时间复杂度为$O(N)$。

例2.[GDKOI2009]猴子

- **题目描述：**

一个猴子找到了很多香蕉树，这些香蕉树都种在同一直线上，而猴子则在这排香蕉树的第一棵树上。这个猴子当然想吃尽量多的香蕉，但它又不想在地上走，而只想从一棵树跳到另一棵树上。同时猴子的体力也有限，它不能一次跳得太远或跳的次数太多。每当他跳到一棵树上，它就会把那棵树上的香蕉都吃了。那么它最多能吃多少个香蕉呢？

- **输入格式：**

输入第一行为三个整数，分别是香蕉树的棵数N，猴子每次跳跃的最大距离D,最多跳跃次数M。下面N行每行包含两个整数a_i,b_i，分别表示每棵香蕉树上的香蕉数，以及这棵树到猴子所在树的距离。输入保证这些树按照从近到远排列并且没有两棵树在同一位置。b_0总是0。

- **输出格式：**

输出只有一行，包含一个整数，为猴子最多能吃到的香蕉数。

- **样例输入：**

5 5 2
6 0
8 3
4 5
6 7
9 10

- **样例输出：**

20

- **数据范围：**

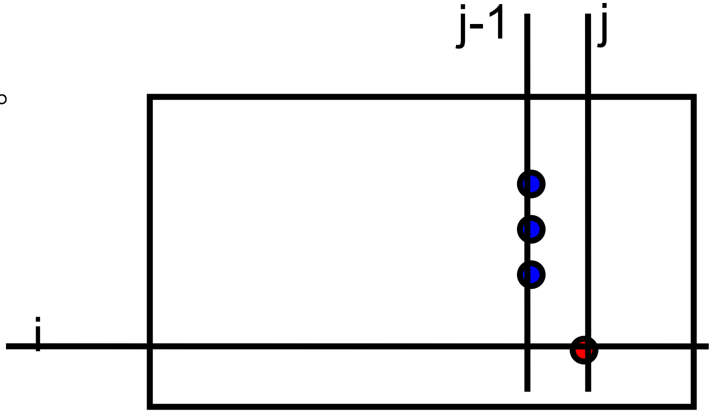
30%：$M < N \le 100$;　50%：$M < N \le 2000$;　100%：$M < N \le 5000, a_i, b_i, D \le 10000$。

例2.[GDKOI2009]猴子

- **分析：**
- 动态规划：状态$f(i,j)$表示从第1棵树不超过j次跳跃跳到第i棵树最多能吃多少香蕉。
- 考虑最后一次跳跃的起点k得到以下状态转移方程：

$$f(i,j)=\begin{cases} a_0 & i=0\text{时} \\ \max\{f(k,j-1)\}+a_i \quad \text{其中}0\le k\le i-1\text{且}b_i-b_k\le D & j>1\text{时} \end{cases}$$

- 答案$Ans=f(N-1,M)$
- 按照列从小到大来处理，计算$f(i,j)$时需要计算第$j-1$列的连续元素的区间最大值，同一列随着i的增加，区间是向下平移的，可以维护一个单调递减的队列。最优值在队首。
- 时间复杂度为$O(NM)$。



例3.多重背包

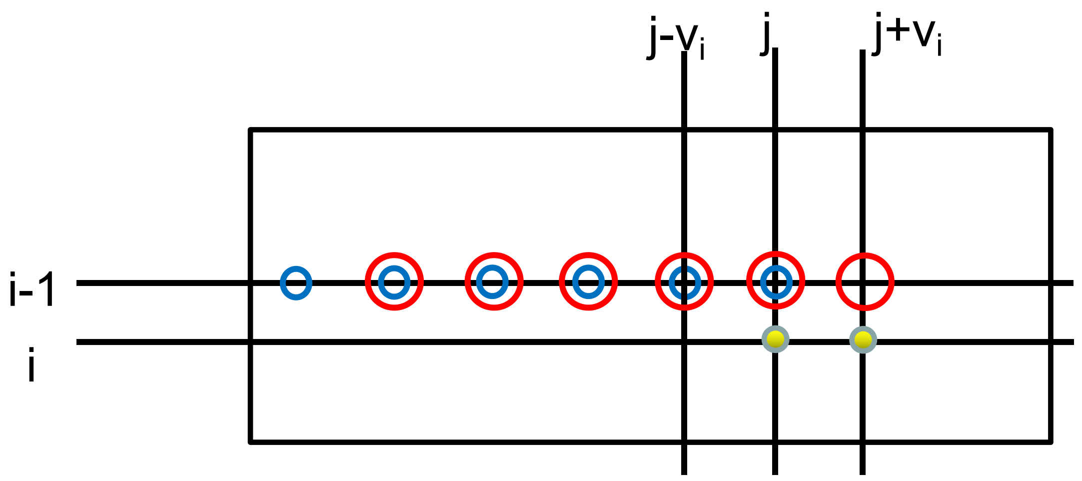
- 有N种物品和一个容量为V的背包。第i种物品最多有m_i件可用，体积是v_i，价值是w_i。选择物品装入背包可使这些物品的体积总和不超过背包容量，且价值总和最大。

例3.多重背包

- **分析：**
- 背包问题，定义状态$f(i,j)$表示用容量为j的背包来装前i个物品能获得的最大价值。
- 通过考虑“第i个物品装几个”得到以下状态转移方程：

$$f(i,j)=\begin{cases}0 & i=0\\ f(i-1,j-x*v_i)+x*w_i & 0\le x\le \min\left\{m_i,\left\lfloor\dfrac{j}{v_i}\right\rfloor\right\}\end{cases}$$

- 时间复杂度为$O\left(V*\sum_{i=1}^{N}m_i\right)$,当然用倍增法优化到$O\left(V*\sum_{i=1}^{N}\log_2^{m_i}\right)$,这里不做介绍
- 观察计算$f(i,j)$和$f(i,j+v_i)$要用到的元素示意图，分别用蓝色和红色标记：



例3.多重背包

•**分析：**

•由上图发现该题可以灵活运用单调队列来优化：

①维护区间最值：

注意这里的区间不是位置连续的元素，如计算$f(i,j)$时涉及的元素是上一行等距的元素，距离为v_i

②区间出现平移：

计算$f(i,j+v_i)$时涉及到的元素组成的区间相对$f(i,j)$在向右平移；

③去除冗余保持单调：

计算$f(i,j)$时，对于两个可选决策x_1,x_2，如果$x_2>x_1$且满足：

$f(i-1,j-x_2*v_i)+x_2*w_i\ge f(i-1,j-x_1*v_i)+x_1*w_i$时，$x_1$可以直接删除。因为对于后面的某一个状态$f(i,j+k*v_i)$来说$f(i,j)$的可选决策$x_1$对应着$f(i,j+k*v_i)$的可选决策$x_1+k$，同样$x_2$对应着$x_2+k$，而不等式

$f(i-1,j+k*v_i-(x_2+k)*v_i)+(x_2+k)*w_i\ge f(i-1,j+k*v_i-(x_1+k)*v_i)+(x_1+k)*w_i$

跟上面的不等式是等价的，x_2还是比x_1优，x_1可以永久删除。

④程序实现：

按i从小到大来处理，一行一行来做，每一行需要建立v_i个单调队列，其中队列0对应背包容量j模v_i等于0的状态，当然也可以把j按照模v_i的值分批次处理以节省空间。

•时间复杂度为$O(NV)$

例4.[HNOI2008]Toy

- **题目描述：**

8月P教授要去看奥运，但是他割舍不下自己的一大堆智力玩具。于是，他决定把所有玩具都运到北京去。P教授使用自己的物体维度压缩器ODZ来给玩具装箱。ODZ可以将任意物品变成一维，再装到一种特殊的一维容器中。P教授有编号为1到N的N件玩具，第i件玩具经过ODZ处理后一维长度是c_i。为了方便整理，P教授要求在一个一维容器中的玩具编号是连续的。同时，如果一个一维容器中有多个玩具，那么相邻两件玩具之间要加入1个单位长度的填充物。形式地说，如果将第i到第j件玩具放在一个容器中，那容器的长度将为$x=j-i+\sum_{k=i}^{j}c_k$。制作容器的费用与容器长度有关，根据P教授的研究，如果容器长度为x，其制作费用为$(x-L)^2$，其中L是一个常量。P教授不关心容器的数目，他可以制造出任意长度的容器(甚至超过L)，但他希望费用最小。

- **输入格式：**

第一行输入两个整数N和L，接下来N行输入c_i。

- **输出格式：**

输出最小费用。

- **样例输入：**

5 4
3
4
2
1
4

- **样例输出：**

1

- **数据范围：**

$1 \le N \le 50000, 1 \le L, c_i \le 10^7$

例4.[HNOI2008]Toy

- **分析：**
- 动态规划：状态$f(i)$表示把前i个玩具装箱所需最小费用，$s(i)$为c_i的前缀和。
- 考虑哪些玩具与玩具i装在同一个容器中，假设是j到i这段连续的玩具，得以下状态转移方程：

$f(i)=\min\left\{f(j-1)+\left(i-j+s(i)-s(j-1)-L\right)^2\right\}$,其中$1\le j\le i$

- 计算$f(i)$时，$i,s(i),L$这些量相当于已知，而含有j的$f(j-1),j,s(j-1)$是未知的，展开平方式，

注意展开过程中参数分离,定义$g(i)=i+s(i)-L,x(j)=j+s(j-1)$。可选决策$j$对应的费用记为$f(i)_j$，

则有 $f(i)_j=f(j-1)+\left(g(i)-x(j)\right)^2=f(j-1)+x(j)^2-2*g(i)*x(j)+g(i)^2$

式子中$f(j-1),x(j)^2,g(i)^2$这三项中i和j这两个参数是完全分离的,$2*g(i)*x(j)$却没有分离，

没有前面几个例子中“$j_2>j_1,f(i)_{j_2}<f(i)_{j_1},j_1$就可以删除”这样明显的单调性，需要深

入的分析。

- 直接做时间复杂度为$O(n^2)$，超时。

例4.[HNOI2008]Toy

- 研究 $j_2 > j_1$ 时 $f(i)_{j_2} < f(i)_{j_1}$ 的前提条件：

$$f(j_2-1)+x(j_2)^2-2*g(i)*x(j_2)+g(i)^2 < f(j_1-1)+x(j_1)^2-2*g(i)*x(j_1)+g(i)^2$$

$$\left(f(j_2-1)+x(j_2)^2\right)-\left(f(j_1-1)+x(j_1)^2\right) < 2*g(i)*(x(j_2)-x(j_1))$$

再令 $y(i)=f(i-1)+x(i)^2$，因 $x(i)=i+s(i-1)$ 是单调递增的，所以有：$\dfrac{y(j_2)-y(j_1)}{x(j_2)-x(j_1)} < 2*g(i)$

- 上面式子左边像 j_1, j_2 两个点形成的斜率，这题要用到新知识：**斜率优化！**

例4.[HNOI2008]Toy

- **斜率优化：**
- 计算$f(i)$时，可选决策$j_2>j_1$，如果j_2比j_1优，令$T(j_1,j_2)=\dfrac{y(j_2)-y(j_1)}{x(j_2)-x(j_1)}$，则必须满足$T(j_1,j_2)<2*g(i)$
- 该题$x(i)$和$g(i)$都是单调的，都是单调增的。下面有两个重要的结论：

- **结论1：计算$f(i)$时，所有可选决策是1到i，可以删除其中的冗余决策，使得队列中从小到大依次存储有价值的可选决策$j_1,j_2,\dots,j_k$，使得这些相邻决策之间的斜率都要大于$2*g(i)$。即：**
$T(j_1,j_2)>2*g(i),T(j_2,j_3)>2*g(i),\dots,T(j_{k-1},j_k)>2*g(i)$。**最优决策是队首元素$j_1$。**

证明：如果队列中相邻两个决策x,y之间的斜率$T(x,y)<2*g(i)$，由于$g(i)$是单调增的，则对于后面的$i_1(i_1>i)$，计算$f(i_1)$时，有：$T(x,y)<2*g(i)<2*g(i_1)\Rightarrow y$永远比$x$优，$x$可以删除。

所以$T(j_1,j_2)>2*g(i),T(j_2,j_3)>2*g(i),\dots,T(j_{k-1},j_k)>2*g(i)$

则对于$f(i)$来说，j_1比j_2优，j_2比j_3优，…，j_{k-1}比j_k优。所以队首j_1是最优的！！

并且在$f(i+1)$时可以在之前维护的队列基础上加入新的决策$i+1$，再继续维护！

例4.[HNOI2008]Toy

- **结论2：可以维护相邻决策之间的斜率单调增**

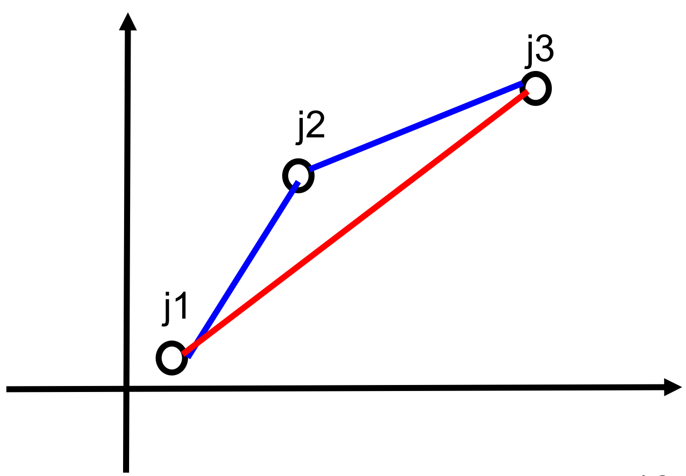
证明：设队列中三个相邻决策 $j_1 j_2 j_3$，假设出现下图所示相邻斜率单调递减的情况。

即 $T(j_1,j_2)>T(j_2,j_3)$，分析 j_2 有没有可能在计算 f 的过程中成为最优决策。

j_2 比 j_1 优 $\Rightarrow T(j_1,j_2)<2*g(i)$

j_2 比 j_3 优 $\Rightarrow T(j_2,j_3)>2*g(i)$

$T(j_2,j_3)>2*g(i)>T(j_1,j_2)$ 出现了矛盾！所以 j_2 不可能成为最优决策，可以删除。得证！



- **综上：**

①应该维护队列中相邻决 策满足：

$$2*g(i)<T(j_1,j_2)<T(j_2,j_3)<\dots<T(j_{k-1},j_k)$$

②最优决策取队首元素

例4.[HNOI2008]Toy

- **程序实现：**
- 在计算f的整个过程中，始终要维护相邻决策满足:$2*g(i)<T(j_1,j_2)<T(j_2,j_3)<\ldots<T(j_{k-1},j_k)$
- 枚举i, 计算$f(i)$, 插入一个新的决策i，相当于在二维平面中插入点i，坐标是$(x(i),y(i))$, 由于$x(i)$递增所以具体程序实现分以下4步：

①**删尾：** 要插入新的可选决策i, 每次选队列最后两个决策j_{k-1},j_k, 如果$T(j_{k-1},j_k)>T(j_k,i)$, 则删除队尾元素j_k。循环做下去，直到队列中的元素个数小于2或者$T(j_{k-1},j_k)<T(j_k,i)$

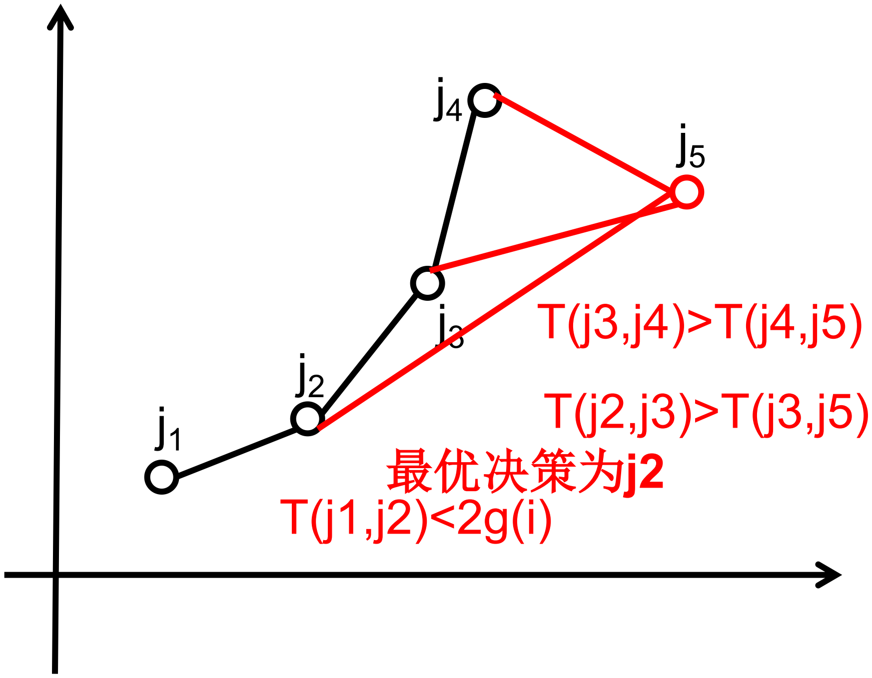
②**插入：** 把新的可选决策i加入队尾；

③**删头：** 取队首两个决策j_1,j_2，如果$T(j_1,j_2)<2*g(i)$，则删除队首元素j_1，循环做下去，直到队列中元素个数小于2或者$T(j_1,j_2)>2*g(i)$;

④**取头：** 最优决策在队首。

例4.[HNOI2008]Toy

- 动画演示：



例4.[HNOI2008]Toy

- **程序代码：**

$f[0]:=0; st:=1; en:=0;$

$for\ i:=1\ to\ n\ do\ begin$

$\quad while(st<=en-1) and (t(q[en-1],q[en])>t[q[en],i])\ do\ dec(en);$ {去尾}

$\quad inc(en); q[en]:=i;$ {插入}

$\quad while(st<=en-1) and (t(q[st],q[st+1])<2*g[i])\ do\ inc(st);$ {删头}

$\quad f[i]:=f[q[st]-1]+\cos t(q[st],i);$ {取头}

$end;$

- 时间复杂度为$O(n)$

例5.[APIO2010]特别行动队

- **题目描述：**

你有一支由n名预备役士兵组成的部队，士兵从1到n编号，要将他们拆分成若干特别行动队调入战场。出于默契的考虑，同一支特别行动队中队员的编号应该连续，即为形如$(i, i+1, \ldots i+k)$的序列。编号为i的士兵的初始战斗力为x_i，一支特别行动队的初始战斗力x为对内士兵初始战斗力之和，即$x = x_i + x_{i+1} + \ldots + x_{i+k}$。通过长期的观察，你总结出一支特别行动队的初始战斗力x将按如下经验公式修正为$x' = a x^2 + bx + c$，其中a, b, c是已知的系数$(a < 0)$。作为部队统帅，现在你要为这支部队进行编队，使得所有特别行动队修正后战斗力之和最大。试求出这个最大和。例如你有4名士兵，$x_1 = 2$，$x_2 = 2$，$x_3 = 3$，$x_4 = 4$。经验公式中的参数为$a = -1, b = 10, c = -20$。此时，最佳方案是将士兵组成3个特别行动队：第一队包含士兵1和士兵2，第二队包含士兵3，第三队包含士兵4。特别行动队的初始战斗力分别为4,3,4，修正后的战斗力分别为4,1,4。修正后的战斗力和为9，没有其他方案能使修正后的战斗力和更大。

- **数据范围：**

$20\% : n \le 1000;$

$50\% : n \le 10{,}000;$

$100\% : 1 \le n \le 1{,}000{,}000, -5 \le a \le -1, |b|, |c| \le 10{,}000{,}000, 1 \le x_i \le 100$

例5.[APIO2010]特别行动队

- **分析：**
- 动态规划：状态$f(i)$表示把前i个士兵拆分成若干个特别行动队能获得的最大修正战斗力。$s(i)$为战斗力x_i的前缀和。
- 通过考虑“跟第i个士兵在同一个行动队的士兵”来进行状态转移，假设是j,则有：

$$f(i)=\max\{f(j-1)+a*(s(i)-s(j-1))^2+b*(s(i)-s(j-1))+c\},\text{其中}1\le j\le i$$

- 展开得：$f(j-1)+a*s(i)^2-2*a*s(i)*s(j-1)+a*s(j-1)^2+b*s(i)-b*s(j-1)+c$
- i已知，j未知。把i和j尽量分离得：

$$\left(f(j-1)+a*s(j-1)^2-b*s(j-1)\right)-2*a*s(i)*s(j-1)+\left(a*s(i)^2+b*s(i)+c\right)$$

令$g(j)=\left(f(j-1)+a*s(j-1)^2-b*s(j-1)\right),h(i)=\left(a*s(i)^2+b*s(i)+c\right)$

$-2*a*s(i)*s(j-1)$这一项无法分离i和j。

- $f(i)=\max\{g(j)+h(i)-2*a*s(i)*s(j-1)\},\text{其中}1\le j\le i$
- 直接做，时间复杂度为$O(n^2)$,可以得50分。

例5.[APIO2010]特别行动队

- **研究** $j_2 > j_1$ 时 $f(i)_{j_2} > f(i)_{j_1}$ **的前提条件：**

$$g(j_2)+h(i)-2*a*s(i)*s(j_2-1)>g(j_1)+h(i)-2*a*s(i)*s(j_1-1)$$

$$g(j_2)-g(j_1)>2*a*s(i)*(s(j_2-1)-s(j_1-1))$$

因初始战斗力 x_i 是正整数，$s(i)=\sum_{j=1}^{i} x_j$ 是单调增的

上式得：$\dfrac{g(j_2)-g(j_1)}{s(j_2-1)-s(j_1-1)}>2*a*s(i)$

- 上面式子左边像 j_1, j_2 两个点形成的斜率，其中 j_1 的坐标是 $(s(j_1-1), g(j_1))$

j_2 的坐标是 $(s(j_2-1), g(j_2))$

- $s(i)$ 是单调的，可以考虑像上一题用斜率优化！

例5.[APIO2010]特别行动队

- **应用斜率优化：**
- 根据上面分析：计算$f(i)$时，可选决策$j_2 > j_1$，令$T(j_1, j_2) = \frac{g(j_2) - g(j_1)}{s(j_2 - 1) - s(j_1 - 1)}$。如果$j_2$比$j_1$优，必须满足$T(j_1, j_2) > 2 * a * s(i)$
- 换句话说，$j_1 < j_2$时，j_1要比j_2优，必须满足$T(j_1, j_2) < 2 * a * s(i)$
- 该题$s(i)$是单调增的，同样有两个重要的结论：

- **结论1：计算$f(i)$时，可以删除冗余决策，使得队列中从小到大依次存储可选决策$j_1, j_2, ..., j_k$，相邻决策之间的斜率都大于$2 * a * s(i)$。即：**$T(j_1, j_2) > 2 * a * s(i), T(j_2, j_3) > 2 * a * s(i), ...,$ $T(j_{k-1}, j_k) > 2 * a * s(i)$。**最优决策是队尾元素$j_k$。**

证明：如果队列中相邻两个决策x, y之间的斜率$T(x, y) < 2 * a * s(i)$, x比y优。由于$s(i)$是单调增的，对于后面的$i_1 (i_1 > i)$，计算$f(i_1)$时，有：$T(x, y) < 2 * a * s(i) < 2 * a * s(i_1) \Rightarrow x$永远比$y$优，$y$可以删除

所以$T(j_1, j_2) > 2 * a * s(i), T(j_2, j_3) > 2 * a * s(i), ..., T(j_{k-1}, j_k) > 2 * a * s(i)$

j_2比j_1优，j_3比j_2优，...，j_k比j_{k-1}优。所以队尾元素j_k是最优的！！

例5.[APIO2010]特别行动队

- **结论2：可以维护相邻决策之间的斜率单调减**

证明：设队列中三个相邻决策 $j_1 j_2 j_3$，假设出现下图所示相邻斜率单调递增的情况。

即 $T(j_1,j_2)<T(j_2,j_3)$，分析 j_2 有没有可能在计算f的过程中成为最优决策。

j_2 比 j_1 优 $\Rightarrow T(j_1,j_2)>2*a*s(i)$

j_2 比 j_3 优 $\Rightarrow T(j_2,j_3)<2*a*s(i)$

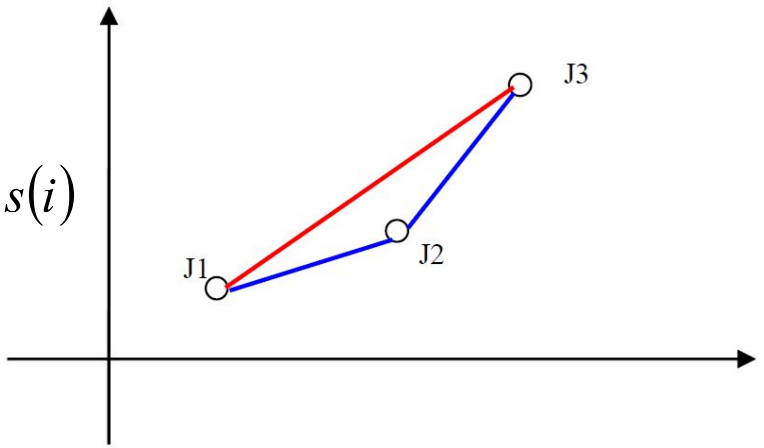
$T(j_1,j_2)>2*a*s(i)>T(j_2,j_3)$，出现了矛盾！所以 j_2 不可能成为最优决策，可以删除。得证！

- **综上：**

①**应该维护队列中相邻决 策满足：**

$T(j_1,j_2)>T(j_2,j_3)>...>T(j_{k-1},j_k)>2*a*s(i)$

②**最优决策取队尾元素**



例5.[APIO2010]特别行动队

- **程序实现：**
- 在计算f的整个过程中，始终要维护相邻决策满足：

$$T(j_1,j_2)>T(j_2,j_3)>\ldots>T(j_{k-1},j_k)>2*a*s(i)$$

- 枚举i, 计算$f(i)$, 插入一个新的决策i，相当于在二维平面中插入点i，坐标是$(s(i-1),g(i))$,

由于$x(i)$递增所以具体程序实现分以下4步：

①**删尾**：要插入新的可选决策i, 每次选队列最后两个决策j_{k-1},j_k, 如果$T(j_{k-1},j_k)<T(j_k,i)$, 则删除队尾元素j_k。循环做下去，直到队列中的元素个数小于2或者$T(j_{k-1},j_k)>T(j_k,i)$

②**插入**：把新的可选决策i加入队尾；

③**删尾**：取队尾两个决策j_{k-1},j_k，如果$T(j_{k-1},j_k)<2*a*s(i)$，则删除队尾元素j_k，循环做下去，直到队列中元素个数小于2或者$T(j_{k-1},j_k)>2*a*s(i)$;

④**取尾**：最优决策在队尾。

- 总时间复杂度为$O(n)$。

二.斜率优化 I 总结

●两个使用前提：

①状态转移方程能通过对比两个可选决策 $j_1 j_2$ 的优劣，使参数分离得到关于斜率的不等式：

如 $T(j_1, j_2) = \frac{y(j_2) - y(j_1)}{x(j_2) - x(j_1)} < g(i)$ 或 $\frac{y(j_2) - y(j_1)}{x(j_2) - x(j_1)} < g(i)$

为了便于分析，我们假设 $j_1 < j_2, j_2$ 是后加入的决策点

②以上斜率不等式中 $x(i)$ 和 $g(i)$ 都是单调的。

二.斜率优化 I 总结

● **四种情况：**

经总结，根据“$T(j_1,j_2)>g(i)$还是$T(j_1,j_2)<g(i)$”和“g的单调性”分为以下四种情况：

①$T(j_1,j_2)<g(i)$，$g(i)$单调增：

维护队列中的相邻决策满足：$g(i)<T(j_1,j_2)<T(j_2,j_3)<\ldots<T(j_{k-1},j_k)$，最优决策取队首元素;

②$T(j_1,j_2)<g(i)$，$g(i)$单调减：

维护队列中的相邻决策满足：$T(j_1,j_2)<T(j_2,j_3)<\ldots<T(j_{k-1},j_k)<g(i)$，最优决策取队尾元素;

③$T(j_1,j_2)>g(i)$，$g(i)$单调增：

维护队列中的相邻决策满足：$T(j_1,j_2)>T(j_2,j_3)>\ldots>T(j_{k-1},j_k)>g(i)$，最优决策取队尾元素;

④$T(j_1,j_2)>g(i)$，$g(i)$单调减：

维护队列中的相邻决策满足：$g(i)>T(j_1,j_2)>T(j_2,j_3)>\ldots>T(j_{k-1},j_k)$，最优决策取队首元素。

三.斜率优化 II

- 状态转移方程能通过对比两个可选决策 $j_1 j_2$ 的优劣，使参数分离得到一个类似斜率的不等式。

如 $T(j_1, j_2) = \dfrac{y(j_2) - y(j_1)}{x(j_2) - x(j_1)} < g(i)$ 或 $> g(i)$

- 以上斜率不等式中 $x(i)$ 是单调的，**但g(i)不是单调的**。

例6.游戏

- **题目描述：**

　　*WYF*从小就爱乱顶，但是顶是会造成位移的。他之前水平有限，每次只能顶出k的位移，也就是从一个整点顶到另一个整点上。我们现在将之简化到数轴上，即从一个整点可以顶到与自己相隔在k之内的数轴上的整点上。现在*WYF*的头变多了，于是他能顶到更远的地方，他能顶到任意整点上。现在他在玩一个游戏，这个游戏里他只能向正方向顶，同时如果他从i顶到j,他将得到$a[j]*(j-i)$的分数，其中$a[j]$是j点上的分数，且要求$j>i$，他最后必须停在n上。

　　现给出$1\sim n$上的所有分数，原点没有分数。他现在在原点，没有分。*WYF*想知道他最多能得多少分。

- **输入格式：**

　　第一行一个整数n。第二行有n个整数,其中第i个数表示$a[i]$。

- **输出格式：**

　　一个整数，表示*WYF*最多能得到的分数。

- **样例输入：**

3

1 1 50

- **样例输出：**

150

- **数据范围：**

　　对于60%的数据，$n\le 1000$;

　　对于100%的数据，$n\le 100000, 0\le a[j]\le 50$。

- **出题人：北师大附属实验学校 庆语其**

例6.游戏

- **分析：**
- 动态规划：$f(i)$表示从原点顶若干次到i的最大得分。
- 考虑最后一次是从j顶到i的，得到以下状态转移方程：

$$f(i)=\max\{f(j)+a[i]*(i-j)\},0\le j\le i$$

- 直接做时间复杂度为$O(n^2)$,60分。
- 有一项是$-j*a[i]$，i和j没有分离，考虑两个决策$j_1<j_2$，什么情况下j_2比j_1优？

$$f(j_2)+a[i]*i-a[i]*j_2>f(j_1)+a[i]*i-a[i]*j_1$$

$$T(j_1,j_2)=\frac{f(j_2)-f(j_1)}{j_2-j_1}>a[i]$$

- 不等式左边也是斜率的形式，决策x的坐标是$(x,f(x))$,横坐标x是递增的，但$a[i]$不是单调的！
- 类似于斜率优化　中的结论1就不存在了，但相邻决策之间的斜率的单调性还是有的！
- 采用前面两题类似的方法可以得出，相邻决策之间的斜率是单调减的。即：

队列中从小到大排列的可选决策$j_1,j_2,\dots,j_k$要满足：$T(j_1,j_2)>T(j_2,j_3)>\dots>T(j_{k-1},j_k)$

例6.游戏

- **求最优决策方法一：二 分法**
- 计算$f(i)$时队列中可选决策有 $j_1, j_2, \ldots, j_k$,假设最优决策是 j_x。则有：

① j_x 比 j_{x-1} 优 $\Leftrightarrow T(j_x, j_{x-1}) > a[i]$

$\because T(j_1, j_2) > T(j_2, j_3) > \ldots > T(j_x, j_{x-1}) > a[i]$

$\therefore$ j_x 比前面的 $j_1, j_2, \ldots, j_{x-1}$ 都要优！

② j_x 比 j_{x+1} 优 $\Leftrightarrow T(j_x, j_{x+1}) < a[i]$

$\because a[i] > T(j_x, j_{x+1}) > T(j_{x+1}, j_{x+2}) > \ldots > T(j_{k-1}, j_k)$

$\therefore$ j_x 比后面的 $j_{x+1}, j_{x+2}, \ldots, j_k$ 都要优！

- 因此只要 j_x 满足$T(j_x, j_{x-1}) > a[i]$且$T(j_x, j_{x+1}) < a[i]$, j_x 就是最优决策
- 因为相邻决策之间的斜 率是单调减的，用二分 法就可以找出 j_x
- 时间复杂度为$O(n \ln n)$

例6.游戏

- **求最优决策方法二：三分法**
- 把队列中的可选决策j作为横坐标，把决策j对应的得分$f(i)_j$作为纵坐标，在二维平面中画出点$(j, f(i)_j)$
- 所有决策点形成一个单峰的模型，单峰求极值可以使用三分法：

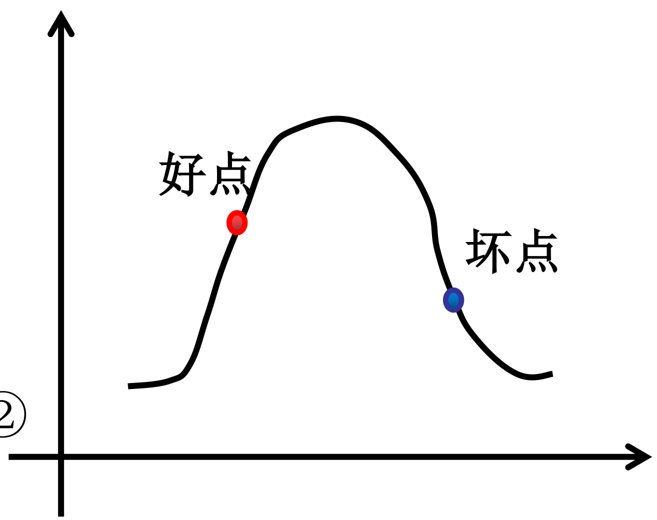
①一开始$l=1, r=k$

②取$x=\left\lfloor \frac{r-l+1}{3} \right\rfloor + l - 1$，$y=\left\lfloor \frac{2*(r-l+1)}{3} \right\rfloor + l - 1$

③计算$f(i)_{j_x}, f(i)_{j_y}$

④如$f(i)_{j_x} > f(i)_{j_y}$则$r=y-1$否则$l=x+1$

⑤如果$r-l+1 \le 2$则直接比较找出答案，否则回到②



- 每次排除$\frac{1}{3}$的区间，时间复杂度也是$O(n \ln n)$

例6.游戏

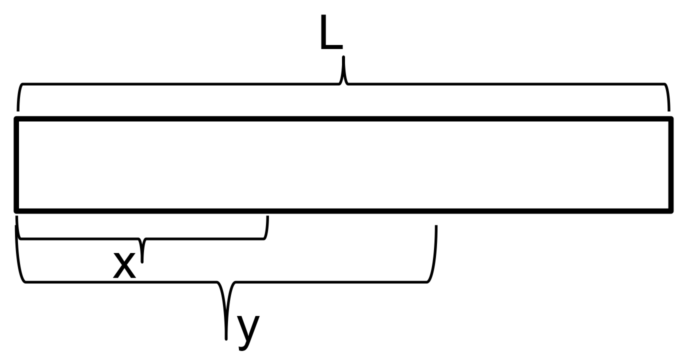
- **求最优决策方法三：黄金分割三分法**
- 另一种三分法，总区间长度为l,选择靠左决策点x,靠右决策点y,x和y与l的比例固定，并满足以下两个要求：

①如果x是坏点，删除x左边的无效区间，y在新的有效区间中是靠左的决策点；

②如果y是坏点，删除y右边的无效区间，x在新的有效区间中是靠右的决策点。

即：

$$\begin{cases} \dfrac{x}{l} = \dfrac{y-x}{l-x} \\ \\ \dfrac{y}{l} = \dfrac{x}{y} \end{cases} \Rightarrow \begin{cases} x = \dfrac{3-\sqrt{5}}{2} * l \\ \\ y = \dfrac{\sqrt{5}-1}{2} * l \end{cases}$$



- 每次删除$\dfrac{3-\sqrt{5}}{2} * l$的区间，每次三分只需计算一次决策点对应的值
- 时间复杂度也是$O(\ln n)$的

四.斜率优化III

- 状态转移方程能通过对比两个可选决策 $j_1 j_2$ 的优劣，使参数分离得到一个类似斜率的不等式。

如 $T(j_1,j_2)=\dfrac{y(j_2)-y(j_1)}{x(j_2)-x(j_1)}<g(i)$ 或 $>g(i)$

- 但以上斜率不等式中**x(i)和g(i)都不是单调的**。
- $x(i)$ 不单调时，插入新的决策点不一定插在最右边或最左边

例7.[NOI2007]货币兑换

- **题目描述：**

小Y最近在一家金券交易所工作。该金券交易所只发行交易两种金券：A纪念券(以下简称A券)和B纪念券(以前简称B券)。每个持有金券的顾客都有一个自己的账户。金券的数目可以是一个实数。

每天随着市场的起伏波动，两种金券都有自己当时的价值，即每一单位金券当天可以兑换的人民币数目。我们记录第k天中A券和B券的价值分别为A_k和B_k(元/单位金券)。

为了方便顾客，金券交易所提供了一种非常方便的交易方式：比例交易法。比例交易法分为两个方面：

a)卖出金券：顾客提供一个[0,100]内的实数OP作为卖出比例，其意义为：将OP%的A券和OP%的B券以当时的价值兑换为人民币；

b)买入金券：顾客支付IP元人民币，交易所将会兑换给用户总价值为IP的金券，并且，满足提供给顾客的A券和B券的比例在第K天恰好为$Rate_k$；

注意：同一天内可以进行多次操作。

小Y时一个很有经济头脑的员工，通过较长时间的运作和行情测算，他已经知道了未来N天内的A券和B券的价值以及$Rate$。他还希望能够计算出来，如果开始时拥有S元钱，那么N天后最多能够获得多少元钱。

- **输入格式：**

第一行两个正整数N、S,分别表示小Y能预知的天数以及初始时拥有的钱数。 接下来N行，第K行三个实数A_k、B_k、$Rate_k$。

- **输出格式：**

只有一个实数$Maxprofit$，表示第N天的操作结束时能够获得的最大的金钱数目。答案保留3位小数。

- **数据规模和约定：**

40%：$N \le 10$；60%：$N \le 1000$；100%：$N \le 100\,000, 0 < A_k \le 10, 0 < B_k \le 10, 0 < Rate_k \le 100, Maxprofit \le 10^9$

- **提示：**

必然存在一种最优的买卖方案满足：每次买进操作使用完所有钱；每次卖出操作卖出所有金券。

例7.[NOI2007]货币兑换

- **问题简述：**

  金券交易所可以对A、B两种金券进行交易。

  接下来的N天内，第i天A、B分别具有单位价值A_i、B_i，每一天用户进行若干次如下操作：

  ①卖出所有的金券；

  ②用所有的钱买入等价值的金券，买入的A、B两种金券的比例为$Rate_i$。

  初始时用户拥有S元钱，问N天后用户最多拥有多少钱？

- **提示的证明：**

- 在进行买进操作的时候，假设有S元，假定一元钱买进的A券和B券到最后会获利为p，一元钱不买金券最后获利为q。

买进时使用钱的比例为x，则最后总利润$=S*x*p+S*(1-x)*q=S*(q+(p-q)*x)$

①当$p \ge q$时，取$x=1$能使获利最大化 ②当$p<q$时，取$x=0$能使获利最大化

也就是说，买进操作要么不买要么全部买进；

- 类似的，可以证明卖出操作也是完全的，即要么不卖要么全卖。

例7.[NOI2007]货币兑换

- **方法一：搜索**
- 根据上面的提示，总结出每一天可能进行的活动有以下4种：

①不进行活动

②全部买进

③全部卖出后再全部买进

④全部卖出

- 可以用搜索来解决，时间复杂度为$O(4^n)$,40分。

例7.[NOI2007]货币兑换

- **方法二：动态规划**
- 考虑最后一天可以进行的活动：

①不进行活动：最大获利等于第$N-1$天的最大获利；

②全部买进：这种操作没有意义；

③全部卖出后再全部买进：这种操作没有意义；

④全部卖出：假设最后一次买入操作发生在第j天，即第$j+1$天到第$N-1$没有进行任何操作。这样实际上就是要求把第j天的最大获利全部买入再在第N天卖出，这就涉及到“第j天的最大获利”这个子问题。

- 动态规划：状态$f(i)$表示前i天的最大获利, $f(1)=S$
- 根据上面的分析，得以下状态转移方程：

$$f(i)=\begin{cases} S, i=1\text{时} \\ \max\begin{cases} f(i-1) \\ f(j)*\dfrac{Rate_j*A_i+B_i}{Rate_j*A_j+B_j} & 1\le j<i \end{cases} \end{cases}$$

- 直接做时间复杂度为$O(n^2)$,60分。

例7.[NOI2007]货币兑换

- **方法三：斜率优化**
- 以上方程主要慢在$f(j)*\dfrac{Rate_i*A_i+B_i}{Rate_j*A_j+B_j}$这里,因为要从1到$i-1$枚举$j$
- 设$g(j)=\dfrac{f(j)}{Rate_j*A_j+B_j}$,则上式$=g(j)*Rate_j*A_i+g(j)*B_i$
- 分析两个决策 j_1、j_2,什么情况下 j_2比 j_1更优?

$$g(j_2)*Rate_{j_2}*A_i+g(j_2)*B_i>g(j_1)*Rate_{j_1}*A_i+g(j_1)*B_i$$

$$\left(g(j_2)*Rate_{j_2}-g(j_1)*Rate_{j_1}\right)*A_i>B_i*\left(g(j_1)-g(j_2)\right)$$

$g(j)$没有单调性:

①$g(j_2)<g(j_1)$时，得：$T(j_1,j_2)=\dfrac{g(j_2)*Rate_{j_2}-g(j_1)*Rate_{j_1}}{\left(-g(j_2)-\left(-g(j_1)\right)\right)}>\dfrac{B_i}{A_i}$

②$g(j_2)=g(j_1)$时，得：$g(j_2)*Rate_{j_2}>g(j_1)*Rate_{j_1}$

以上不等式左边是斜率的形式，决策点j的横坐标是$-g(j)$，纵坐标是$g(j)*Rate_j$

把所有决策点按照横坐标从小到大排序得决策序列$j_1,j_2,\ldots,j_k$，利用前面所学易知：

$$T(j_1,j_2)>T(j_2,j_3)>\ldots>T(j_{k-1},j_k)$$

例7.[NOI2007]货币兑换

- **方法三：斜率优化**
- g不单调时，插入新决策点时不可以直接插在队尾，有可能会插在队列中间，

然后再维护相邻决策之间的斜率单调递减！

- 插入一个新决策x的操作如下：

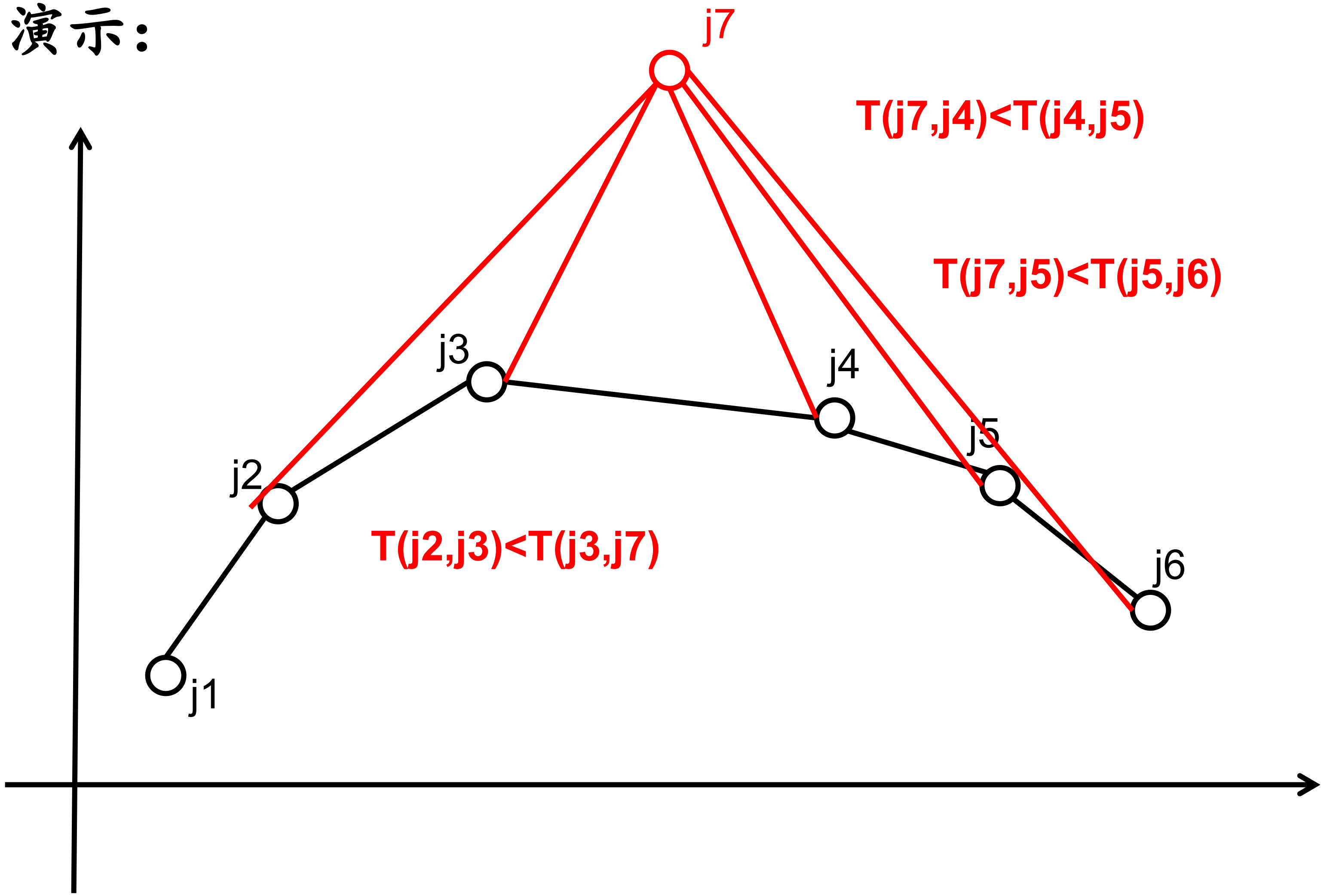
①找到x左边的决策x_1,右边的决策x_2，如果$T(x_1,x)<T(x,x_2)$出现了斜率上升，x不用插入，否则进行②

②插入x,并分别维护x的左边和右边，两边都要保证斜率是递减的，其中左边的维护可以每次找x的前一个决策点x_1和x_1的前一个决策点x_2，如果$T(x_2,x_1)<T(x_1,x)$则删除x_1，重复执行直到左边的决策点少于2个或者满足$T(x_2,x_1)>T(x_1,x)$。右边的维护类似。

- 可以用平衡树来维护，$Treap$或$Splay$都可以。
- 因为$\frac{B_i}{A_i}$不单调，寻找最优决策可以采用例6中的二分法，这里不再介绍。
- 时间复杂度为$O(n\ln n)$,100分。

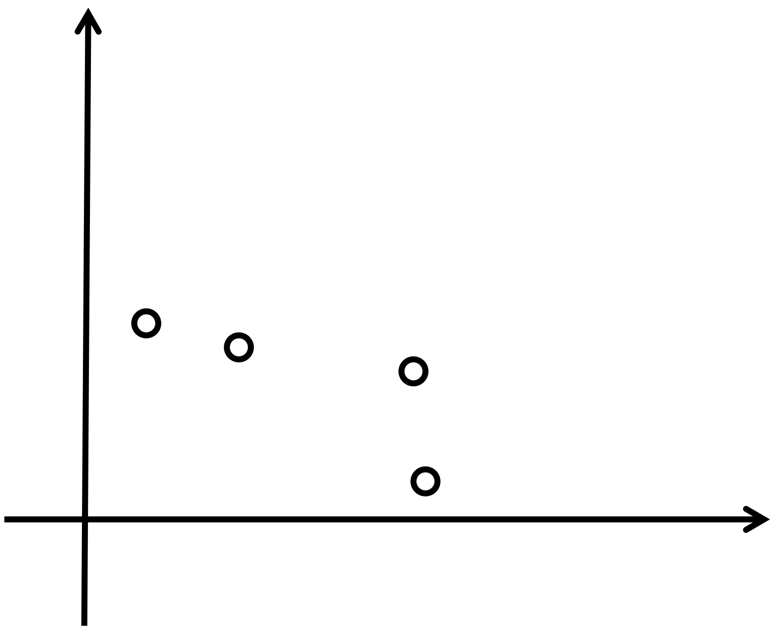
例7.[NOI2007]货币兑换

- 动画演示：



例7.[NOI2007]货币兑换

- 方法四：线性规划
- $f(i)=\max\left\{f(j)*\frac{Rate_j*A_i+B_i}{Rate_j*A_j+B_j}\right\}$
- 令$x_j=\frac{Rate_j*f(j)}{Rate_j*A_j+B_j}, y_j=\frac{f(j)}{Rate_j*A_j+B_j}$，决策$j$对应的获利$=A_i*x_j+B_i*y_j$



例7.[NOI2007]货币兑换

- **几何意义：**

①决策j在二维平面上对应着点$\left(x_j, y_j\right)$,过这个点作一条斜率为$-\frac{A_i}{B_i}$的直线，

直线方程为：$\frac{y-y_j}{x-x_j}=-\frac{A_i}{B_i} \Rightarrow y=-\frac{A_i}{B_i}*x+\frac{A_i*x_j+B_i*y_j}{B_i}$

直线在y轴的截距为$\frac{A_i*x_j+B_i*y_j}{B_i}$，恰好为获利的$\frac{1}{B_i}$

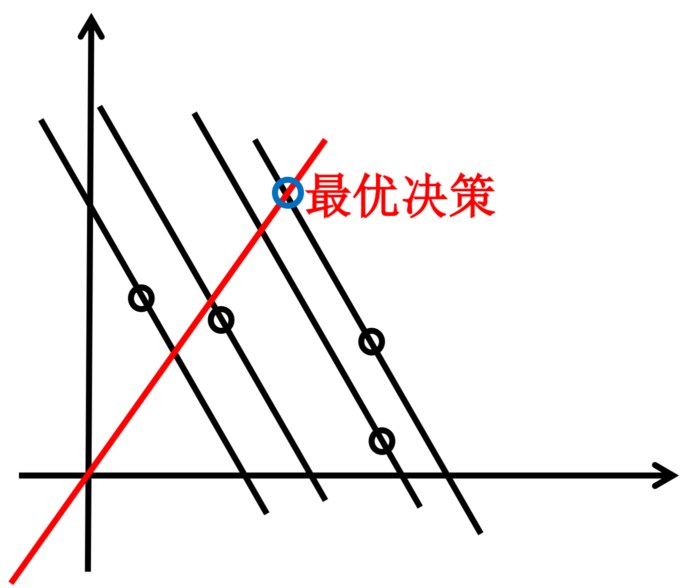
②直线上每一个点在第i天的价值都是一样的；

例7.[NOI2007]货币兑换

- **几何意义：**

③过原点作一条斜率为$\frac{1}{Rate_i}$的直线，两直线的交点就是用“使用决策j计算$f(i)$时的获利”在第i天购买金券对应到二维平面上的点；

④过每个决策点作斜率为$-\frac{A_i}{B_i}$的直线，再过原点作一条斜率为$\frac{1}{Rate_i}$的直线,所有交点中离原点最远的就是最优决策。

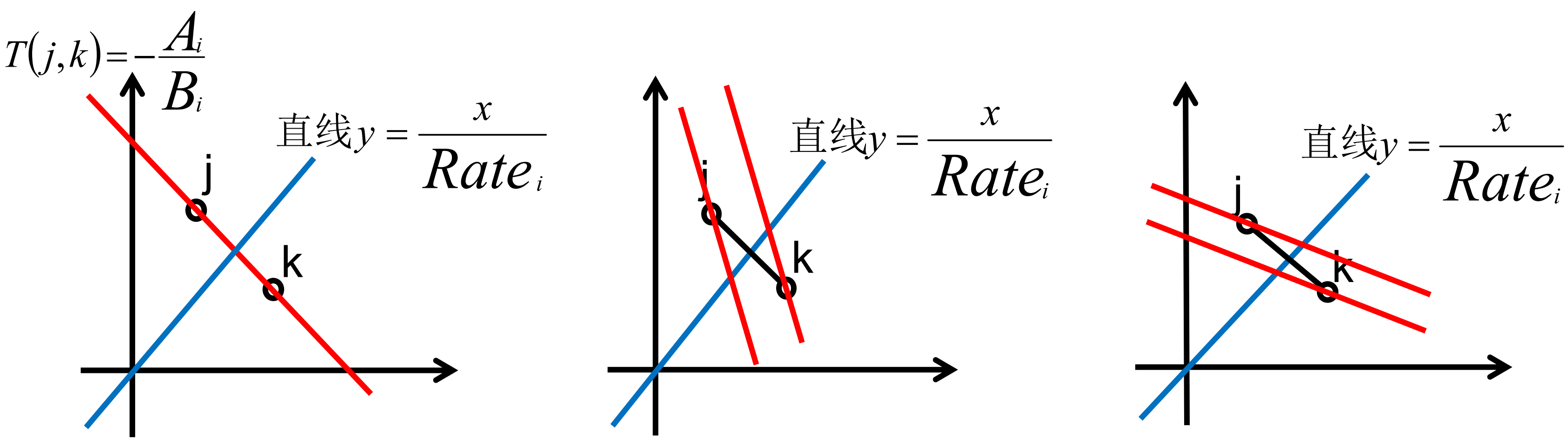


例7.[NOI2007]货币兑换

- **优化一：去除冗余**

把所有决策按照x升序，x相同时按照y降序排序，维护序列使得决策点的y随着x增加而减少

- **优化二：维护相邻决策斜率单调减**
- 假设$x_j < x_k$,点j和点k连线的斜率为$T(j,k)$,分析决策j和决策k的优劣情况：



(1) $T(j,k) = -\frac{A_i}{B_i}$,决策j与k一样优 (2) $T(j,k) > -\frac{A_i}{B_i}$,决策k更优 (3) $T(j,k) < -\frac{A_i}{B_i}$,决策j更优

例7.[NOI2007]货币兑换

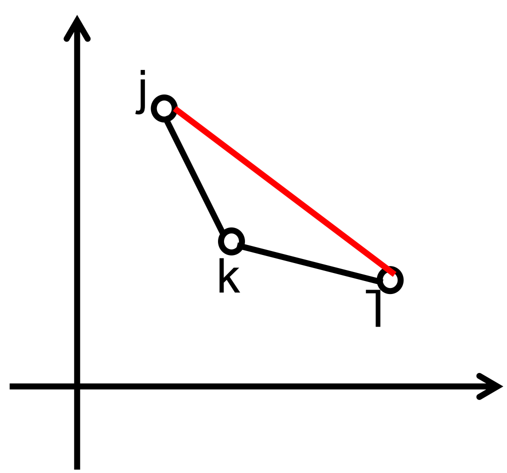
- **优化二：维护相邻决策 斜率单调减**
- **证明：** 假设三个相邻的决策j,k,l满足$T(j,k)<T(k,l)$，即如下图所示。

考虑决策k在这种情况下现在包括 将来有没有可能成为最 优决策

根据上面的分析：k比j优 $\Rightarrow T(j,k)>-\dfrac{A_i}{B_i}$　k比l优 $\Rightarrow T(k,l)<-\dfrac{A_i}{B_i}$

$T(k,l)>-\dfrac{A_i}{B_i}>T(j,k)$，与前面的假设$T(j,k)>T(k,l)$产生矛盾。

决策k永远不可能成为最优决 策，可以删除。得证！



例7.[NOI2007]货币兑换

- **方法四：线性规划**

上面的结论与方法三类似，实现与方法三一样，维护序列用平衡树来实现，寻找最优决策用二分来实现。时间复杂度为$O(n \ln n)$。

五.斜率优化IV

- 状态转移方程能通过对比两个可选决策 $j_1 j_2$ 的优劣，使参数分离得到一个类似斜率的不等式。

如$T(j_1,j_2)=\dfrac{y(j_2)-y(j_1)}{x(j_2)-x(j_1)}<g(i)$或$>g(i)$

- 以上斜率不等式中，**x(i)不是单调的，g(i)单调。**
- 有了前面的基础，相信大家能自行分析出来，这里不再讨论。

六、总结

- **补充两点：**

①斜率不等式中x单调时，插入新决策点直接插在队尾，否则可能插在中间，需要用数据结构来维护；

②斜率不等式中g单调时，队列中的决策会增加一个限制条件，最优决策在队首或队尾，否则要用二分来寻找最优决策。