時をかけるビ太郎
Bitaro, who Leaps through Time

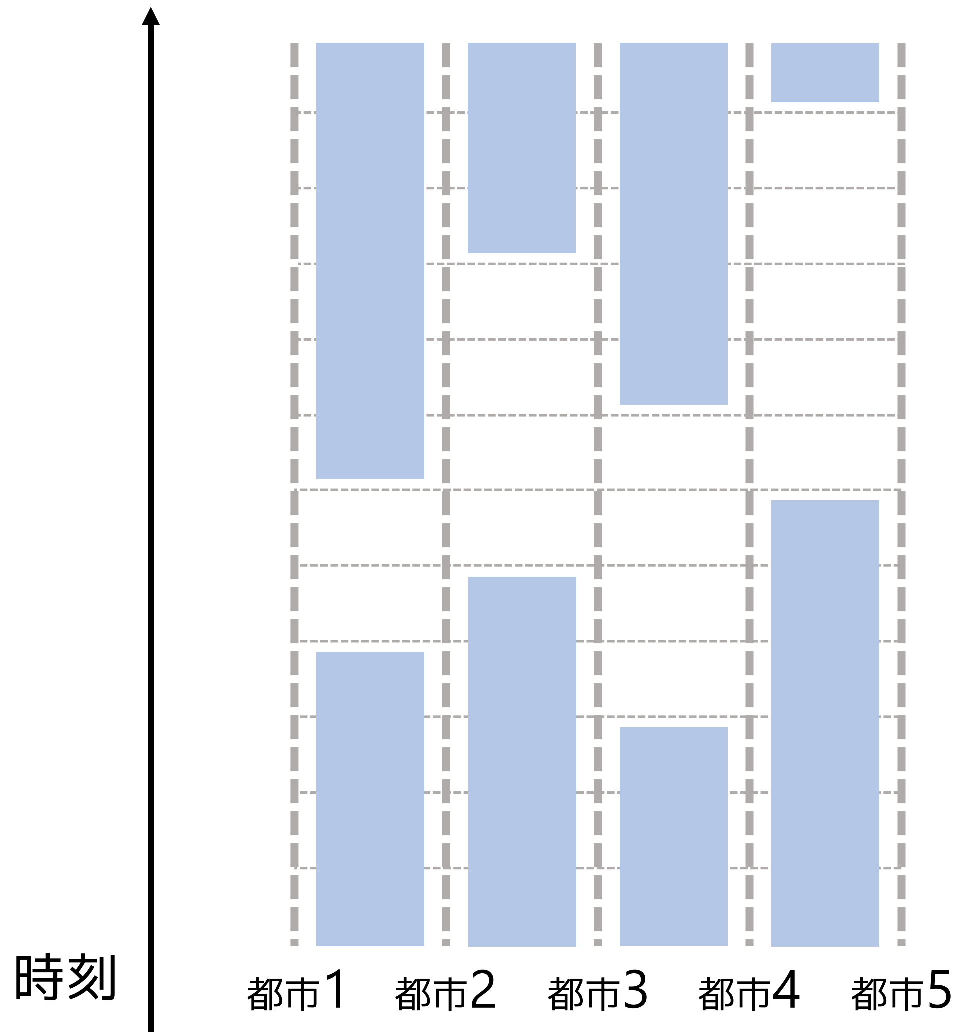
JOI2018/2019 Spring Training Camp Day3

wafrelka

問題概要

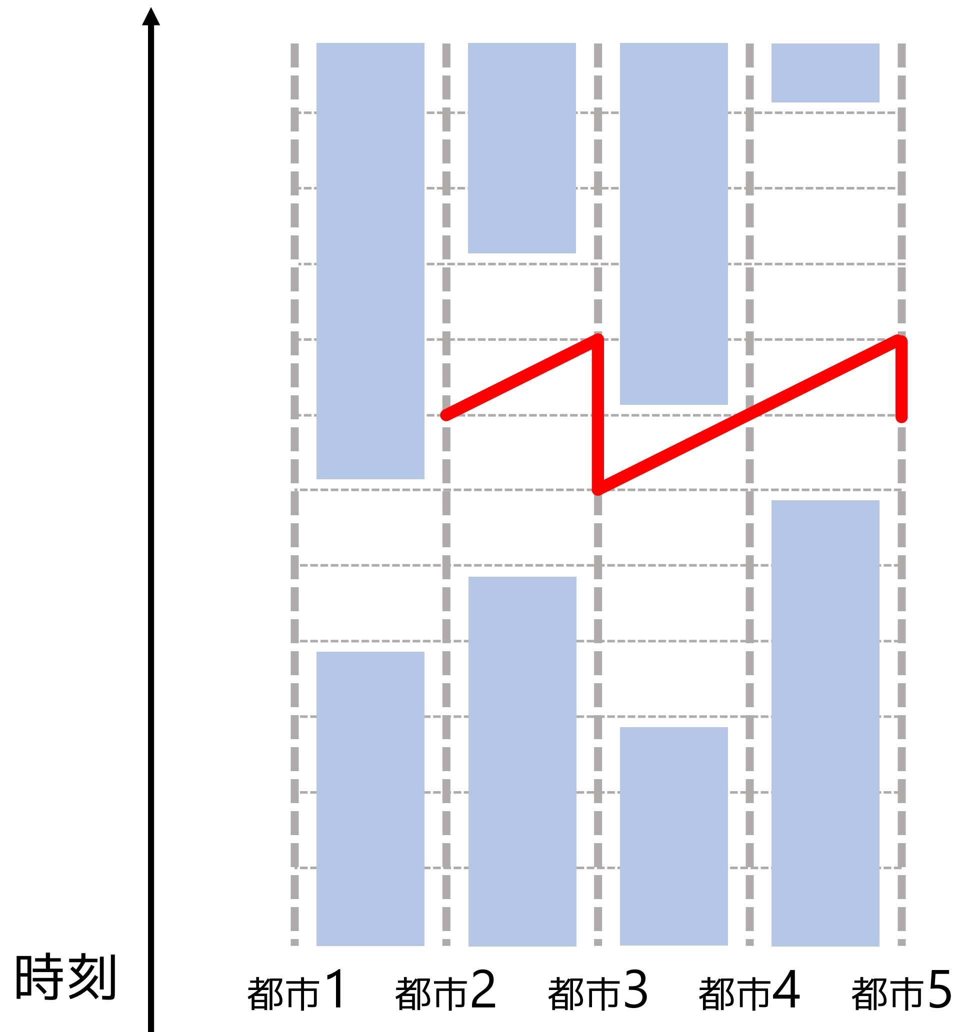
- N 個の都市が一列に並んでいる
- $N-1$ 本の道路がある
  - 道路 i を使うと都市 i と都市 $i+1$ を 1 秒で行き来できる
  - 道路 i は時刻 L_i から時刻 R_i の間しか開いていない
- ビ太郎は都市にいるときタイムリープができる
  - 1 回タイムリープをすると 1 秒だけ時間を遡れる
- 以下のクエリを処理してください
  - 道路 i が開いている時刻を S_j から E_j までに変更する
  - ビ太郎が時刻 B_j の時点で都市 A_j にいるとき
時刻 D_j の時点で都市 C_j にいるように移動・タイムリープをするときの最小のタイムリープ回数

問題概要 – サンプル



$\{(L_i, R_i)\} =$
$\{(3,5), (4,8), (2,6), (5,10)\}$

問題概要 – サンプル

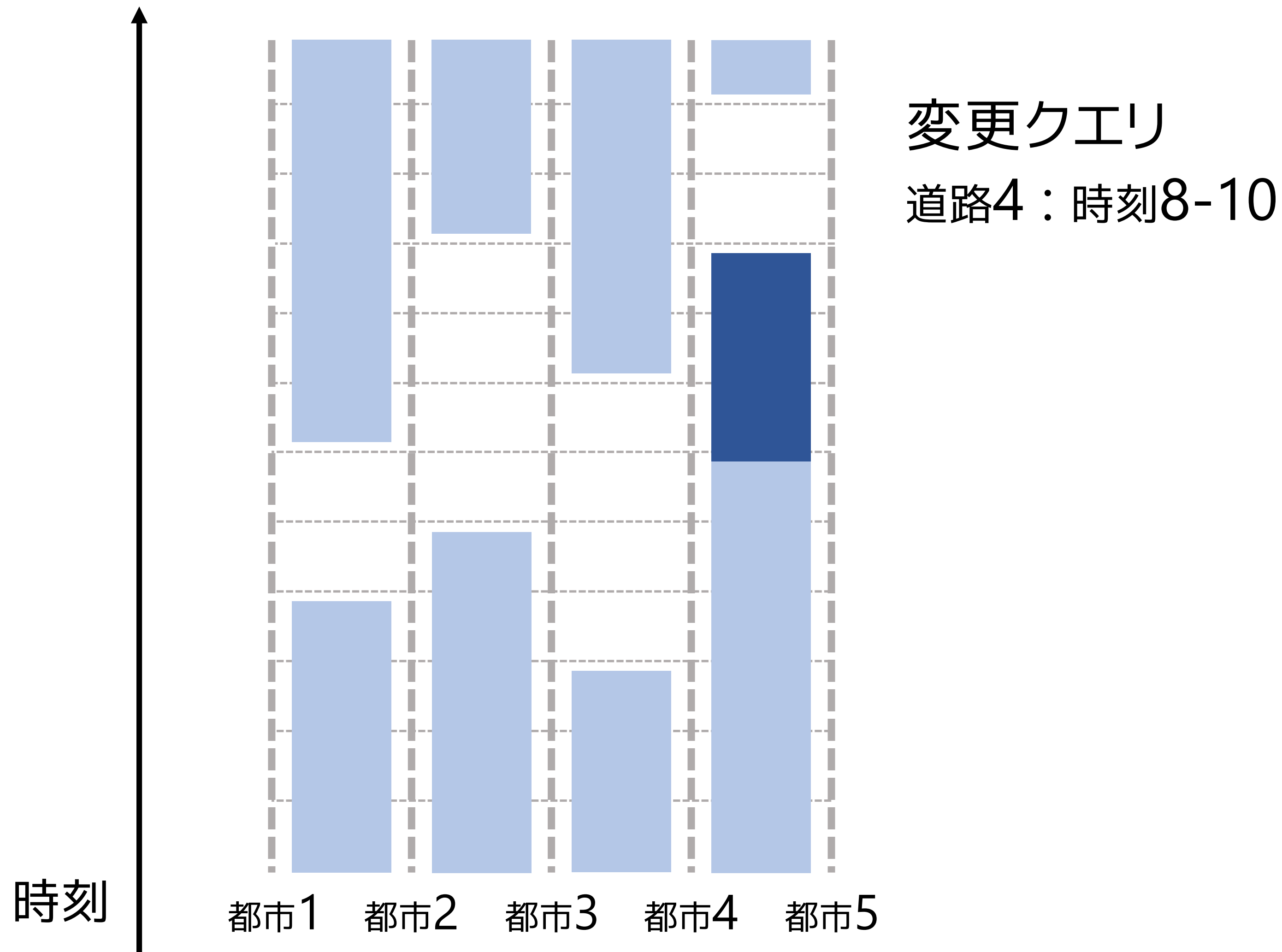


移動クエリ

都市2, 時刻6 → 都市5, 時刻6

タイムリープ3回

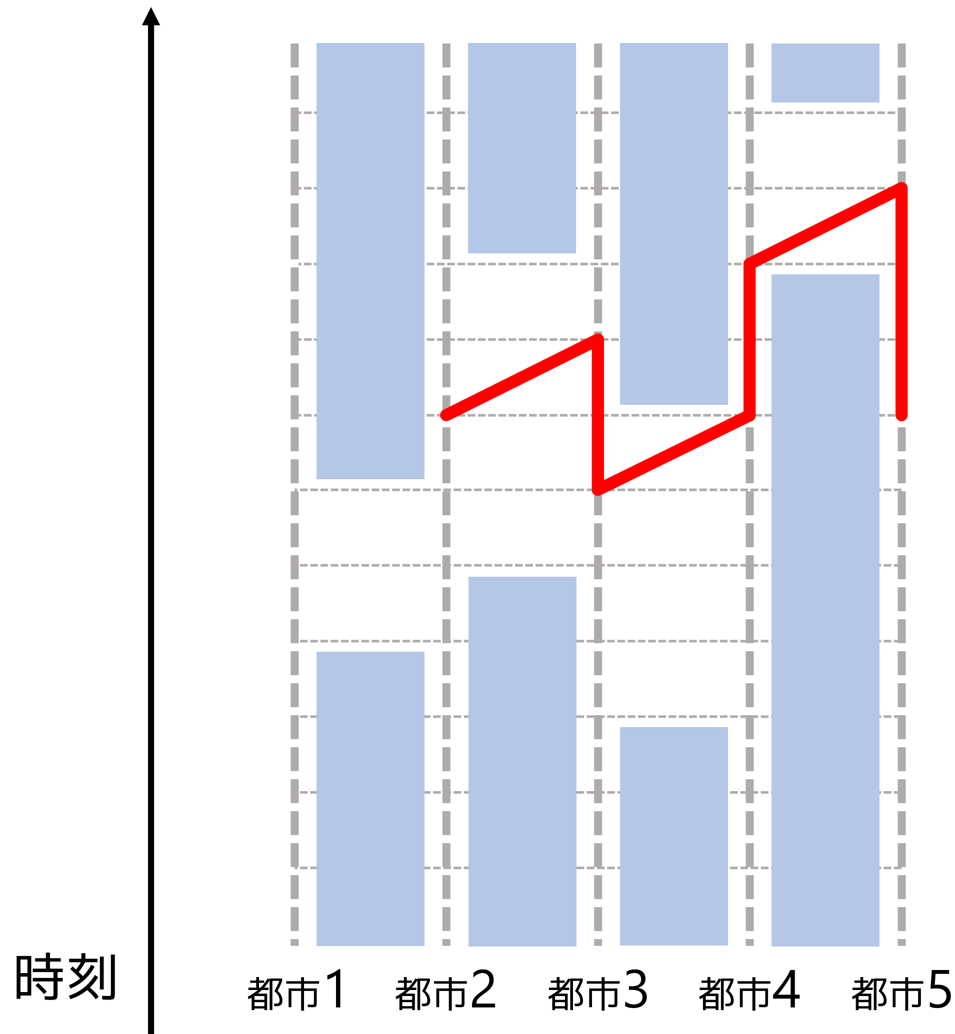
問題概要 – サンプル



変更クエリ

道路4：時刻8-10

問題概要 – サンプル



移動クエリ

都市2, 時刻6 → 都市5, 時刻6

タイムリープ5回

小課題 1

- 制約: $N \leq 1000, Q \leq 1000$
- 移動クエリごとにシミュレーションかな?

考察 – ビ太郎の行動

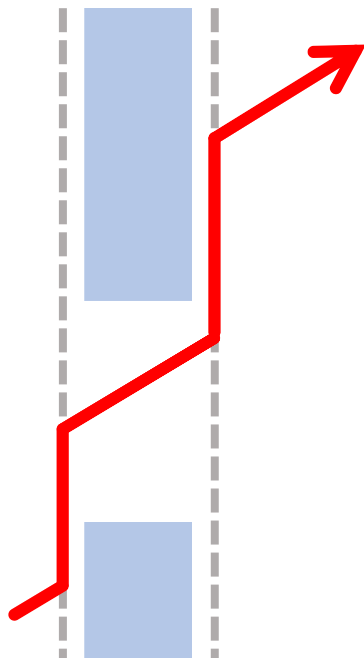
- ビ太郎の行動は 3 種類
  - 1 秒かけて隣の都市へ移動する
  - 都市で X 秒だけ待機する
  - 都市で Y 回タイムリープする
- これらの組み合わせを考えればよい

考察 – ビ太郎の行動

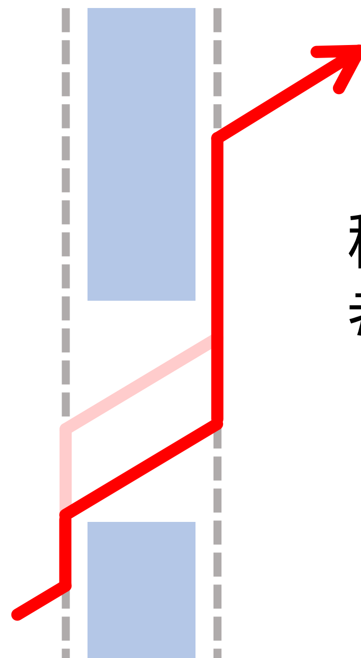
- タイムリープ回数が同じならば
  同じ都市へ行く複数の行動のうち
  到着時刻がいちばん早いものが嬉しい
- 目的地とは反対側の都市へ移動する必要はない
- → 都市 A_i から都市 B_i $(A_i < B_i)$ に移動するときは
  都市 $A_i, A_i + 1, \ldots, B_i$ の出発時刻だけ考えればよい

考察 – ビ太郎の行動

- 各都市での待機･タイムリープは必要最小限でよい
- 各都市の出発･到着時刻も整数値のみ考えればよい



移動A



移動B

移動Aの代わりに移動Bを考慮すればよい

部分点解法 1

- 移動クエリごとにシミュレーションする
- いまいる都市および現在時刻を記録しておく
- 目的地に向かって移動をする
- 道路が開いていないときは必要最低限の
  待機・タイムリープを行う

- $O(NQ)$ で小課題 1 が解けて 4 点が得られる

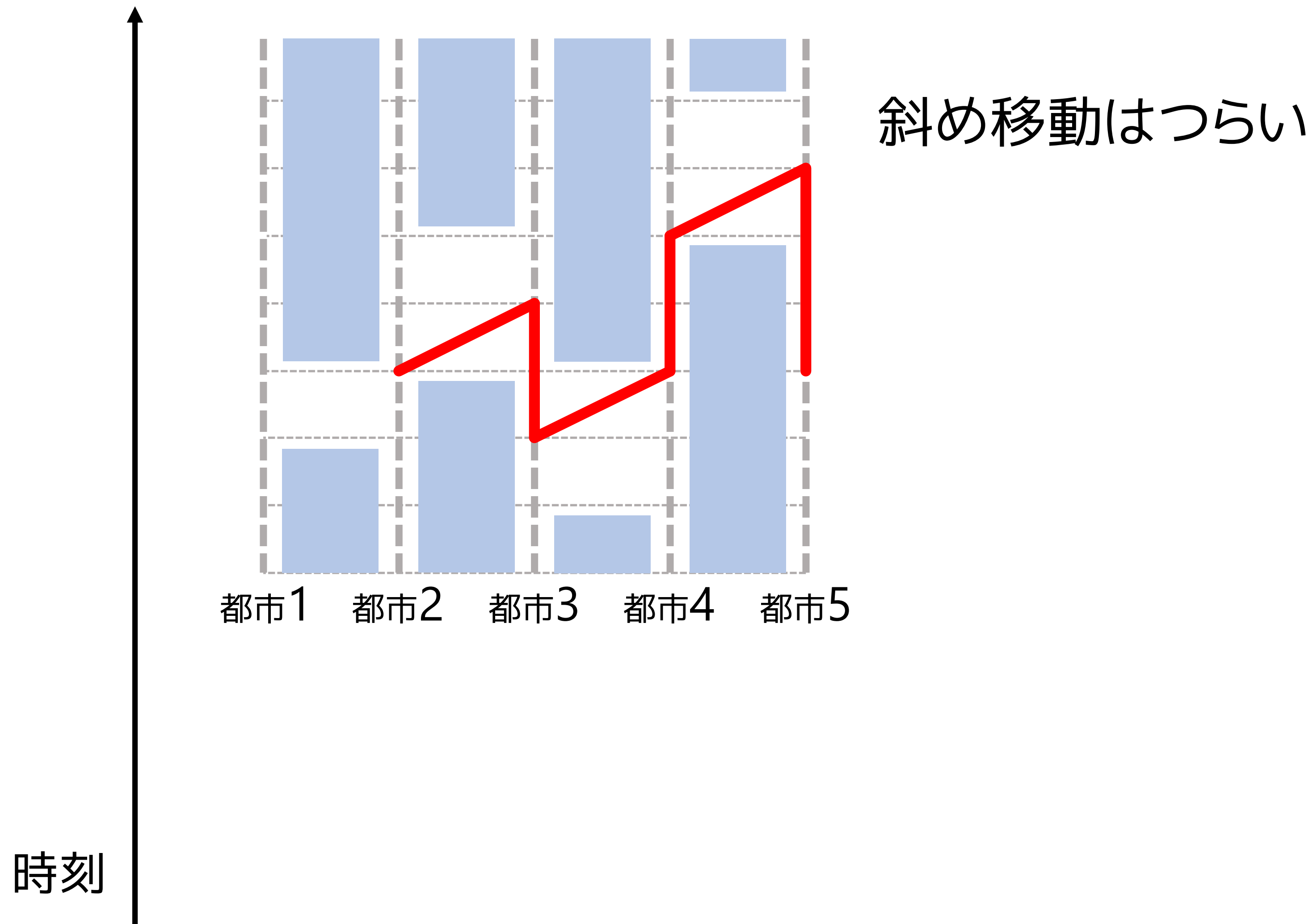
小課題 2

- 制約: $N \leq 300000, Q \leq 300000$
- 変更クエリがない

問題の変形 – 移動クエリ

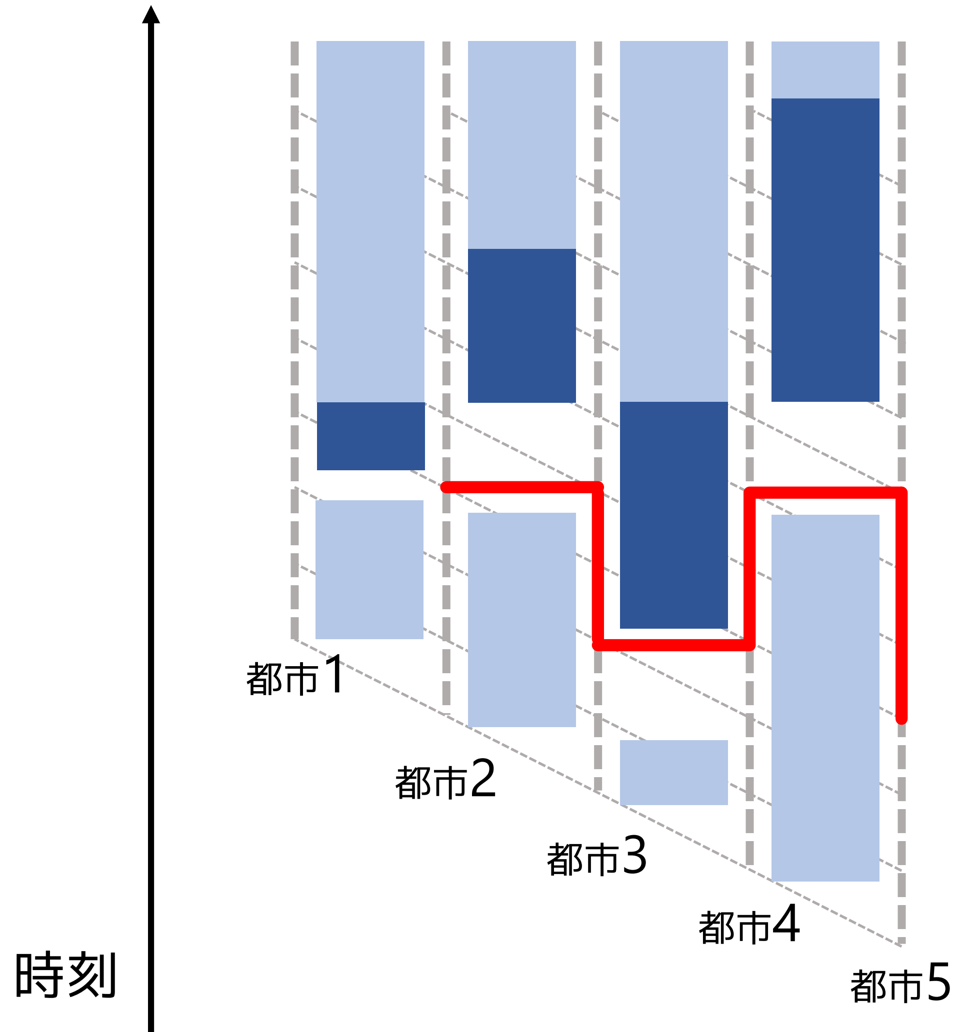
- 以降は都市 A_i から都市 B_i $(A_i < B_i)$ への移動クエリのみを考える
  - $A_i > B_i$ の場合は都市の順番をひっくり返して考えれば OK
  - $A_i = B_i$ の場合は明らか

問題の変形 – 時間軸



斜め移動はつらい

問題の変形 – 時間軸



時間軸を歪めて
都市 i の時刻が
i だけ進んでいることにする

移動コスト : 0
移動クエリ : $(B_i - A_i, D_i - C_i)$
道路 i : $\left(L_i - i, R_i - (i+1)\right)$

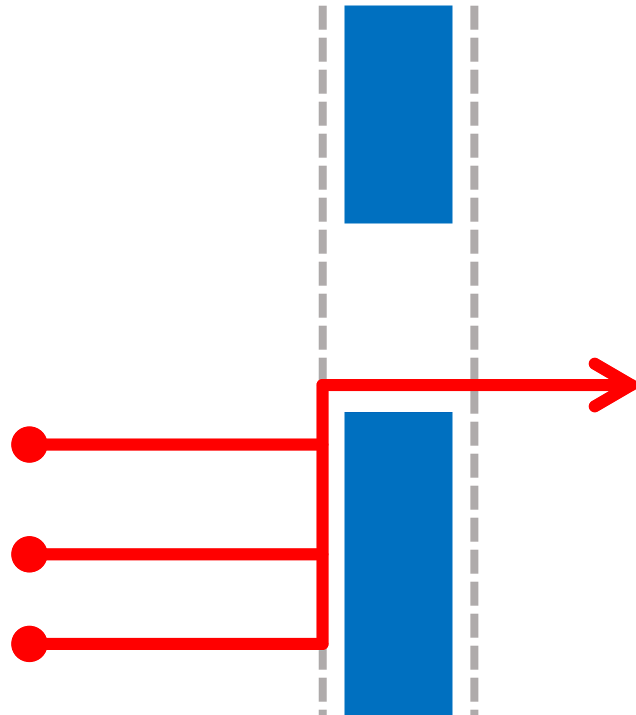
問題の変形

- 以下のような問題になった
- 長方形のブロックがいくつか並んでいる
- 以下のクエリを処理してください
  - ブロックの位置を変更する
  - 指定された始点から終点まで移動するときの下方向移動コストの最小回数を計算する

y
x

考察 – 共通化

- あるブロックにぶつかるような経路はすべて
  それ以降は同じような経路をたどる



最初にぶつかるブロック

- とある頂点から右に移動していったとき
  最初にぶつかるブロックは $O(N)$ で計算できる
- std::set を使って左側からスキャンするなどで
  頂点 K 個に対しても $O(N + K \log K)$ で計算できる
  - $x = 1$ から $x = N$ までスキャンしていく
  - $x = k$ に注目しているとき
    $x = k$ でブロックにぶつかる頂点をセットから削除して
    $x = k$ な頂点をセットに追加する
  - $x = k$ で上 (下) ブロックにぶつかる頂点は
    セットから y の値が最大 (最小) な頂点を取り出すのを
    繰り返すことで効率的に削除できる

ダブリング

- 各移動クエリの始点 + ブロックの角に関して
  そこから右側に進んだとき最初にぶつかるブロックを
  求める
- すると各始点から右側に進んだとき
  2^k 回目にぶつかるブロックが求められる (ダブリング)
- ブロックを回避するために必要なコストも
  同時に計算できる

部分点解法 2

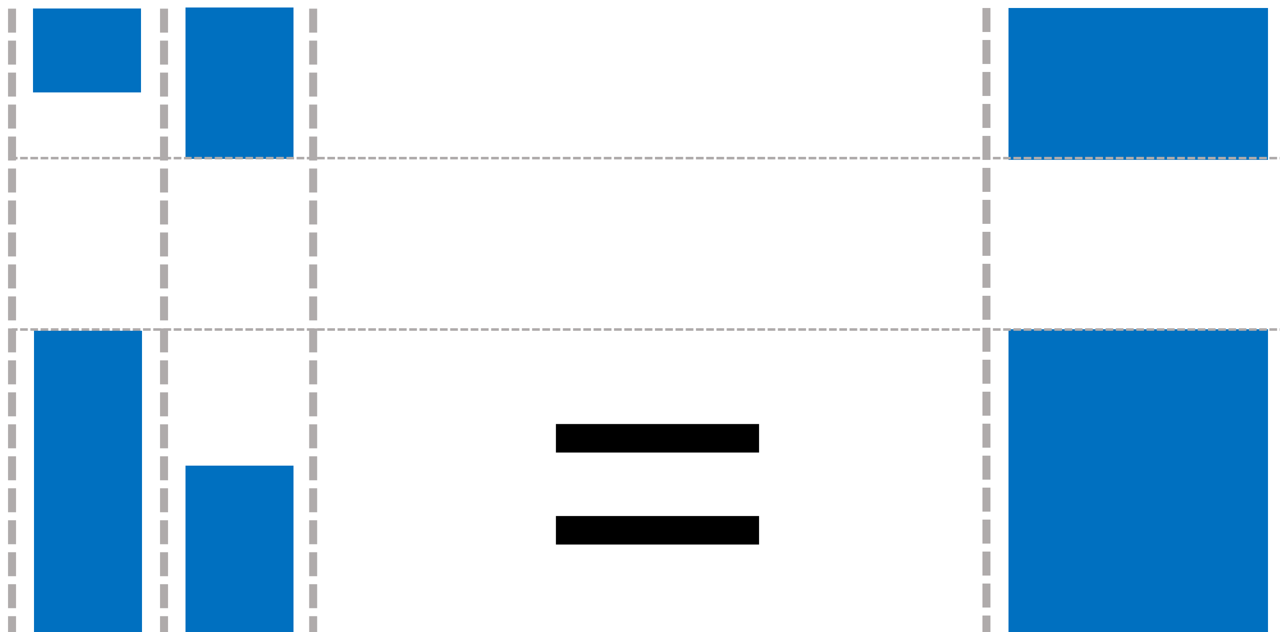
- 各移動クエリの始点 + 各ブロックの角に関して
ダブリングを使うことで 2^k 回目にぶつかるブロックと
そこまでのコストを求めておく
  - $O\left((N+Q)\log(N+Q)\right)$
- 各移動クエリに対して
終点にたどり着く前に最後にぶつかるブロックと
そこまでのコストを計算する
  - $O(\log N)$
- 変更クエリがなければ
全体として $O\left((N+Q)\log(N+Q)\right)$ 計算できる

小課題 3

- 制約: $N \leq 300000, Q \leq 300000$
- 変更クエリあり
- 部分点解法 2 は無理かな

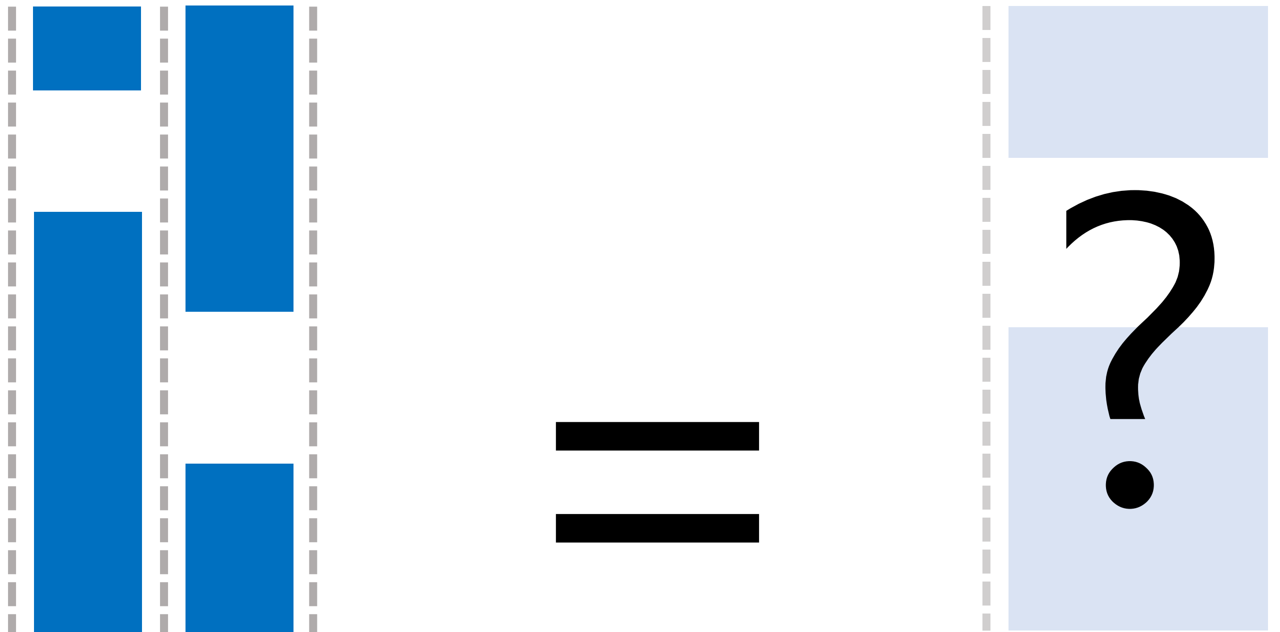
考察 – セグメントツリー?

- 左側の区間 2 つは右側の区間と同一視できる
- 通過後の座標およびコストが一致する
- セグメントツリーの機運かな?



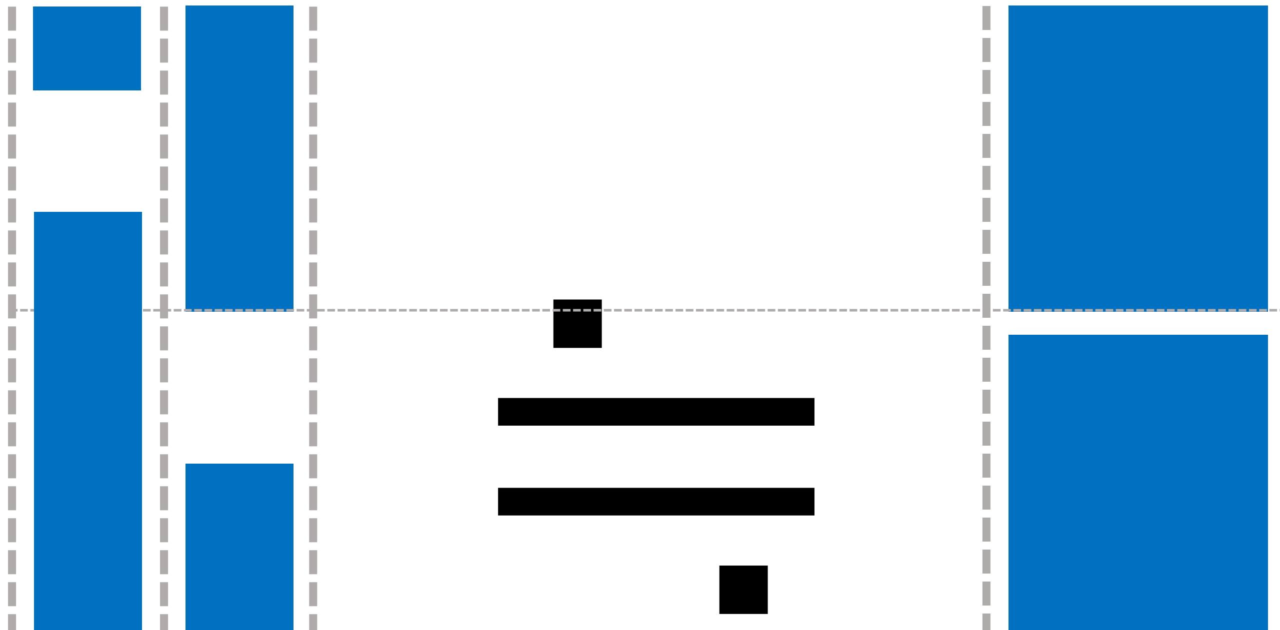
考察 – セグメントツリー?

- 左側の区間 2 つに対応する単体の区間は**存在しない**



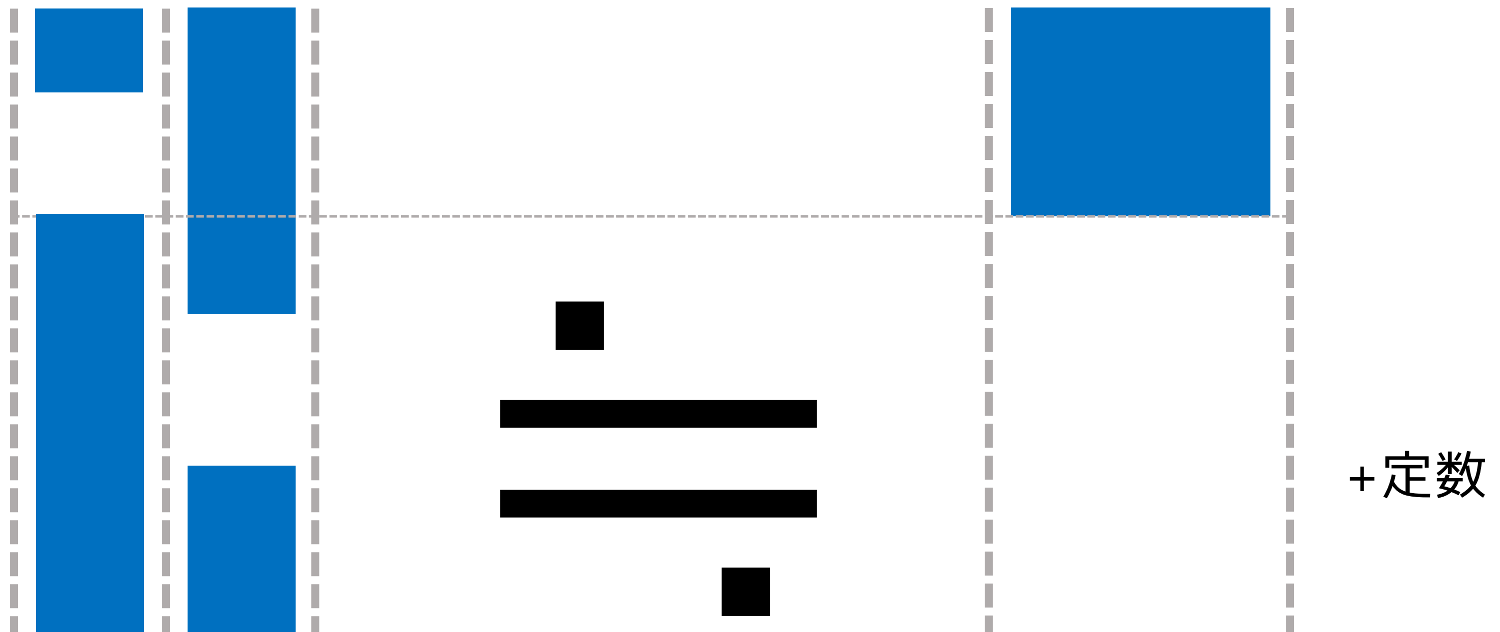
考察 – セグメントツリー?

- 通過後の座標が一致するものはある



考察 – セグメントツリー?

- 通過後のコストがおおよそ一致するものもある (定数だけずれる)



考察 – マージ

- 座標とコストを同時に一致させるようなものは存在しないことがあるのでうまくマージできない
- → 通過後の座標とコストを分解して考える

考察 – マージ

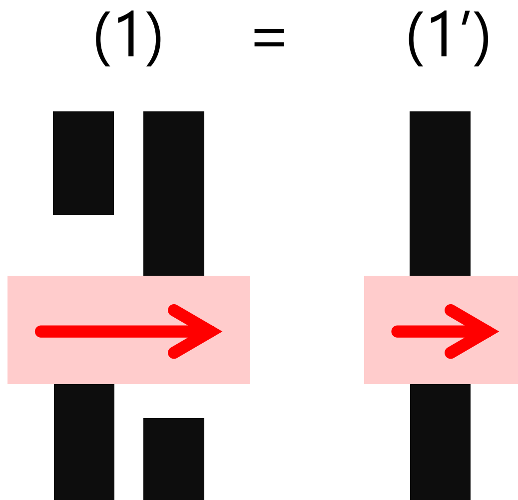
- 各区間は以下の 2 つのどちらかだと見なせる

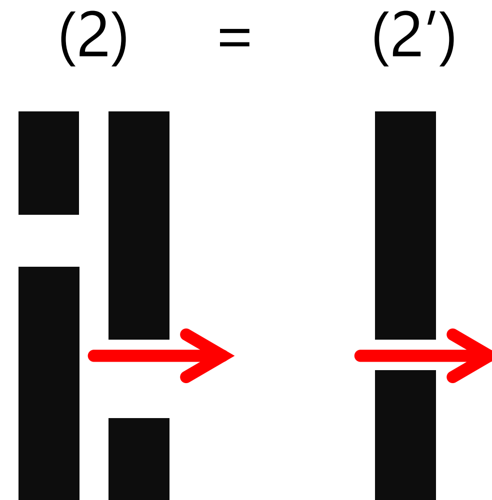
- a. 通過後の座標・コストが
  ある 1 つのブロック対を通過したときと一致する
- b. 通過後の座標は一意に定まり
  コストはとある上ブロックを通過したときと
  定数を除いて一致する

- **キーポイント**
  これらの区間を合成してもふたたび a, b の
  どちらかの種類の区間になる

マージ – 座標 (かんたん)

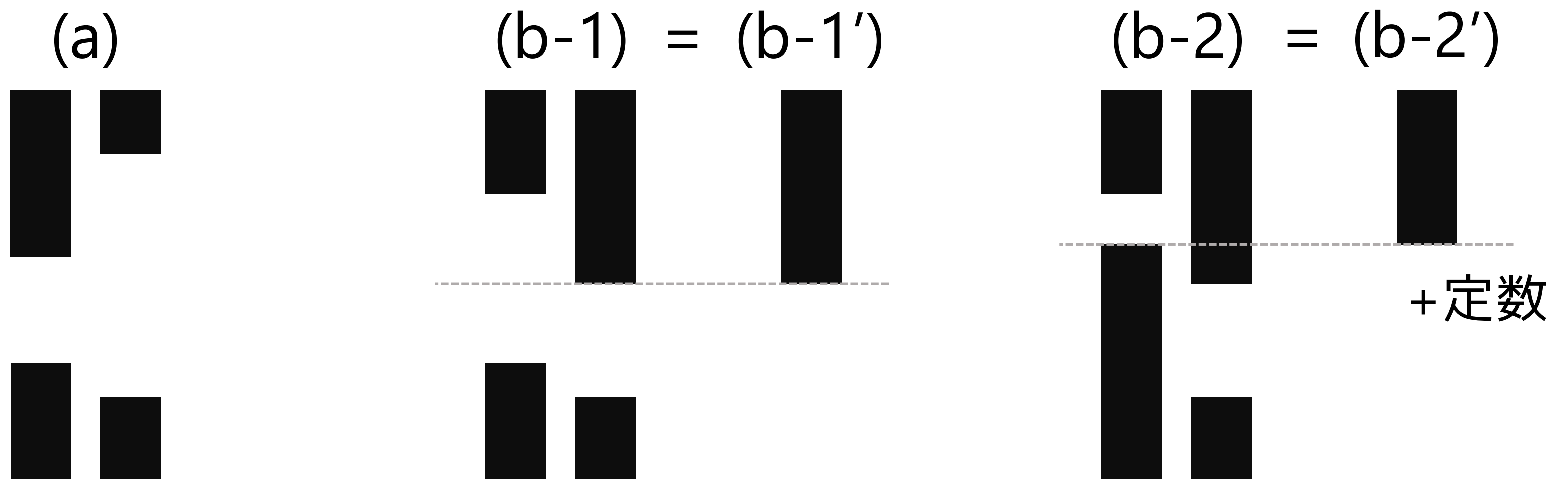
- (1) 空いている場所について共通部分があるなら合成結果は共通部分だけ空いているブロック対
- (2) 共通部分がないなら通過後の座標は右側のブロック対の端点のどちらか





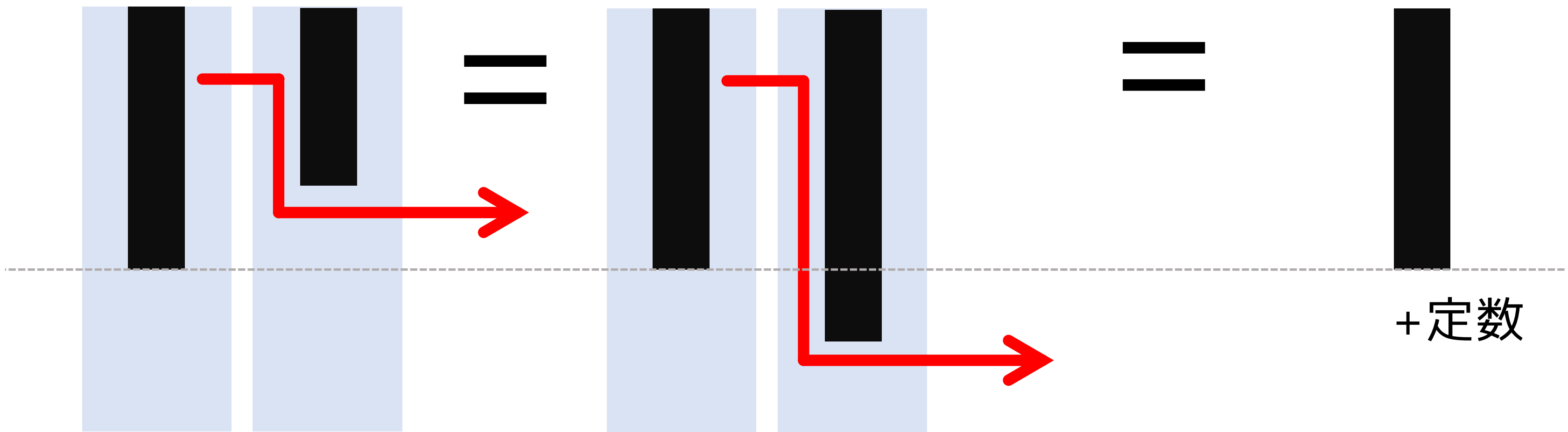
マージ – コスト (むずかしい)

- 左側の区間がタイプ a (座標･コスト一致) のとき
- 右側の区間のコスト担当の上ブロックについて
- (a) 左側の上ブロックより小さいとき
  合成後は左側の上ブロックと一致
- (b) そうでないとき
  合成後はとある上ブロック + 定数と一致



マージ – コスト (むずかしい)

- 左側の区間がタイプ b (座標･コスト不一致) のとき
- 左側の区間を通ると座標は一意に決まるので
  右側の区間を通るコストも一意に決まる
- もちろん各区間で足される定数コストも足す



マージ – 一貫性

- a+a, a+b, b+a, b+b の 4 つの組み合わせについて以下の 2 つを確認した
  - 座標のマージがうまくいく (タイプ a または b の形になる)
  - コストのマージがうまくいく (タイプ a または b の形になる)

- 座標とコストを組み合わせたときに座標はタイプ a の形だけどコストはタイプ b の形という事態が発生するとまずい
- そういう事態は起こりうるのか？

マージ – 一貫性

- 実は心配しなくてよい
- 座標がタイプ a の形になるのは
  タイプ a の区間を 2 つ足し合わせたとき
- タイプ a を 2 つ足し合わせても
  タイプ a,b のどちらかにしかならない (最初に確認した)

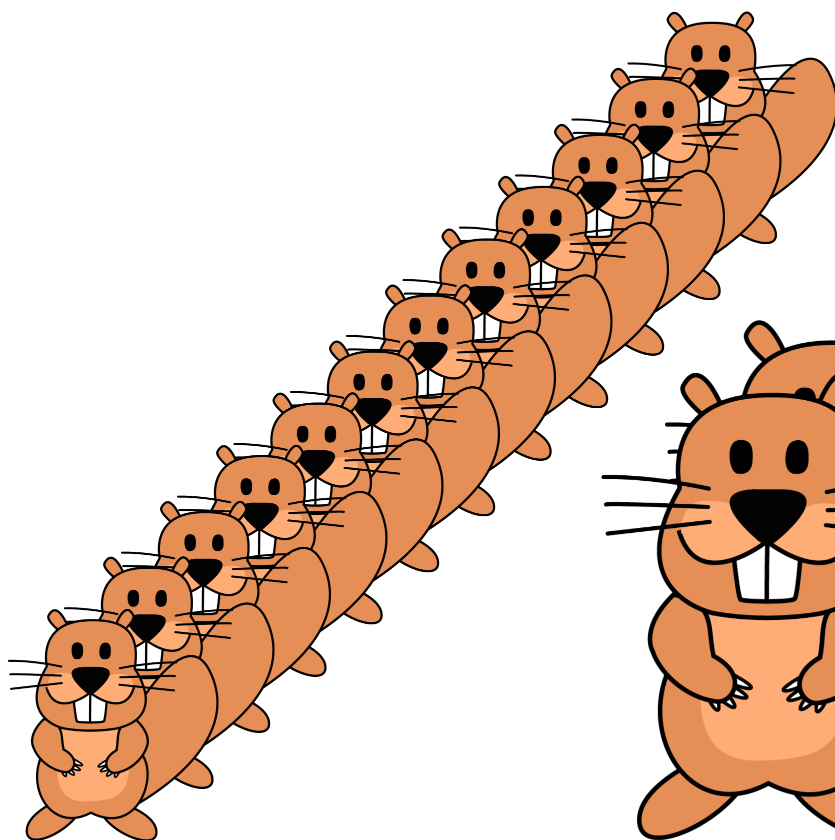
満点解法

- それぞれのブロック対を葉ノードとする
  セグメントツリーを構築する
- 変更クエリについて
  - セグメントツリーの葉ノードおよびその親たちを更新する
  - $O(\log N)$
- 移動クエリ $(x_1, y_1) \to (x_2, y_2)$ について
  - $x = x_1$ から $\mathrm{x} = x_2$ の間にあるブロック対をすべてマージ
  - 区間を通過した後のコスト･座標を計算
  - $O(\log N)$
- 全体として $O(Q \log N)$ で計算できる

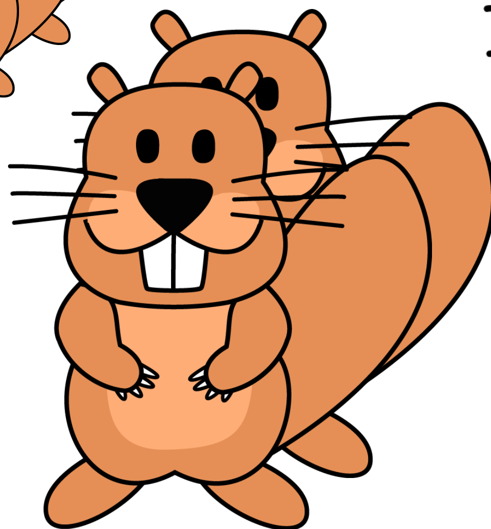
反省

- スライドを作った後に思いましたが
  コスト関数と座標関数の合成をするセグメントツリー
  だと思うこともできて
  そっちのほうが分かりやすかったかな……。

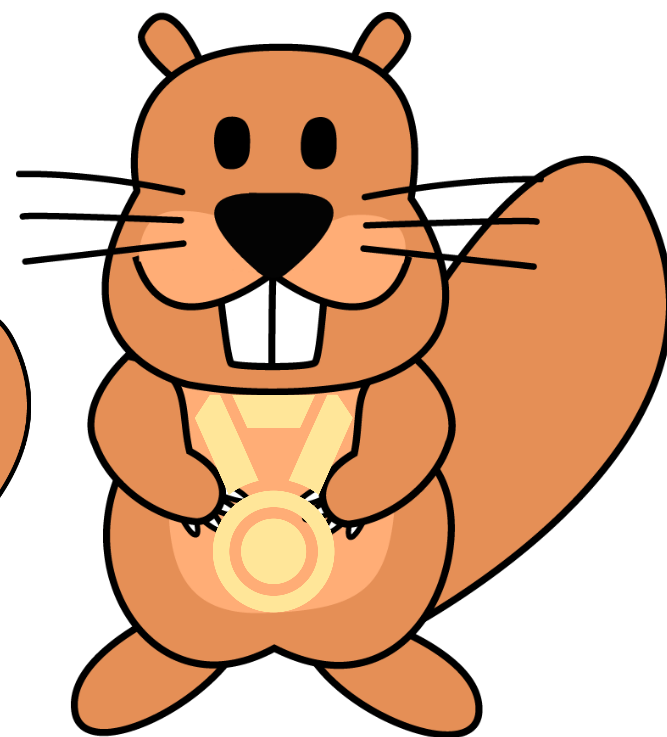
得点分布



4



34



100