单调队列优化DP

西南大学附属中学 High School Affiliated to Southwest University

- 基本模型：
  - $f[i] = \min/\max_{L(i) \le j \le R(i)} \{f[j] + val(i) + val(j)\}$
- L(i)、R(i)是关于变量i 的一次函数，限制j 的取值范围，且保证上下界变化具有单调性。
- val(i)是仅与变量i 有关，val(j) 是仅与变量j 有关。
- 使用单调队列维护f[j]+val(j)。
- 例如：$f[i][j] = P_i * j + \max_{j-L_i \le k \le S_i-1} \{f[i-1][k] - P_j * k\}$ (POJ 1821 Fence)
  - 将i看作定值，j是状态变量，k是决策变量。
  - 维护决策点k单调递增，数值$f[i-1][k] - P_j * k$ 单调递减。

西大附中信息学竞赛 High School Affiliated to Southwest University

单调队列优化DP

- 基本模型：
  - $f[i] = \underset{L(i)\le j\le R(i)}{min/max}\{f[j] + val(i) + val(j)\}$
- $f[i] = \underset{L(i)\le j\le R(i)}{min/max}\{f[j] + val(i) + val(j) + val(i, j)\}$
- val(i,j)包含了变量i、j 的乘积项，即**同时与变量 i 和 j 有关**。
- 单调队列优化不再适用

西南大学附属中学信息学竞赛
High School Affiliated to Southwest University

FZOI

信息学

斜率优化DP

例1：任务安排1

问题描述：

N个任务被分成若干批，每批包含相邻的若干任务。从时刻0开始，这些任务被分批加工。执行第i 个任务所需时间为Ti。每批任务开始前，机器需要S 的启动时间，故执行一批任务所需时间是启动时间S 加上每个任务所需时间之和。

一个任务执行后，将在机器中等待，直到该批任务全部执行完毕，也就是说，整批任务在同一时刻完成。每个任务的费用是它完成的时刻乘以一个费用系数Ci。

请你规划一个分组方案，使得总费用最少。

西 大 附 中 信 息 学 竞 赛
High School Affiliated to Southwest University

数据范围：1≤N≤5000，1≤S≤50,1≤Ti,Ci≤100

例1：任务安排1

问题分析：

- 状态：
  - **f[i][j]**，前i 个任务分成j 批执行的最小费用。
- 决策：考虑第j 批执行包含的任务
- 状态转移方程：
  - $sumT[i] = \sum_{j=1}^{i} T[j]$，$sumC[i] = \sum_{j=1}^{i} C[j]$
  - $f[i][j] = \min_{0 \le k < i} \{f[k][j-1] + (S * j + sumT[i]) * (sumC[i] - sumC[k])\}$
  - 考虑第j 批执行的是k+1~ i 个任务
- 枚举第j-1 批和第j 批的分界点k 为DP的决策。
- 时间复杂度为$O(N^3)$。

西大附中信息学竞赛 High School Affiliated to Southwest University

例1：任务安排1

问题分析：

- 状态能否优化?
- 题目并没有规定分成多少批次。
- 之所以需要批次，是因为想知道有多少次启动时间S，从而计算出每批任务完成的时间。
- 实际上，可以将每批任务花费的启动时间S，对之后任务的影响**提前计算**。

西 大 附 中 信 息 学 竞 赛
High School Affiliated to Southwest University

例1：任务安排1

问题分析：

- 状态：
  - f[i]，表示前i 个任务划分成若干批执行的最小费用。
- 考虑当前批次执行的任务，状态转移方程：
  - $f[i] = \min_{0 \le j < i} \{f[j] + sumT[i] * (sumC[i] - sumC[j]) + S * (sumC[N] - sumC[j])\}$
  - 当前批次执行的任务为第j+1~i 个任务
  - 机器的启动时间会对第j 个任务以后的所有任务产生影响，提前将影响累加到最小费用中
- 这就是“**费用提前计算**”的经典思想
- 时间复杂度为$O(N^2)$。

西大附中信息学竞赛 High School Affiliated to Southwest University

例2：任务安排2

问题描述：

N个任务被分成若干批，每批包含相邻的若干任务。从时刻0开始，这些任务被分批加工。执行第i 个任务所需时间为Ti。每批任务开始前，机器需要S 的启动时间，故执行一批任务所需时间是启动时间S 加上每个任务所需时间之和。

一个任务执行后，将在机器中等待，直到该批任务全部执行完毕，也就是说，整批任务在同一时刻完成。每个任务的费用是它完成的时刻乘以一个费用系数Ci。

请你规划一个分组方案，使得总费用最少。

西 大 附 中 信 息 学 竞 赛
High School Affiliated to Southwest University

数据范围：**$1 \le N \le 3*10^5$，$1 \le S, Ti, Ci \le 512$**

例2：任务安排2

西南大学附属中学
High School Affiliated to Southwest University

问题分析：

- $f[i] = \min_{0 \le j < i} \{f[j] + sumT[i] * (sumC[i] - sumC[j]) + S * (sumC[N] - sumC[j])\}$

同时和变量i 和变量j 有关，单调队列不适用

- 将min函数去掉，把关于j 的值f[j] 和sumC[j] 看作变量，其余部分看作常数。
- 移项得到：
- $f[j] = (sumT[i] + S) * sumC[j] + f[i] - sumT[i] * sumC[i] - S * sumC[N]$

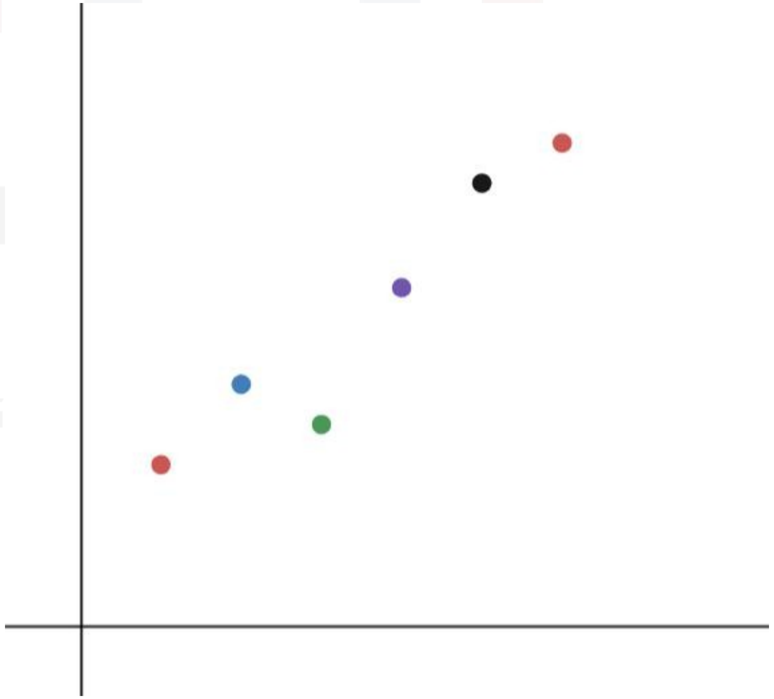
西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛
High School Affiliated to Southwest University

例2：任务安排2

问题分析：

- $f[j] = (sumT[i] + S) * sumC[j] + f[i] - sumT[i] * sumC[i] - S * sumC[N]$

- 将sumC[j] 看作是横坐标，f[j]看作是纵坐标，就是一个形如y=ax+b的一条直线。
- 斜率：$sumT[i] + S$
- 截距：$f[i] - sumT[i] * sumC[i] - S * sumC[N]$
- 对于每一个决策j，都对应直角坐标系中的一个点(sumC[j],f[j])。

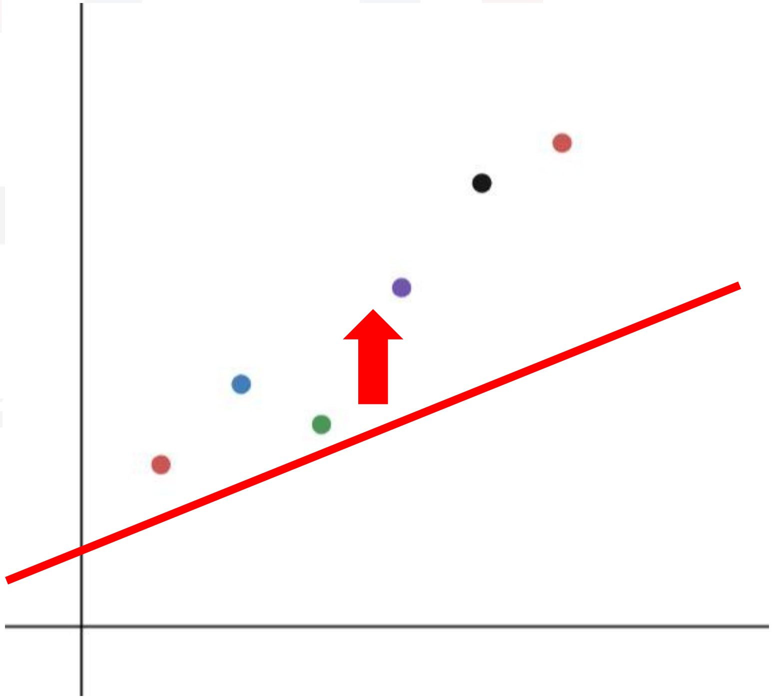


例2：任务安排2

西南大学附属中学
High School Affiliated to Southwest University

问题分析：

- $f[j] = (sumT[i] + S) * sumC[j] + f[i] - sumT[i] * sumC[i] - S * sumC[N]$
- 当变量i 不变时，直线的斜率是固定值$sumT[i] + S$。
- sumT[i],sumC[i],S,sumC[N]为固定值。当截距取得最小值时，f[i]也取到最小值。
- 求解最小截距的方法：
  - 将斜率为$sumT[i] + S$ 的直线经过每一个决策点，截距最小的为最优决策。
  - 将直线从下往上移动，遇到的第一个决策点就是最优决策。

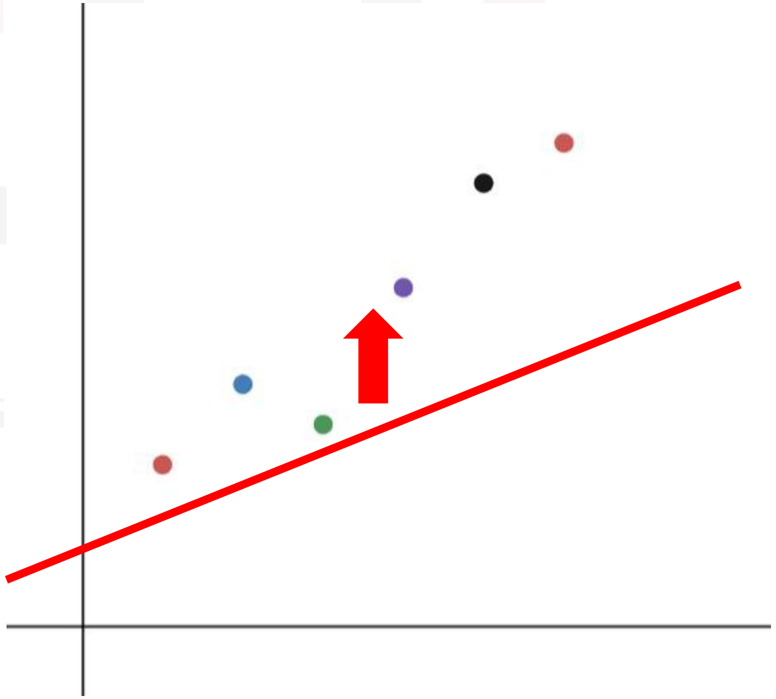


例2：任务安排2

西南大学附属中学
High School Affiliated to Southwest University

问题分析：

- 所有的决策点(sum[j],f[j])都是有用的吗？
- 利用"**及时排除无用决策**"思想，将无用的决策点删除。

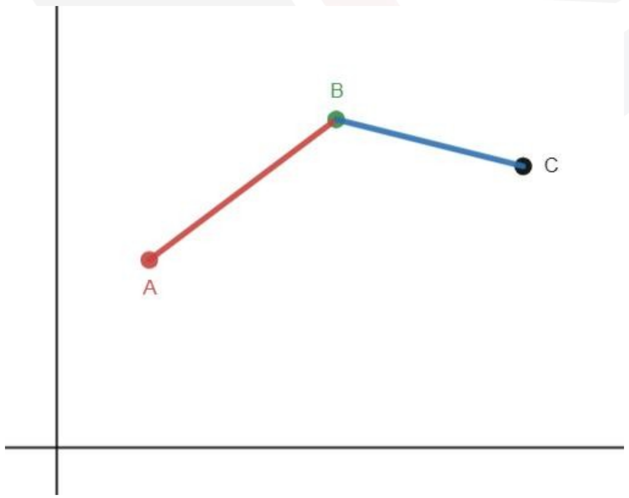


例2：任务安排2

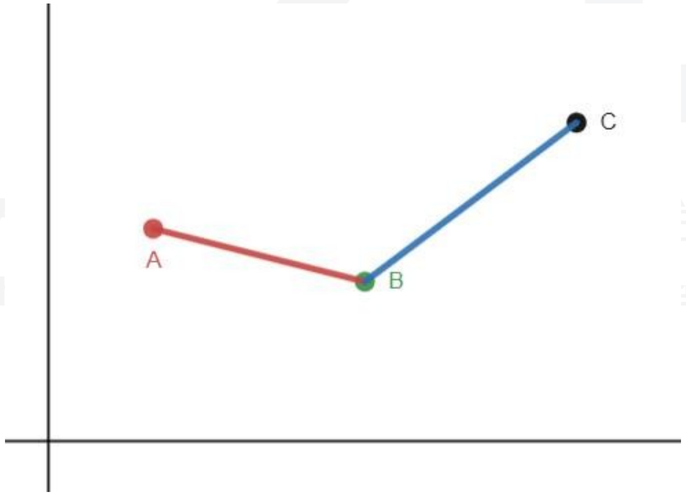
问题分析：

•假设存在三个决策j_1，j_2，j_3，对应的决策点为(sumC[j_1],f[j_1])，(sumC[j_2],f[j_2])，(sumC[j_3],f[j_3])，设三点分别为A，B，C。

- 不妨设 $j_1<j_2<j_3$，因为T，C均为整数，有sumC[j_1]<sumC[j_2]<sumC[j_3]
- 有两种情况



上凸

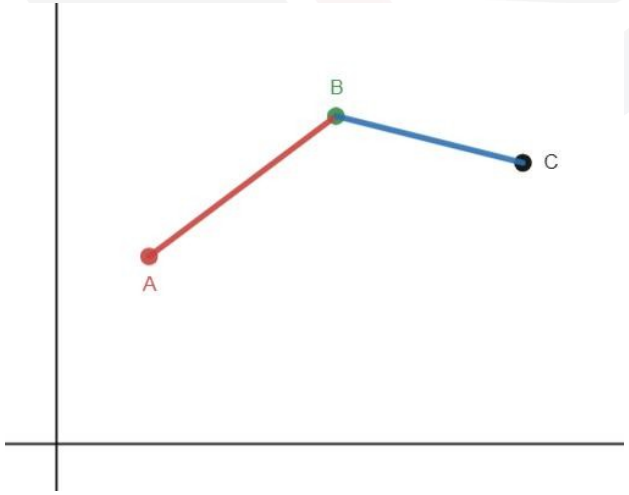


下凸

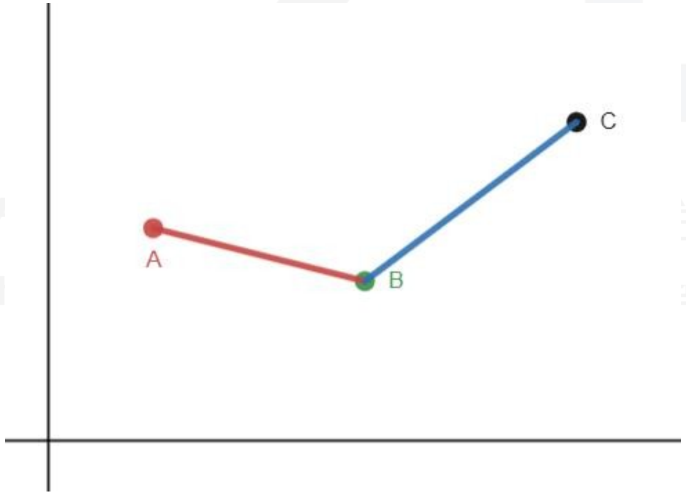
例2：任务安排2

问题分析：

- 对于上凸，无论斜率是多少，j_2都不可能是最优决策，可以排除。
- 对于下凸，j_2可能是最优决绝，当且仅当：
- $\frac{f[j_2]-f[j_1]}{sumC[j_2]-sumC[j_1]} < \frac{f[j_3]-f[j_2]}{sumC[j_3]-sumC[j_2]}$



上凸

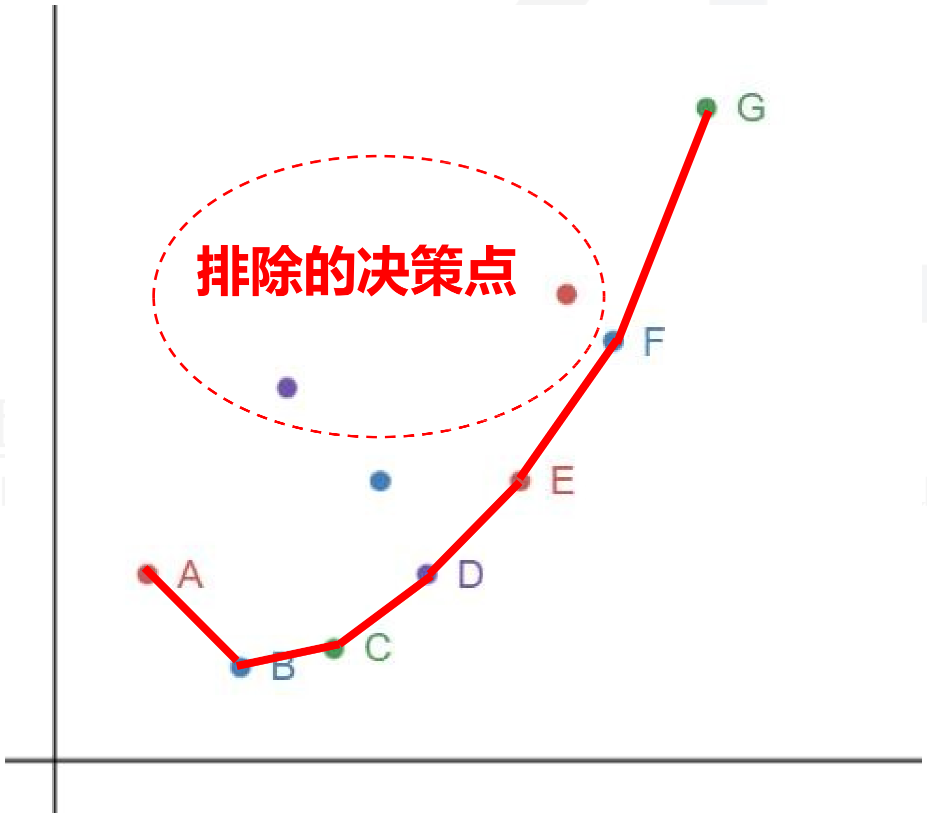


下凸

例2：任务安排2

问题分析：

- 将不可能的最优决策排除后，将剩下的点集相邻两点连线
- 形成的**线段的斜率从左到右是单调递增**
- 需要维下凸壳
- 使用**单调队列**维护



例2：任务安排2

问题分析：

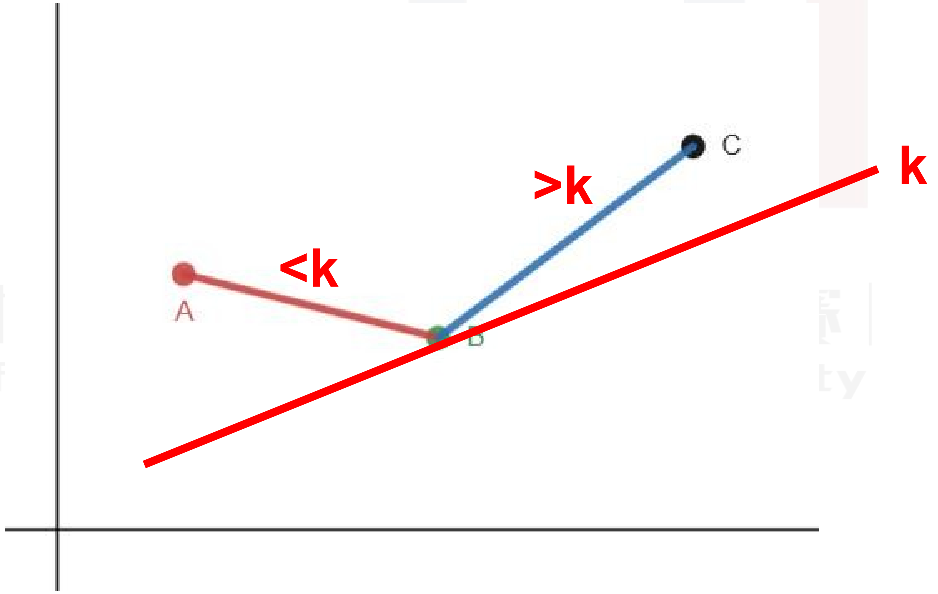
- 哪一个点是最优决策呢？

•对于斜率为k的直线，若某个点左侧线段的斜率小于k，右侧线段的斜率大于k，那么该点就是最优决策点。

- 如何在斜率单调递增的队列中找到最优决策点？

- **二分**

- 时间复杂度为O(NlogN)。



例2：任务安排2

西南大学附属中学
High School Affiliated to Southwest University

问题分析：

- 还能不能再优化?
- f[j] = $(sumT$[i] + $S)*sumC$[j] + f[i] − $sumT$[i] $*\ sumC$[i] − $S*sumC[N]$
- 因为sumC是单调递增的，新的决策点的横坐标一定大于之前所有决策点的横坐标
- 斜率S+sumT[i]也是单调递增的
- 只**维护相邻两点线段斜率大于**S+sumT[i]**的决策点**，那么最优决策点就是队头。

例2：任务安排2

问题分析：

- 对于每个状态i：

1. 检查队头的两个决策q[l] 和q[l+1]，若斜率 $\frac{f[q[l+1]]-f[q[l]]}{sumC[q[l+1]]-sumC[q[l]]}$ ≤S+sumT[i]，则将q[l] 出 队，继续检查队头

2. 直接取出队头q[l]为最优决策，计算f[i]。

3. 将新决策i 加入队尾，插入前，若三个决策点j_1=q[r-1]，j_2=q[r]，j_3=i 不满足下凸，则 j_2=q[r]是无用决策，将q[r]出队，继续检查队尾。

- 时间复杂度为O(N)。

西大附中信息学竞赛
High School Affiliated to Southwest University

斜率优化

- 维护队列中相邻两个元素的某种“比值”的“单调性”
- 因为该比值对应坐标系中的斜率
- 所以称为斜率优化
- 英文称为convex hull trick(直译：凸壳优化策略)

西|大|附|中|信|息|学|竞|赛
High School Affiliated to Southwest University

例3：任务安排3

问题描述：

N个任务被分成若干批，每批包含相邻的若干任务。从时刻0开始，这些任务被分批加工。执行第i 个任务所需时间为Ti。每批任务开始前，机器需要S 的启动时间，故执行一批任务所需时间是启动时间S 加上每个任务所需时间之和。

一个任务执行后，将在机器中等待，直到该批任务全部执行完毕，也就是说，整批任务在同一时刻完成。每个任务的费用是它完成的时刻乘以一个费用系数Ci。

请你规划一个分组方案，使得总费用最少。

西大附中信息学竞赛
High School Affiliated to Southwest University

数据范围：**$1 \le N \le 3*10^5$，$1 \le S,Ci \le 512$，$-512 \le Ti \le 512$。**

例3：任务安排3

问题分析：

- Ti可以为负数，
- $f[j] = (sumT[i] + S) * sumC[j] + f[i] - sumT[i] * sumC[i] - S * sumC[N]$
- 那么斜率S+sumT[i]不再单调
- 不能仅仅只维护相邻两点线段斜率大于S+sumT[i]的决策点，需要维护所有下凸壳的决策点。
- 如何找到最优决策？
- 单调队列中二分
- 时间复杂度为O(NlogN)。

西 | 大 | 附 | 中 | 信 | 息 | 学 | 竞 | 赛
High School Affiliated to Southwest University

例4：任务安排4

西南大学附属中学 High School Affiliated to Southwest University

问题分析：

- Ti为正数，Ci可以为负数?
- $f[j] = (sumT[i] + S) * sumC[j] + f[i] - sumT[i] * sumC[i] - S * sumC[N]$

方法一：

- 新增加的决策点的横坐标sumC[i]不再单调递增，会插入到凸壳中间的位置，队列不能实现插入操作

•平衡树支持动态插入，利用平衡树维护斜率单调性

方法二：

- 可以倒序DP，设计一个状态转移方程，让sumT为横坐标，sumC为斜率的一项，转为为例3 的情况，使用单调队列维护凸壳，使用二分查找求出最优策略。

西大附中信息学竞赛
High School Affiliated to Southwest University

例5：任务安排5

问题分析：

- Ti，Ci均可以为负数?
- $f[j] = (sumT[i] + S) * sumC[j] + f[i] - sumT[i] * sumC[i] - S * sumC[N]$
- 斜率、横坐标都不是单调的
- 使用平衡树，支持动态插入，查询前驱、后继。
- CDQ(https://www.cnblogs.com/Parsnip/p/10832015.html)

西大附中信息学竞赛 High School Affiliated to Southwest University

总结

- $f[i-1][k] + sumA[k] = A_j * k + f[i][j] - A_j * j - sumA[j]$
- $f[j] = (sumT[i] + S) * sumC[j] + f[i] - sumT[i] * sumC[i] - S * sumC[N]$

- 斜率优化DP，将状态转移方程转换为y=kx+b的形式
- b中仅包含与状态变量有关的项，所求状态包含在内
- kx包含决策变量与状态变量的乘积项
- y中仅包含与决策变量有关的项

西 大 附 中 信 息 学 竞 赛
High School Affiliated to Southwest University