

## E - Vibe Coding 解説

原案：shobonvip      問題準備：shobonvip

## E：問題概要

- ソフト開発をしよう！  $N$  個の工程からなり、好きな順番で実行できる
- 工程  $i$  を手作業で行うと、 $A_i$  時間かかる
- 工程  $i$  を生成 AI で行うと、今までの工程たちの  $B_j$  の総和と  $B_i$  の和だけ時間がかかる
- すべての工程を完了するまでの時間を最小化する

### 制約

$$N \leq 2 \times 10^5$$

$$1 \leq A_i \leq 10^9$$

$$1 \leq B_i \leq 10^8$$

## E: 例

| 工程 $i$ | 1  | 2 | 3  |
|--------|----|---|----|
| $A_i$  | 10 | 6 | 10 |
| $B_i$  | 5  | 2 | 4  |

工程 2 を AI:  $B_2 = 2$

工程 3 を AI:  $B_2 + B_3 = 2 + 4 = 6$

工程 1 を手作業:  $A_1 = 10$

合計:  $2 + 6 + 10 = 18$

これが最小らしい ( ' ・ ω ・ ` ) パット見分からない

## E：最適な実行順序

- 今のままでは選び方が多すぎる
- とりあえず各工程を AI / 手作業のどちらで行うか固定しよう そうすると…
- **AI を手作業より先に行うと損しない** ← Exchange Argument から分かる
- **AI に任せる工程は  $B_i$  の昇順にする** ←これも Exchange Argument

最適解の構造はこれ！

AI： $B_i$  の昇順 → 手作業：任意の順番

## E : $O(N^2)$ DP

$B_1 \geq B_2 \geq \dots \geq B_N$  (降順) となるように並べ直そう  
はじめに全工程を手作業で行うと仮定する。そして、

$f_i(k)$  = 先頭  $i$  個の工程で  $k$  個を AI にしたときの最大短縮時間

とする。すると漸化式ができる

$$f_i(k) = \max(f_{i-1}(k), f_{i-1}(k-1) + A_i - kB_i).$$

これで DP ができるので  $O(N^2)$  はできた…がまだ間に合わない

$$\text{答え} = \sum_{i=1}^N A_i - \max_k f_N(k), \quad \text{計算量} = O(N^2).$$

## E: DP の差分

$i$  個目の工程まで処理した DP の差分列を

$$d_i(k) = f_i(k) - f_i(k-1) \quad (1 \leq k \leq i)$$

とする。次の工程を  $(a, b)$  とすると、工程を AI にする方がよい条件は

$$f_i(k-1) + a - kb \geq f_i(k) \iff a - kb \geq d_i(k).$$

実は、次の主張が成り立つ！！！！！！！！ (´・ω・`)シャキーン

### すごい不変条件

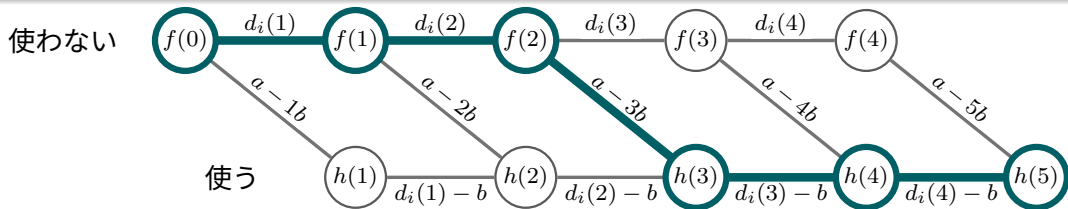
- ①  $d_i(k) - d_i(k+1) \geq B_i$
- ②  $a - kb \geq d_i(k)$  が一度成り立つと、それより大きい  $k$  でも成り立ち続ける

証明は帰納法で出来る 自分はこれ実験しないと分からなかった (´・ω・`)

## E: 差分列の更新

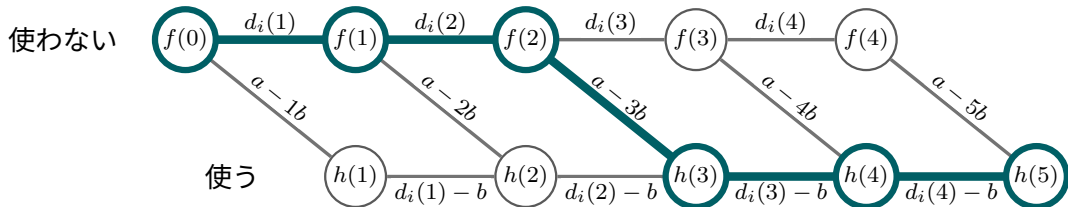
### すごい不変条件

- ①  $d_i(k) - d_i(k+1) \geq B_i$   
↑ これは元の列が凸より強いことを表す
- ②  $a - kb \geq d_i(k)$  が一度成り立つと、それより大きい  $k$  でも成り立ち続ける  
↑ AI の採用がバラバラに行われることはなく、すごく秩序がある



$$f(k) = f_i(k) \quad h(k) = f_i(k-1) + a - kb$$

## E : DP 高速化



$$f(k) = f_i(k) \quad h(k) = f_i(k-1) + a - kb$$

境界を  $q$  とすると、新しい差分列は

$$d_i(1), \dots, d_i(q-1), a - qb, d_i(q) - b, d_i(q+1) - b, \dots$$

二分探索で  $a - kb \geq d_i(k)$  となる境界が分かる  $\rightarrow$  DP 高速化できそう

## E: DP 高速化

$$d_i(1), \dots, d_i(q-1), a - qb, d_i(q) - b, d_i(q+1) - b, \dots.$$

まず、空の列  $d$  を用意する。次が高速にできればよい

- $a - kb \geq d_k$  となる最小の  $k$  を見つける
- $d$  の  $q$  番目に  $a - qb$  を挿入
- $d$  の区間に一斉に  $-b$  を足す

これは**平衡二分探索木**を使えば全部  $O(\log N)$  ができる！

merge-split 型の implicit Treap なら実装量は軽いかも

区間加算は遅延評価すれば ok (今回加算で可換なのでより楽) Treap なら split して根の lazy に足すだけ

## E: 計算量

- Treap で  $O(N \log N)$  時間、 $O(N)$  空間
- 平衡二分探索木ではなく、平方分割で実装してもよい  $O(N\sqrt{N})$  時間  
だがキャッシュが効いてめっちゃ速い

別解: 最小費用流で考察すると貪欲ができる (自分は最初こっちで考察してた)

- そのままだと  $O(N^2)$  時間
- CHT+平方分割で  $O(N\sqrt{N})$
- もう少し考察すれば再び treap で  $O(N \log N)$  になるらしい……?

実験なしならこっちの方が思い付くかも?

- AC teams / Trying teams
  - $1 + ? / 3$
- First Acceptance
  - Rinshan Solution (236 min)