提高算法班

DP优化2

Mas

#2871、树上背包

题目描述

有 N 个物品,编号分别为 $1 \sim N$,物品 i 的重量为 w_i ,价值为 v_i

给出每个物品依赖于哪个物品,用 fa_i 表示:如果要选取物品,就必须先选取物品 fa_i

另外用 $fa_i = 0$ 表示该物品不依赖于任何物品

保证每个物品最多只依赖一个物品,保证依赖关系合理不会出现环

背包最多能装载的重量为 W ,请问背包中最多能装入多大价值的物品

输入格式

第一行输入两个正整数 N, W

第二行输入 N 个正整数 fa_i

第三行输入 N 个正整数 w_i

第四行输入 N 个正整数 v_i

输出格式

一个整数,表示答案

数据范围

对于全部的数据 $1 \le N \le 5 \times 10^4, 1 \le W \le 6 \times 10^4, 1 \le N \times W \le 6.5 \times 10^7$

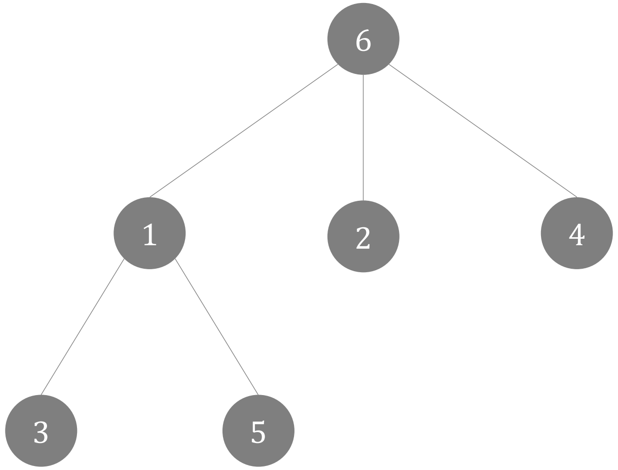
朴素做法时间复杂度 $O(NW^2)$

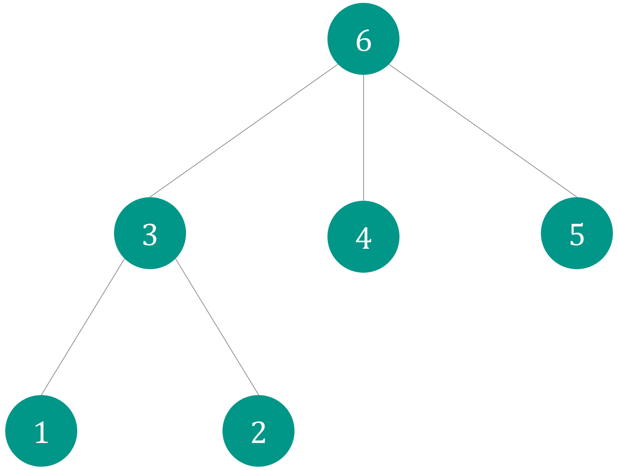
不妨求出树的 DFS 后序遍历序列 dfn

按后序遍历顺序考虑加入各点

不难看出各点被考虑加入时其必然为某棵子树的根

且其子树已被考虑





#2871、树上背包

记 cnt_i 表示以 dfn_i 为根的子树节点数量

设 $\mathrm{dp}[i][j]$ 表示按后序遍历顺序考虑前 i 个物品且背包容量为 j 时的最大价值

若选取 i 对应物品则可以选取其子树即

$$\mathrm{dp}[i-1][j-\mathrm{w}_i]+\mathrm{v}_i$$

若不选取 i 对应物品则其子树都不可选即

$$\mathrm{dp}[i-\mathrm{cnt}_i][j]$$

状态转移方程为

$$\mathrm{dp}[i][j]=\begin{cases}\mathrm{dp}[i-\mathrm{cnt}_i][j]\,, & j<\mathrm{w}_i\\ \max(\mathrm{dp}[i-\mathrm{cnt}_i][j],\mathrm{dp}[i-1][j-\mathrm{w}_i]+\mathrm{v}_i), & j\ge \mathrm{w}_i\end{cases}$$

时/空间复杂度 $O(\mathrm{NW})$

该做法无法得到子树节点的具体选取信息，若多次询问 $\mathrm{subtree}_u$ 内容量为 W’ 时的答案，无法处理

优化上下界的树形背包

考虑如下问题：

有 N 个物品背包容量为 W ，每个物品重量为 1，价值 v_i

记 S 为 u 转移时已考虑过的子树大小，cnt_u 为以 u 为根的子树大小

考虑朴素做法中的 u 和 v 其中 $v \in \mathrm{son}_u$

- 若 $j > \mathrm{W}$

  此时 $\mathrm{dp}[u][j]$ 无意义

- 若 $j > \mathrm{S}$

  此时 $\mathrm{dp}[u][j]$ 无意义

- 若 $k > \mathrm{cnt}_v$

  此时 $\mathrm{dp}[v][k]$ 无意义

那么对于 $\mathrm{dp}[u][j+k] = \min(\mathrm{dp}[u][j] + \mathrm{dp}[v][k])$，优化转移的上下界

优化上下界的树形背包

对于 $\mathrm{dp}[u][j]$

j 的上界为 $\min(\mathrm{W},\mathrm{S})$

对于 $\mathrm{dp}[v][k]$

k 的上界为 $\min(\mathrm{cnt}_v,\mathrm{W}-j)$ 即保证 $j+k\le \mathrm{W}$

对于 S 需在处理完 v 的合并后再令 $\mathrm{S}\leftarrow \mathrm{S}+\mathrm{cnt}_v$

否则当依赖关系为链时时间复杂度将退化为 $O(\mathrm{NW}^2)$

该做法时间复杂度 $O(\mathrm{NW})$

对时间复杂度进行分析

对子树进行合并时，两个节点 u,v 仅会在 $\mathrm{LCA}_{u,v}$ 处进行一次合并（同一子树内时不再合并）

令 T_u 表示处理以 u 为根的子树的代价

t_u 表示处理节点 u 的代价

优化上下界的树形背包

那么

$$T_u = \left(\sum_{v \in \mathrm{son_u}} T_v\right) + t_u$$

其中

$$t_u = \min(W, \mathrm{cnt}_{v_1}) \times \min(W, \mathrm{cnt}_{v_1}) + \min(W, \mathrm{cnt}_{v_1} + \mathrm{cnt}_{v_2}) \times \min(W, \mathrm{cnt}_{v_2}) + \cdots$$

$$\le \min(W, \mathrm{cnt}_u) \times \left(\sum_{v \in \mathrm{son_u}} \min(W, \mathrm{cnt}_v)\right)$$

$$= \min(W, \mathrm{cnt}_u) \times \min(W, \mathrm{cnt}_u)$$

对于子树都为叶子的 u 有(不妨设 $\mathrm{cnt}_u \le W$)

$$T_u \le \mathrm{cnt}_u^2 + \sum 1$$

优化上下界的树形背包

对于 $\mathrm{cnt}_u \le W$ 时的节点有

$$T_u = \left(\sum_{v \in \mathrm{son_u}} T_v\right) + t_u \le \left(\sum_{v \in \mathrm{son_u}} \mathrm{cnt}_v^2\right) + \mathrm{cnt}_u^2$$

由于 **平方的和** 不超过 **和的平方**（不难通过数学归纳法证明）

$$\mathrm{cnt}_u = \left(\sum_{v \in \mathrm{son_u}} \mathrm{cnt}_v\right) \Rightarrow \left(\sum_{v \in \mathrm{son_u}} \mathrm{cnt}_v^2\right) \le \left(\sum_{v \in \mathrm{son_u}} \mathrm{cnt}_v\right)^2 = \mathrm{cnt}_u^2$$

$$\Rightarrow T_u \le 2\mathrm{cnt}_u^2$$

对于 $\mathrm{cnt}_u \ge W$ 时的节点有

$$T_u = \left(\sum_{\substack{v \in \mathrm{son_u} \\ \mathrm{cnt}_v \le W}} T_v\right) + \left(\sum_{\substack{v \in \mathrm{son_u} \\ \mathrm{cnt}_v > W}} T_v\right) + t_u$$

优化上下界的树形背包

$$\le \left(\sum_{\substack{v\in \mathrm{son_u} \\ \mathrm{cnt}_v \le \mathrm{W}}} \mathrm{cnt}_v^2\right) + \left(\mathrm{W} \times \sum_{\substack{v\in \mathrm{son_u} \\ \mathrm{cnt}_v > \mathrm{W}}} \mathrm{cnt}_v\right) + \mathrm{t}_u$$

$$\le \mathrm{W} \times \mathrm{cnt}_u + \mathrm{W} \times \mathrm{cnt}_u + \mathrm{W} \times \min(\mathrm{W}, \mathrm{cnt}_u)$$

那么整个问题求解代价为 $O(\mathrm{NW})$

当 NW 同阶时，从另一视角理解：

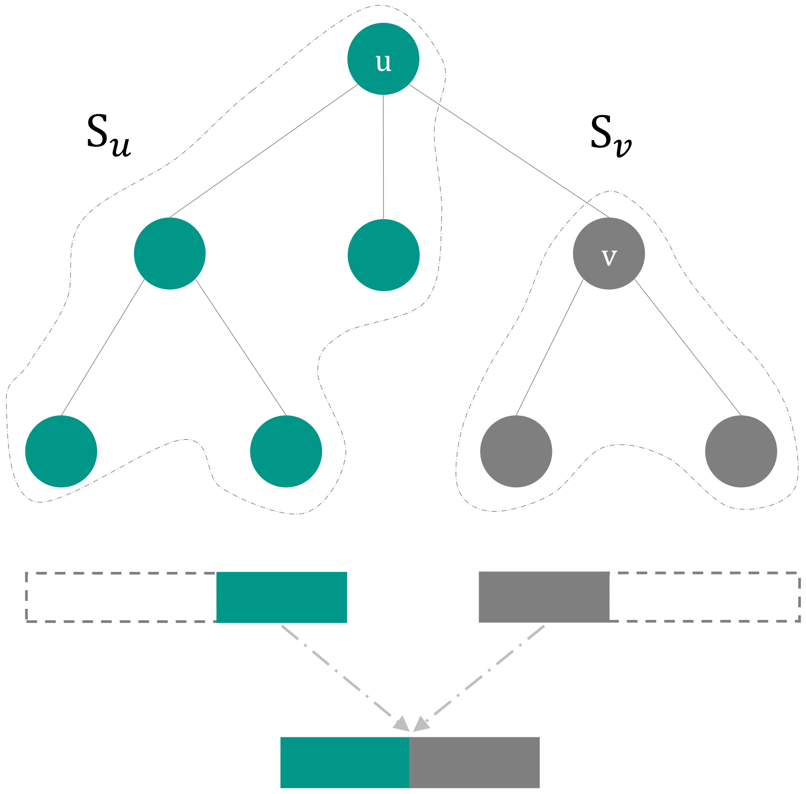
两个节点 u, v 仅会在 $\mathrm{LCA}_{u,v}$ 处进行一次合并共有 N^2 个点对

故时间复杂度 $O(\mathrm{NW})$

若不同阶，也可从数形结合的角度理解

将已合子树序列记为 S_u 将，以 v 为根子树的 DFS 序列记为 S_v

合并过程为:



优化上下界的树形背包

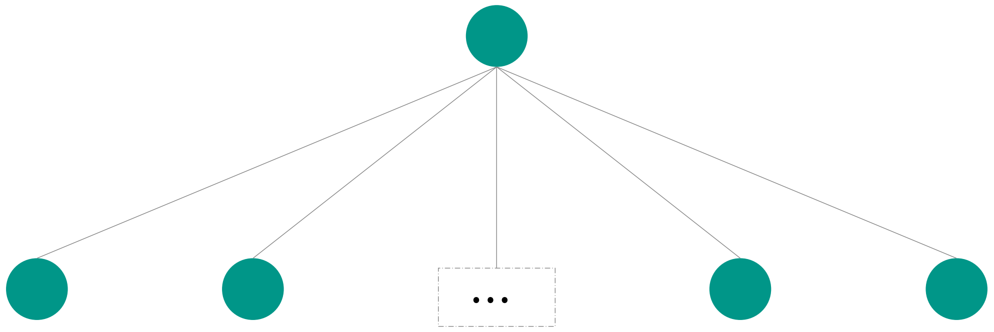
$\text{Suffix}(S_u, x)$ 与 $\text{Prefix}(S_u, x)$ 组合得到答案

其中 $x + y \le \text{W}$

那么总代价即为长度为 N 的序列中 长度不超过 W 的子段个数

显然至多有 NW 个

故时间复杂度 $O(\text{NW})$



当每个节点重量不为 1 时，优化上下界的算法时间复杂度 **不为** $O(\text{NW})$

如右图：

根节点重量为 1，其它节点都为根节点孩子 且 叶子节点且权值都为 W

当合并至 $\text{cnt}_1 \ge \text{W}$ 后，接下来的每次合并 $\text{dp}[1][j]$ 中 j 的上下界都为 $1 \sim \text{W}$

当 j 取值固定时 k 的上下界为 $\text{W} - j + 1 \sim \text{W}$ 那么一次合并的代价约为 $O(\text{W}^2)$

此时时间复杂度约为 $O(\text{NW}^2)$

#761、最大连续和

题目描述

给你一个长度为 n 的整数序列 $\{A_1, A_2, \cdots, A_n\}$

要求从中找出一段连续的长度不超过 m 的非空子序列,使得这个序列的和最大

输入格式

第一行为两个整数 n, m

第二行为 n 个用空格分开的整数序列,每个数的绝对值都小于 1000

输出格式

仅一个整数,表示连续长度不超过 m 的最大子序列和

样例输入

```
6 4
1 -3 5 1 -2 3
```

样例输出

```
7
```

维护前缀和 sum_i

设 $dp[i]$ 为以 A_i 结尾且长度不超过 m 的最大序列和

$$dp[i] = \max_{\max(0,i-m) \le j<i} \{\text{sum}_i - \text{sum}_j\}$$

朴素枚举时间复杂度 $O(n^2)$

当 sum_i 确定后，仅需保证 sum_j 最小即可令 $dp[i]$ 取得最值

不难看出窗口大小为 m，单调队列维护最小值更新答案即可

时间复杂度 $O(n)$

数据范围

对于 50% 的数据 $1 \le N, M \le 10^4$

对于 100% 的数据 $1 \le N, M \le 2 \times 10^5$

#765、烽火传递

题目描述

烽火台是重要的军事防御设施,一般建在交通要道或险要处

一旦有军情发生,则白天用浓烟,晚上有火光传递军情

在某两个城市之间有 n 座烽火台,每个烽火台发出信号都有一定的代价

为了使情报准确传递,在连续 m 个烽火台中至少要有一个发出信号

现在输入 n, m 和每个烽火台的代价,请计算总共最少的代价在两城市之间来准确传递情报

输入格式

第一行是 n, m ,表示 n 个烽火台和连续烽火台数 m

第二行 n 个整数表示每个烽火台的代价 $\mathrm{a_i}$

输出格式

输出仅一个整数,表示最小代价

数据范围

对于全部数据 $1 \leq \mathrm{n}, \mathrm{m} \leq 2 \times 10^5, 1 \leq \mathrm{a_i} \leq 1000$

设 $\mathrm{dp}[i]$ 表示前 i 个烽火台已传递且第 i 个点燃的最小代价

$$\mathrm{dp}[i] = \min_{\max(0, i-m) \le j < i} \{ \mathrm{dp}[j] + a_i \}$$

直接枚举 j 时间复杂度 $O(nm)$

维护窗口大小为 m 的单调递增队列

答案为

$$\min_{n-m+1 \le i \le n} \{ \mathrm{dp}[i] \}$$

时间复杂度 $O(n)$

#762、修剪草坪

题目描述

在一年前赢得了小镇的最佳草坪比赛后，FJ 变得很懒,再也没有修剪过草坪

现在,新一轮的最佳草坪比赛又开始了，FJ 希望能够再次夺冠

然而 FJ 的草坪非常脏乱，因此 FJ 只能够让他的奶牛来完成这项工作

FJ 有 N 只排成一排的奶牛,编号为 $1 \sim N$.

每只奶牛的效率是不同的，奶牛 i 的效率为 E_i

靠近的奶牛们很熟悉,如果 FJ 安排超过 K 只连续的奶牛，那么这些奶牛就会罢工去开派对

因此,现在 FJ 需要你的帮助，计算 FJ 可以得到的最大效率，并且该方案中没有连续的超过 K 只奶牛

输入格式

第一行：空格隔开的两个整数 N, K

第二到 $N+1$ 行：第 $i+1$ 行有一个整数 E_i

输出格式

一行一个值,表示 FJ 可以得到的最大的效率值

数据范围

对于全部数 $1 \leq N \leq 10^5, 0 \leq E_i \leq 10^9, 1 \leq k \leq 2.5 \times 10^4$

思路1

不妨求出每 $k+1$ 个至少不选一个的最小效率

设 $dp[i]$ 为前 i 头且第 i 头不选的最小损失效率

由于 $E_i \geq 0$ 若连续 $K+1$ 头不选显然非最优

$$dp[i] = \min_{\max(0, i-k-1) \leq j < i} \{ dp[j] \} + w_i$$

单调队列维护即可，答案为

$$\left(\sum_{i=1}^{n} E_i\right) - \min_{n-k \leq i \leq n}(dp[i])$$

时间复杂度 $O(N)$

#762、修剪草坪

思路2

维护前缀和 sum_i，设 $\text{dp}[i]$ 考虑前 i 头牛的最大效率和

若第 i 头不取

$$\text{dp}[i] = \text{dp}[i-1]$$

若第 i 头取 考虑枚举上一头不取的牛 j ，即 $j \sim i$ 都取

$$\text{dp}[i] = \max_{0 \le i-j \le k}\{\text{dp}[j-1] + \text{sum}_i - \text{sum}_j\}$$

直接枚举 j 时间复杂度 $O(nk)$，可将 sum_i可以视作常量

$$\text{dp}[i] = \max_{0 \le i-j \le k}\{\text{dp}[j-1] - \text{sum}_j\} + \text{sum}_i$$

不难看出要求窗口大小为 k 的最大值，单调队列维护即可

答案为 $\text{dp}[n]$，时间复杂度 $O(n)$

#763、旅行问题

题目描述

John 打算驾驶一辆汽车周游一个环形公路

公路上总共有 n 车站,每站都有若干升汽油(有的站可能油量为零)
每升油可以让汽车行驶一千米

John 必须从某个车站出发一直按顺时针(或逆时针)方向走遍所有的车站,并回到起点

在一开始的时候,汽车内油量为零, John 每到一个车站就把该站所有的油都带上(起点站亦是如此
行驶过程中不能出现没有油的情况

任务：判断以每个车站为起点能否按条件成功周游一周

输入格式

第一行是一个整数 n ,表示环形公路上的车站数

接下来 n 行,每行两个整数 p_i, d_i
分别表示表示第 i 号车站的存油量和第 i 号车站到下一站的距离

输出格式

输出共 n 行

如果从第 i 号车站出发
一直按顺时针(或逆时针)方向行驶,能够成功周游一圈,则在第 i 行输出 `TAK` ,否则输出 `NIE`

对于任意一个站点 i 能够顺时针到达下一个站点 $i+1$

需满足 $p_i - d_i \geq 0$

记 $\text{sum}_i = \sum_{j=1}^{i}(p_j - d_j)$

数据范围

对于全部数据, $3 \leq n \leq 10^6, 0 \leq p_i \leq 2 \times 10^9, 0 < d_i \leq 2 \times 10^9$

#763、旅行问题

对于站点 i 若其能够顺时针完成一圈

$\forall\, i \le j \le i+n-1$满足 $\mathrm{sum}_j - \mathrm{sum}_{i-1} \ge 0$

对于每一个点 i 直接枚举 j 验证，时间复杂度 $O(n)$

若

$$\min_{i\le j\le i+n-1}\{\,\mathrm{sum}_j\,\} \ge \mathrm{sum}_i$$

那么一定可以保证可以顺时针到达

$2n \to 1$ 枚举每个点 i 维护窗口大小为 n 的单调增队列，取出对头验证即可

同理，对于站点 i 能够逆时针到达下一个站点 $i+1$，需要满足 $p_i - d_{i-1} \ge 0$

求出后缀和 $1 \to 2n$ 枚举每个点 i，维护窗口大小为 n 的单调减队列验证即可

总时间复杂度 $O(n)$

#2887、单调队列优化多重背包

题目描述

有 N 种物品和一个容量是 V 的背包

其中第 i 种物品最多有 $\mathrm{k_i}$ 件

每件体积 $\mathrm{w_i}$ 价值 $\mathrm{v_i}$

求解将哪些物品装入背包，可使物品体积总和不超过背包容量，且价值总和最大

输出最大价值

输入格式

第一行两个整数 $\mathrm{N, V}$ ，表示物品种数和背包容积。

接下来有 N 行

每行三个整数 $\mathrm{w_i, v_i, k_i}$,表示第 i 种物品的体积、价值和数量

输出格式

输出一个整数,表示最大价值

数据范围

对于全部得数据 $1 \le \mathrm{N} \le 1000, 1 \le \mathrm{V} \le 30000, 1 \le \mathrm{w_i, v_i, k_i} \le 30000$

多重背包—单调队列优化

记 $r = j \bmod w_i$ 观察状态转移方程

$$\mathrm{dp}[i][j] = \max_{\substack{0 \le k \le \mathrm{K}_i \\ j \ge k \times \mathrm{w}_i}} \{ \mathrm{dp}[i-1][j - k \times \mathrm{w}_i] + k \times \mathrm{v}_i \}$$

$$\mathrm{dp}[i][j - \mathrm{w}_i] = \max_{\substack{0 \le k+1 \le \mathrm{K}_i \\ j \ge (k+1) \times \mathrm{w}_i}} \{ \mathrm{dp}[i-1][j - (k+1) \times \mathrm{w}_i] + k \times \mathrm{v}_i \}$$

$$\mathrm{dp}[i][j - 2 \times \mathrm{w}_i] = \max_{\substack{0 \le k+2 \le \mathrm{K}_i \\ j \ge (k+2) \times \mathrm{w}_i}} \{ \mathrm{dp}[i-1][j - (k+2) \times \mathrm{w}_i] + k \times \mathrm{v}_i \}$$

$$\cdots$$

$$\mathrm{dp}[i][r + w_i] = \max\{ \mathrm{dp}[i-1][r + \mathrm{w}_i], \mathrm{dp}[i-1][r] + \mathrm{v}_i \}$$

$$\mathrm{dp}[i][r] = \mathrm{dp}[i-1][r]$$

对于 $dp[i][j]$ 其仅依赖于前 $i-1$ 件物品背包容量为 r 的剩余类的状态有关，可将上述转移方程写为

多重背包—单调队列优化

$$\mathrm{dp}[i][j]=\max_{x\in\{k\times \mathrm{w}_i+r,\ k\in\mathbb{N}\}}\left\{\mathrm{dp}[i-1][x]+\frac{j-x}{\mathrm{w}_i}\times \mathrm{v}_i\right\}$$

不妨枚举 r 再从小到大枚举背包容量

该问题为在至多 k_i+1 个状态中找出最大值

考虑使用单调队列优化

对于每一个 r 需要维护 $\left\lfloor\frac{V}{r}\right\rfloor$ 个状态

每个dp$[i][j]$ 找出最大值的转移均摊时间复杂度为$O(1)$

总时间复杂度为

$$O\left(\mathrm{N}\times \mathrm{w}_i\times\left\lfloor\frac{\mathrm{V}}{\mathrm{w}_i}\right\rfloor\right)\le O(\mathrm{NV})$$

#2888、赤壁之战

题目描述

给定一个长度为 N 的序列 A ，求 A 有多少个长度为 M 的严格递增子序列

输入格式

第一行包含整数 T ,表示共有 T 组测试数据

每组数据,第一行包含两个整数 N 和 M

第二行包含 N 个整数,表示完整的序列 A

输出格式

每组数据输出一个结果,每个结果占一行

输出格式为 `Case #x: y` , x 为数据组别序号,从 1 开始, y 为结果

由于答案可能很大,请你输出对 1000000007 取模后的结果

数据范围

对于全部的数据 $1 \leq \mathrm{T} \leq 100, 1 \leq \mathrm{M} \leq \mathrm{N} \leq 1000, \sum_{t=1}^{i} \mathrm{N_i} \times \mathrm{M_i} \leq 10^7$

序列中的整数的绝对值不超过 10^9

设 $\mathrm{dp}[i][j]$ 为长度为 i 以 $a[j]$ 结尾的严格上升子序列方案数

答案为

$$\sum_{i=1}^{n} \mathrm{dp}[m][i]$$

显然 $\mathrm{dp}[1][j] = 1$

#2888、赤壁之战

对于 $i > 2$ 有

$$\mathrm{dp}[i][j] = \sum_{\substack{k<i \\ \mathrm{A}_k < \mathrm{A}_i}} \mathrm{dp}[i-1][k]$$

直接枚举并转移时间复杂度 $O(n^2 m)$

上述做法性能瓶颈在于统计前缀小于 A_j 的 $\mathrm{dp}[i-1][k]$

将 a_i 离散化从前往后扫描，对于每一个长度 i 建立一棵线段树/树状数组

只需要统计 $-\infty \sim \mathrm{A}_j - 1$ 中长度为 $i-1$ 的方案数(前缀区间查询)

对于每个 A_j 统计完后,需要在 A_j 位置加上 $\mathrm{dp}[i-1][j]$ 的方案数(单点修改)

时间复杂度 $O(n\, m \log n)$

#2889、牛棚清洁

题目描述

FJ 的奶牛们从小娇生惯养,她们无法容忍牛棚里的任何脏东西

FJ 发现如果要使这群有洁癖的奶牛满意,他必须雇佣她们中的一些来清扫牛棚, FJ 的奶牛中有 N 头愿意通过清扫牛棚挣一些零花钱

由于在某个时段中奶牛们会在牛棚里随时随地地乱扔垃圾,自然地,她们要求在这段时间里,无论什么时候至少要有一头奶牛正在打扫

需要打扫的时段从某一天的第 M 秒开始，到第 E 秒结束
注意这里的秒是指时间段而不是时间点,也就是说,每天需要打扫的总时间是 $\mathrm{E}-\mathrm{M}+1$ 秒

FJ 已经从每头牛那里得到了她们愿意接受的工作计划:
对于某一头牛,她每天都愿意在第 $\mathrm{T}_1\ldots\mathrm{T}_2$ 秒的时间段内工作,所要求的报酬是 S 美元

与需打扫时段的描述一样,如果一头奶牛愿意工作的时段是每天的第 $10\sim20$ 秒,那她总共工作的时间是 11 秒,而不是 10 秒

FJ 一旦决定雇佣某一头奶牛,就必须付给她全额的工资,而不能只让她工作一段时间,然后再按这段时间在她愿意工作的总时间中所占的百分比来决定她的工资

现在请你帮 FJ 决定该雇佣哪些奶牛以保持牛棚的清洁,当然,在能让奶牛们满意的前提下, FJ 希望使总花费尽量小

输入格式

第 1 行输入三个正整数 $\mathrm{N},\mathrm{M},\mathrm{E}$

第 $2\sim\mathrm{N}+1$ 行每行输入三个正整数 $\mathrm{T}_1,\mathrm{T}_2,\mathrm{S}$

数据规模

对于全部的数据 $1\leq n\leq 10000, 0\leq\mathrm{M}\leq\mathrm{E}\leq 86399, \mathrm{M}\leq\mathrm{T}_1\leq\mathrm{T}_2\leq\mathrm{E}, 0\leq\mathrm{S}\leq 500000$

输出格式

输出一个整数,表示 FJ 需要为牛棚清理工作支付的最少费用

如果清理工作不可能完成,那么输出 -1

#2889、牛棚清洁

设 $\mathrm{dp}[i]$ 为覆盖时段 $[\mathrm{M}, i]$ 的最小代价

初始时令 $\mathrm{dp}[\mathrm{M}-1] \leftarrow 0$ 答案为 $\mathrm{dp}[\mathrm{E}]$

将所有区间按右端点排序，对于当前任务 $[l_i, r_i]$ 有

$$\mathrm{dp}[r_i] = \min_{l_i-1 \le x \le r_i-1} \{\, \mathrm{dp}[k] \,\} + s_i$$

直接枚举 k 时间复杂度 $O(\mathrm{E}^2)$

上述做法性能瓶颈在于找出 $a_i - 1 \sim r_i - 1$ 范围内的最小值(区间最值查询)

并需要修改 $\mathrm{dp}[r_i]$ (单点更新)，考虑使用线段树维护

时间复杂度 $O(n \log n)$

#2889、牛棚清洁

将每个任务看作从一条有向边

需要求将 M ~ E 所有点覆盖的最小代价

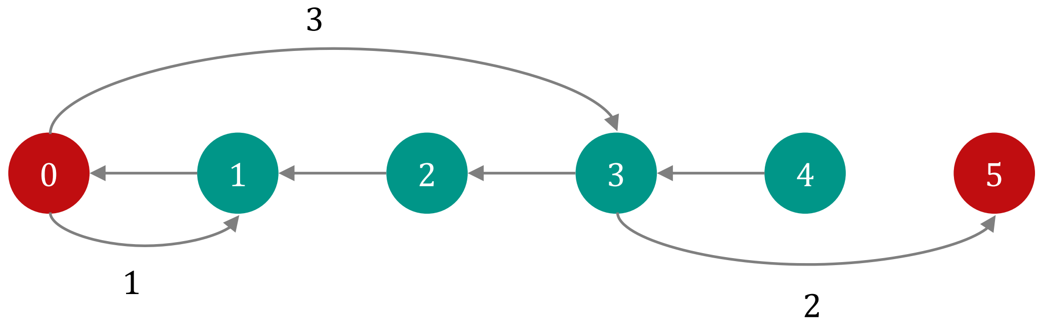
只有在一条边以 v 为终点或跨过 v 时才算 v 被覆盖

只有当一个点被覆盖后，才能作为新的起点

- 对于$i \in [\mathrm{L}, \mathrm{R}]$ 连一条 $i+1 \to i$ 边权为 0 的有向边
- 对于每个任务连一条 $l_i \to r_i + 1$ 边权为 s_i 的有向边

问题等价于从 $\mathrm{M} \to \mathrm{E}+1$ 的最短路径长度

使用 Dijkstra/SPFA 求解即可



#769、任务安排 1

实验舱 青少年编程 走近科学 走进名校

题目描述

有 N 个任务排成一个序列在一台机器上等待执行,它们的顺序不得改变

机器会把这 N 个任务分成若干批,每一批包含连续的若干个任务

从时刻 0 开始,任务被分批加工,执行第 i 个任务所需的时间是 T_i

另外,在每批任务开始前,机器需要 S 的启动时间

故执行一批任务所需的时间是启动时间 S 加上每个任务所需时间之和

一个任务执行后,将在机器中稍作等待,直至该批任务全部执行完毕

也就是说,同一批任务将在同一时刻完成

每个任务的费用是它的完成时刻乘以一个费用系数 C_i

请为机器规划一个分组方案,使得总费用最小

输入格式

第一行是 N , 第二行是 S

下面 N 行每行有一对正整数,分别为 T_i 和 C_i

表示第 i 个任务单独完成所需的时间是 T_i 及其费用系数 C_i

输出格式

一个数,最小的总费用

设 $dp[i][j]$ 为考虑前 i 个任务分 j 批执行的最小代价

答案为 $\min\limits_{1\le i\le n}\{\,\mathrm{dp}[n][i]\,\}$

$$\mathrm{dp}[i][j]=\min_{0\le k<i}\left\{\mathrm{dp}[k][j-1]+\left(S\times j+\sum_{u=1}^{i}T_u\right)\times\sum_{u=k+1}^{i}C_u\right\}$$

$\sum_{u=1}^{i}T_u$ 和 $\sum_{u=k+1}^{i}C_u$ 可预处理

时间复杂度 $O(n^3)$

数据范围

对于全部数据, $1\le N\le 5000, 0\le S\le 50, 1\le T_i, C_i\le 100$

#769、任务安排 1

状态中加入 j 这一维度是为了计算之前启动的耗时

若在时刻 i 启动机器该次的 S 将会影响 i 之后的所有任务

即对全局增加了 $\mathrm{S} \times \sum_{k=j+1}^{n} \mathrm{C}_k$ 的影响

不妨将该次 S 对后续的影响提前计算

设 $dp[i]$ 为考虑前 i 个任务执行的最小代价，答案为 $\mathrm{dp}[n]$

$$\mathrm{dp}[i] = \min_{0 \le j < i} \left\{ \mathrm{dp}[j] + \left(\mathrm{S} \times \sum_{k=j+1}^{n} \mathrm{C}_k \right) + \left(\sum_{k=1}^{i} \mathrm{T}_k \right) \times \left(\sum_{k=j+1}^{i} \mathrm{C}_k \right) \right\}$$

其中 $\sum_{k=j+1}^{n} \mathrm{C}_k$，$\sum_{k=1}^{i} \mathrm{T}_k$，$\sum_{k=j+1}^{i} \mathrm{C}_k$ 可使前缀和优化

时间复杂度 $O(n^2)$

斜率、截距

斜率

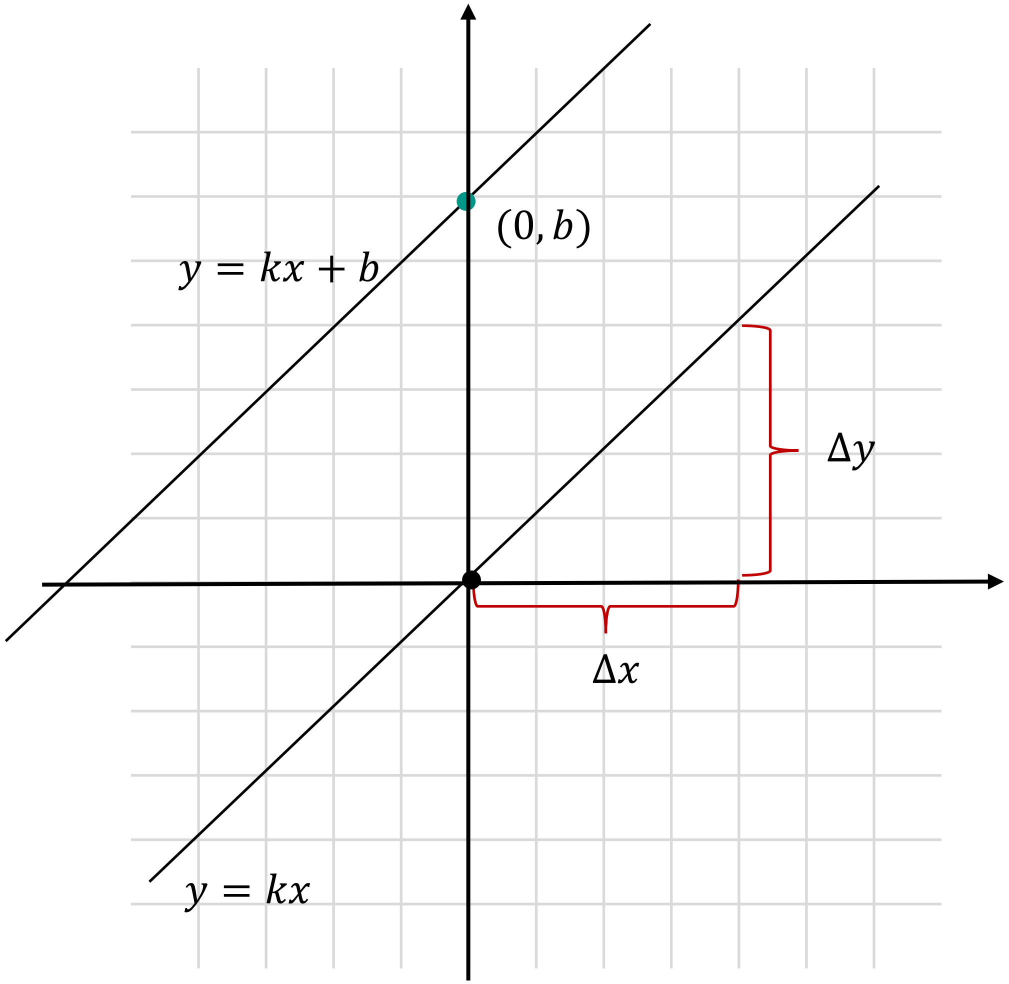
斜率反映直线的倾斜程度，一般用 k 表示

直线向上方向与 x 轴正方向构成的夹角为倾斜角，记为 α

$$k = \tan\alpha$$

对于直线 L 上不重合的两点 $(x_1, y_1), (x_2, y_2)$

$$k = \frac{y_2 - y_1}{x_2 - x_1}$$



截距

直线与 y 轴交点 $(0, b)$ 将其纵坐标 b 称为截距

斜截式

直线可由斜率 k 与截距 b 唯一确定，形如

$$y = kx + b$$

的方程称为直线斜截式

#770、任务安排 2

实验舱 青少年编程 走近科学 走进名校 LAB

题目描述

有 N 个任务排成一个序列在一台机器上等待执行,它们的顺序不得改变

机器会把这 N 个任务分成若干批,每一批包含连续的若干个任务

从时刻 0 开始,任务被分批加工,执行第 i 个任务所需的时间是 T_i

另外,在每批任务开始前,机器需要 S 的启动时间,故执行一批任务所需的时间是启动时间 S 加上每个任务所需时间之和

一个任务执行后,将在机器中稍作等待,直至该批任务全部执行完毕

也就是说,同一批任务将在同一时刻完成

每个任务的费用是它的完成时刻乘以一个费用系数 C_i

请为机器规划一个分组方案,使得总费用最小

记

$$\mathrm{sumT}_i = \sum_{j=1}^{i} \mathrm{T}_i$$

$$\mathrm{sumC}_i = \sum_{j=1}^{i} \mathrm{C}_i$$

输入格式

第一行是 N , 第二行是 S

下面 N 行每行有一对正整数,分别为 T_i 和 C_i
表示第 i 个任务单独完成所需的时间是 T_i 及其费用系数 C_i

输出格式

一个数,最小的总费用

数据范围

对于全部数据, $1 \le N \le 3 \times 10^5, 0 \le S \le 512, 1 \le T_i, C_i \le 512$

#770、任务安排 2

将 #769、任务安排 1 中状态转移方程可写为

$$\mathrm{dp}[i] = \min_{0 \le j < i} \left\{ \mathrm{dp}[j] + \left(\mathrm{S} \times \left(\mathrm{sumC}_n - \mathrm{sumC}_j \right) \right) + \mathrm{sumT}_i \times \left(\mathrm{sumC}_i - \mathrm{sumC}_j \right) \right\}$$

将其整理

$$\mathrm{dp}[i] = \min_{0 \le j < i} \{ \mathrm{dp}[j] - (\mathrm{S} + \mathrm{sumT}_i) \times \mathrm{sumC}_j \} + \mathrm{sumT}_i \times \mathrm{sumC}_i + \mathrm{S} \times \mathrm{sumC}_n$$

不妨将 min 函数去掉，可将关于 i 的值 $\mathrm{dp}[i]$，sumC_i 看作变量其余部分看作常量

$$\underbrace{\mathrm{dp}[j]}_{y} = \overbrace{(\mathrm{S} + \mathrm{sumT}_i)}^{k} \times \underbrace{\mathrm{sumC}_j}_{x} + \overbrace{\mathrm{dp}[i] - \mathrm{sumT}_i \times \mathrm{sumC}_i - \mathrm{S} \times \mathrm{sumC}_n}^{B}$$

若将 sumC_j 看作横坐标 $\mathrm{dp}[j]$ 看作纵坐标

上述表达式为一条以 $\mathrm{S} + \mathrm{sumT}_i$ 为斜率 $\mathrm{dp}[i] - \mathrm{sumT}_i \times \mathrm{sumC}_i - \mathrm{S} \times \mathrm{sumC}_n$ 为截距的直线

#770、任务安排 2

可以看出

每个决策 j 都对应坐标系中的一个点 $(\mathrm{sumC}_j, \mathrm{dp}[j])$

每个待求解状态 $\mathrm{dp}[i]$ 都对应一条直线的截距

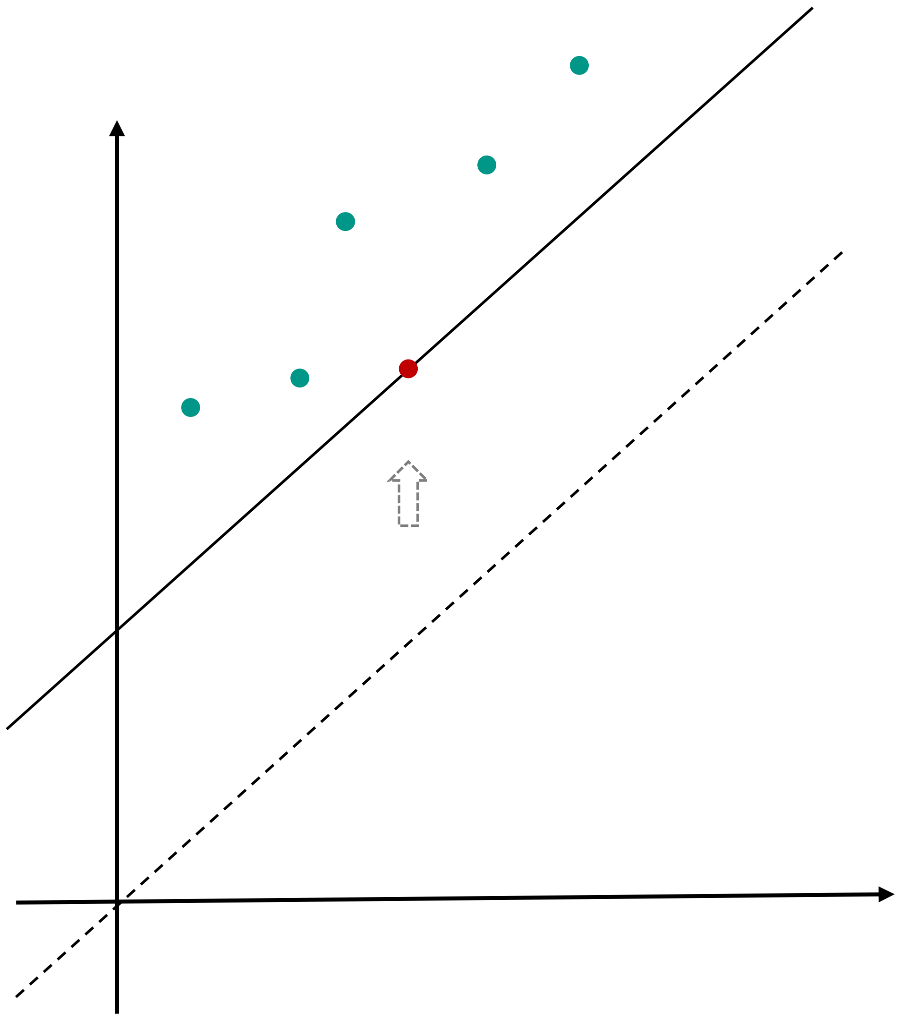
其中斜率为定值 $\mathrm{S} + \mathrm{sumT}_i$

当截距取取得最小值时 $\mathrm{dp}[i]$ 也取得最小值

当直线经过决策点 $(\mathrm{sumC}_j, \mathrm{dp}[j])$ 都可解出一个截距，截距最小即为最优决策

体现在直线上即为固定斜率 $(\mathrm{S} + \mathrm{sumT}_i)$ 将直线自下而上平移

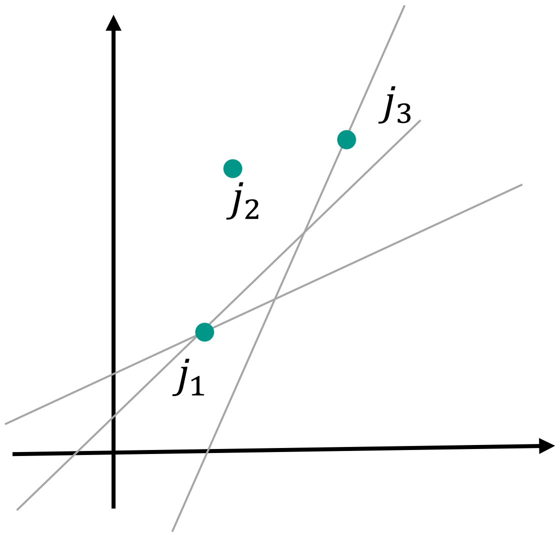
第一次接触到的即为最优决策点

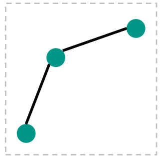


考虑任意三个决策点 $(\mathrm{sumC}_{j_1}, \mathrm{dp}[j_1])$，$(\mathrm{sumC}_{j_2}, \mathrm{dp}[j_2])$，$(\mathrm{sumC}_{j_2}, \mathrm{dp}[j_2])$

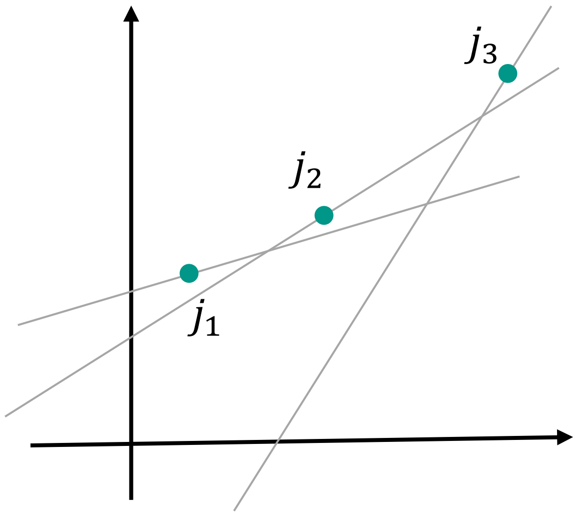
不妨设 $j_1 < j_2 < j_3$，因 C 非负所以有 $\mathrm{sumC}_{j_1} < \mathrm{sumC}_{j_2} < \mathrm{sumC}_{j_3}$

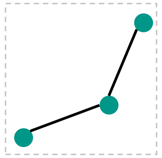
#770、任务安排 2





上凸





下凸

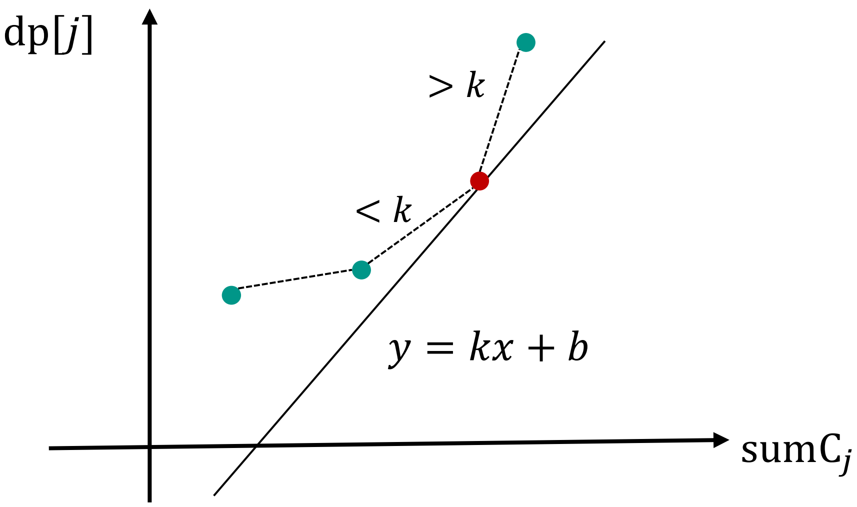
观察发现当 j_1,j_2 连线与 j_2,j_3 连线构成上凸形状无论斜率取何值 j_2 都不为最优决策

当 j_1,j_2 连线与 j_2,j_3 连线构成下凸形状 j_2 可能成为最优决策

j_2 可能成为最优决策点当且仅当

$$\frac{\mathrm{dp}[j_2]-\mathrm{dp}[j_1]}{\mathrm{sumC}_{j_2}-\mathrm{sumC}_{j_1}}<\frac{\mathrm{dp}[j_3]-\mathrm{dp}[j_2]}{\mathrm{sumC}_{j_3}-\mathrm{sumC}_{j_2}}$$

#770、任务安排 2



仅需在求解过程中维护下凸壳

发现最优决策点左侧斜率小于 k，右侧大于 k

本题中 $0 \le j < i$，随着 i 的增大每次新增一个决策点

由于 C 非负 sumC 单调非降即新增点位于之前点的右侧，同时 $S + sumT_i$ 也为单调非降

若仅保留凸壳上相邻两点斜率大于 $sumT_i + S$ 的部分，那么凸壳最左侧的点一定为最优决策点

考虑单调队列维护

#770、任务安排 2

- 对于队首的两个决策点 Q_l 和 Q_{l+1}，若

$$\frac{\mathrm{dp}[Q_{l+1}]-\mathrm{dp}[Q_l]}{\mathrm{sumC}_{Q_{l+1}}-\mathrm{sumC}_{Q_l}}\le S+\mathrm{sumT}_i$$

  则将队首出队，直到不满足条件

- 取队首作为最优决策点更新 $\mathrm{dp}[i]$
- 将新决策点 i 加入队列，若队尾两个决策点 Q_r 和 Q_{r-1} 与 i 满足

$$\frac{\mathrm{dp}[Q_r]-\mathrm{dp}[Q_{r-1}]}{\mathrm{sumC}_{Q_r}-\mathrm{sumC}_{Q_{r-1}}}\ge\frac{\mathrm{dp}[i]-\mathrm{dp}[Q_r]}{\mathrm{sumC}_i-\mathrm{sumC}_{Q_r}}$$

  则将队尾出队，直到不满足条件

计算斜率时为避免精度误差建议采用乘法，时间复杂度 $O(n)$

由于单调队列中依赖于斜率，所以该种优化方式被称为 **斜率优化**，也被称为 **凸壳优化 (convex hull trick)**

#771、任务安排 3

实验舱 青少年编程 LAB 走近科学 走进名校

题目描述

有 N 个任务排成一个序列在一台机器上等待执行,它们的顺序不得改变

机器会把这 N 个任务分成若干批,每一批包含连续的若干个任务

从时刻 0 开始,任务被分批加工,执行第 i 个任务所需的时间是 T_i

另外,在每批任务开始前,机器需要 S 的启动时间,故执行一批任务所需的时间是启动时间 S 加上每个任务所需时间之和

一个任务执行后,将在机器中稍作等待,直至该批任务全部执行完毕

也就是说,同一批任务将在同一时刻完成

每个任务的费用是它的完成时刻乘以一个费用系数 C_i

请为机器规划一个分组方案,使得总费用最小

输入格式

第一行是 N , 第二行是 S

下面 N 行每行有一对正整数,分别为 T_i 和 C_i
表示第 i 个任务单独完成所需的时间是 T_i 及其费用系数 C_i

输出格式

一个数,最小的总费用

数据范围

对于全部数据, $1 \leq N \leq 3 \times 10^5, 1 \leq S \leq 2^8, |T_i| \leq 2^8, 0 \leq C_i \leq 2^8$

#771、任务安排 3

本题中 T 可能为负，则意味着 sumT 不再有单调性

从而 $\mathrm{S} + \mathrm{sumT}_i$ 不具备单调性

相比于上一题并不能仅保留凸壳上相邻两点斜率大于 $\mathrm{sumT}_i + \mathrm{S}$ 的部分

依然使用单调队列维护下凸壳

由于队首不一定为最优决策点

那么可在单调中二分找出最优决策点

第一个满足左侧斜率小于 $\mathrm{sumT}_i + \mathrm{S}$ 右侧斜率大于 $\mathrm{sumT}_i + \mathrm{S}$ 的决策点

时间复杂度 $O(n\log n)$

实验舱
青少年编程
走近科学 走进名校

谢谢观看