




Matryoshka 解説

tozangezan

# 問題概要

- $N$  個のマトリョーシカ人形がある (底面の直径  $R_i$ 、高さ  $H_i$ )  

- $Q$  個のクエリが与えられる
- クエリ: 底面の直径  $A_i$  以上、高さ  $B_i$  以下のすべてのマトリョーシカ人形を取り出したとき、最小何個の**入れ子**にまとめることができるか？

# 例

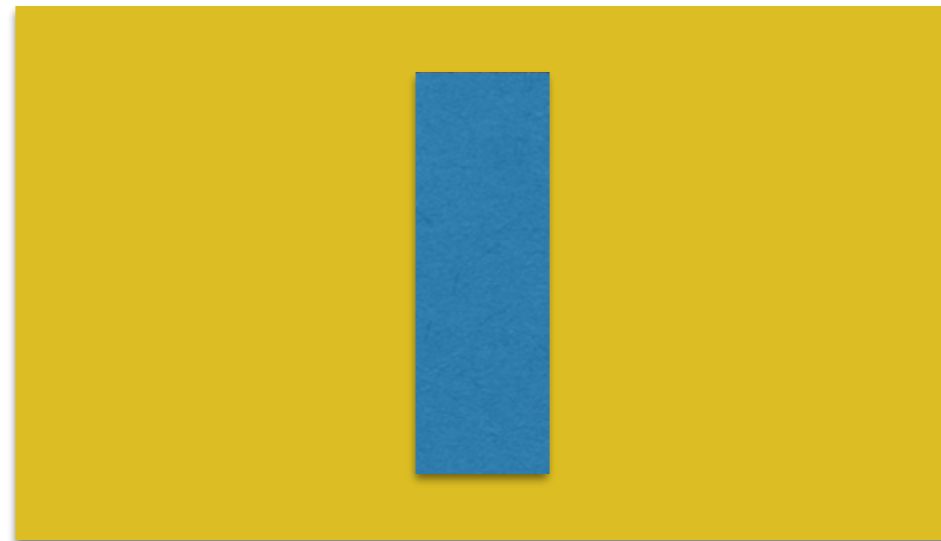
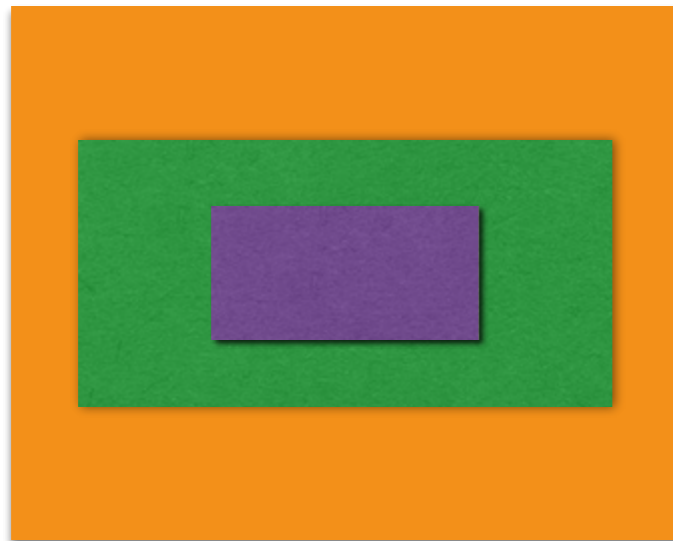
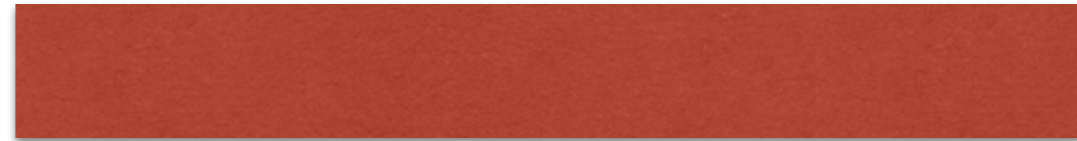
- $R_i = \{1, 4, 2, 8, 5, 7\}$
- $H_i = \{3, 2, 1, 1, 4, 4\}$



# 例

•  $R_i = \{1, 4, 2, 8, 5, 7\}$

•  $H_i = \{3, 2, 1, 1, 4, 4\} \rightarrow \text{答え: } 3$

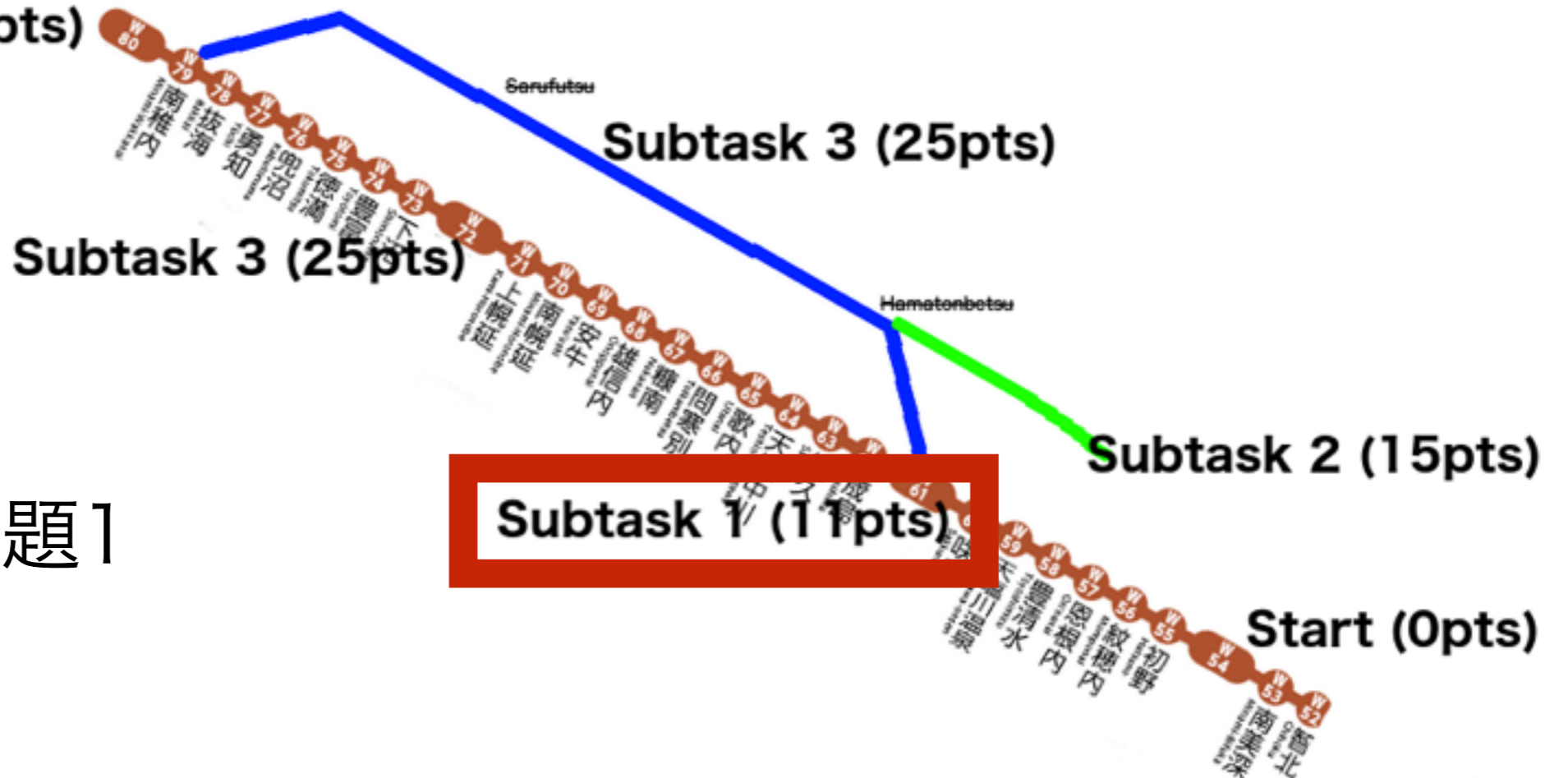




# 満点解法への道のり



Subtask 4 (49pts)



Next: 小課題1

# 小課題 1 (11点)

- ・ 全探索でどれの中にどれを入れていくかを試していきます
- ・ DFS で全探索 (枝刈りしなくても間に合います)
- ・ DFS で全探索するのに慣れてない人は PCK の過去問が練習になると思います

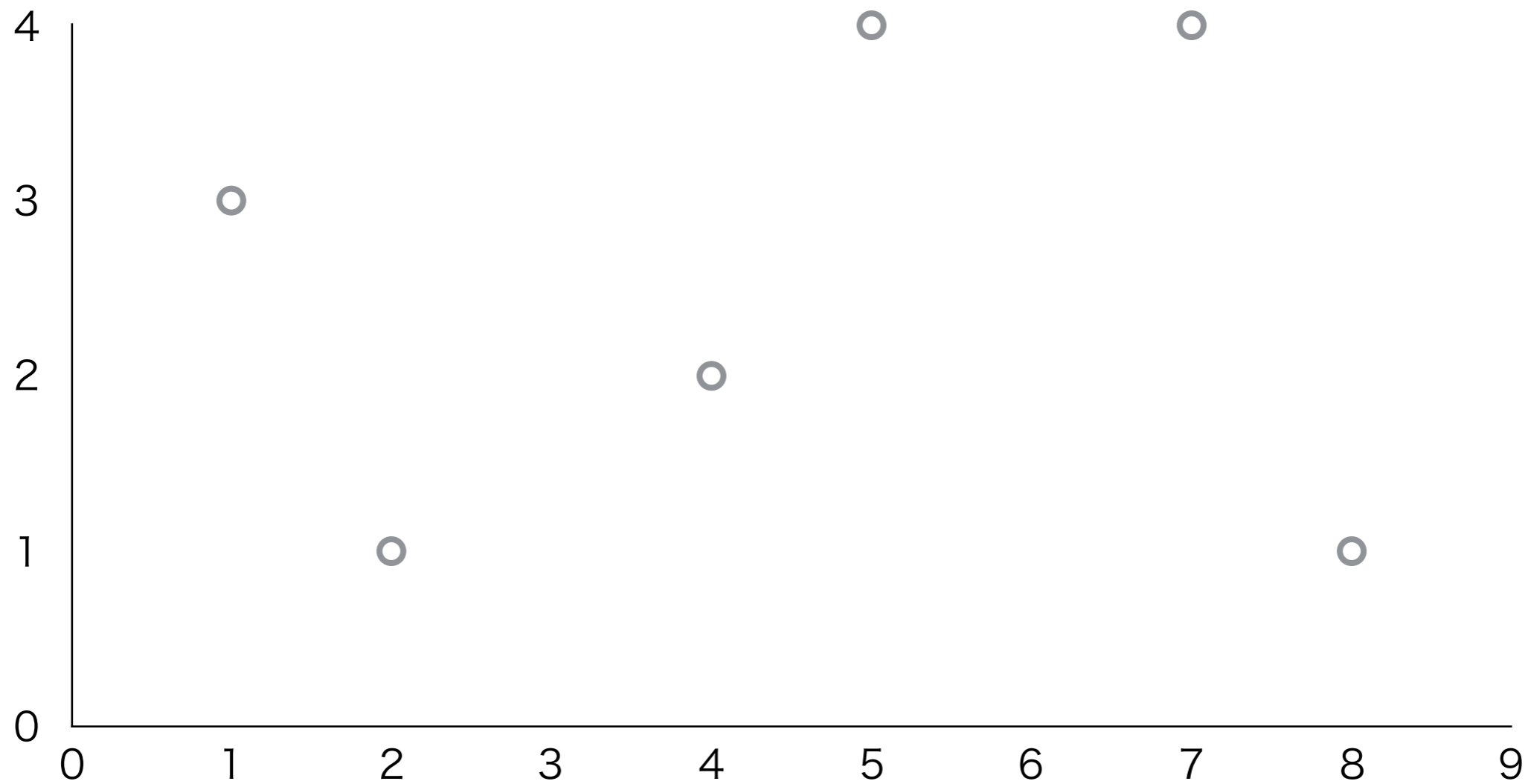
# とりあえず

- ・ 長方形をたくさん並べたり数字のペアで考えるとややこしいので、現状をより簡単に把握しましょう
- ・ それぞれのマトリョーシカ人形は 2 つのパラメータからなるものです



# とりあえず

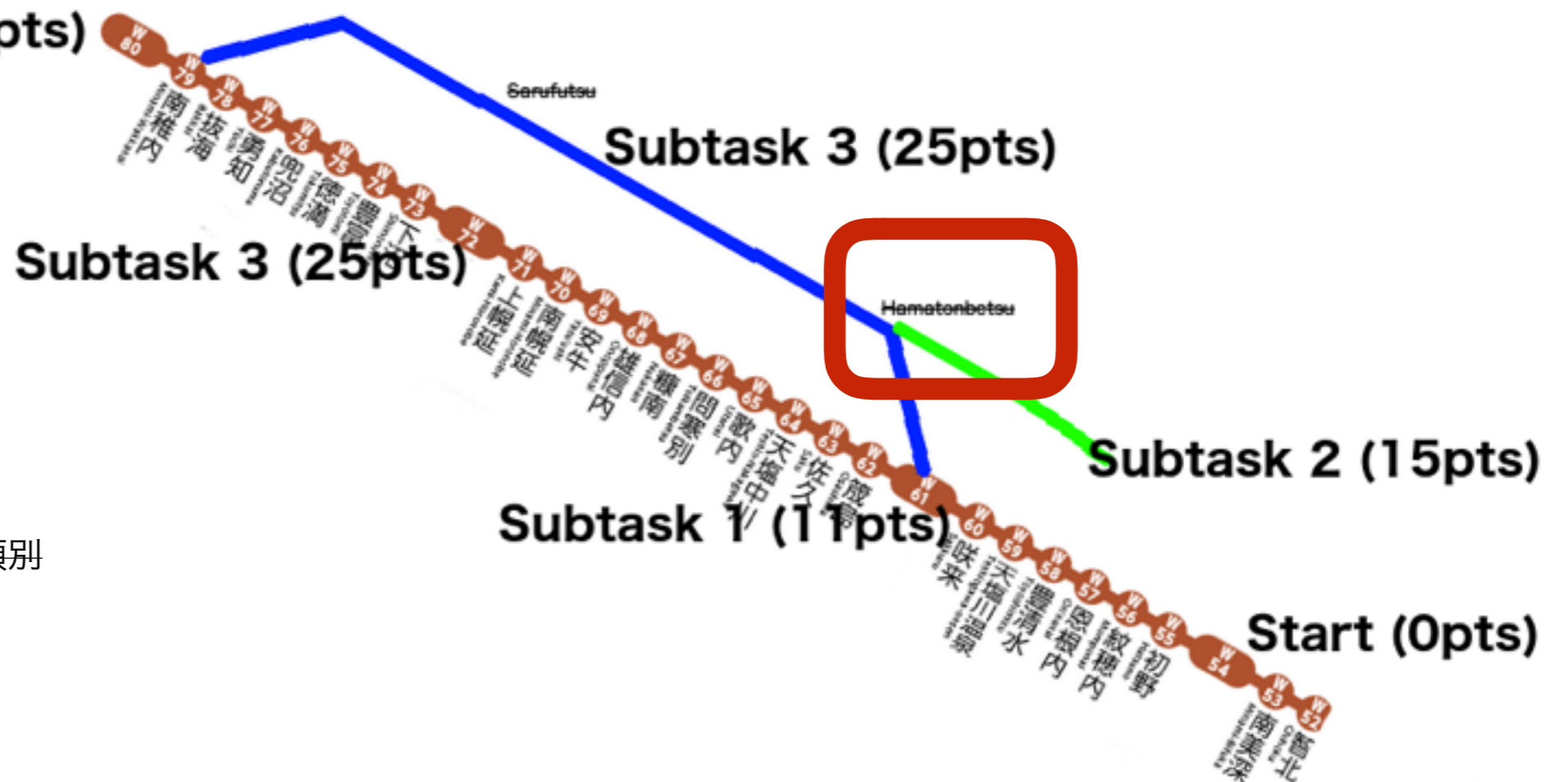
- ・ x-y 座標上の点にしてみました



# 満点解法への道のり



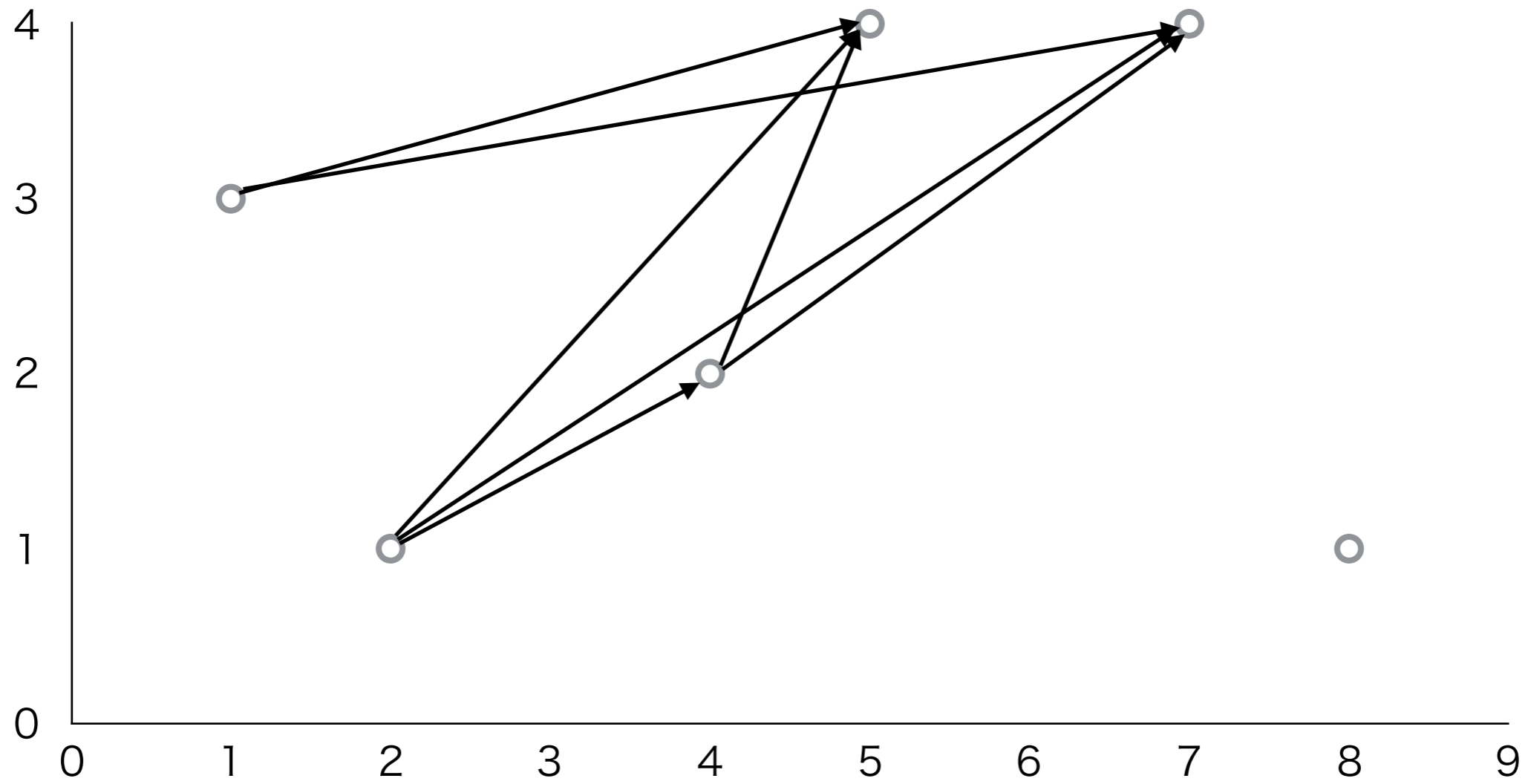
Subtask 4 (49pts)



Next: 浜頓別

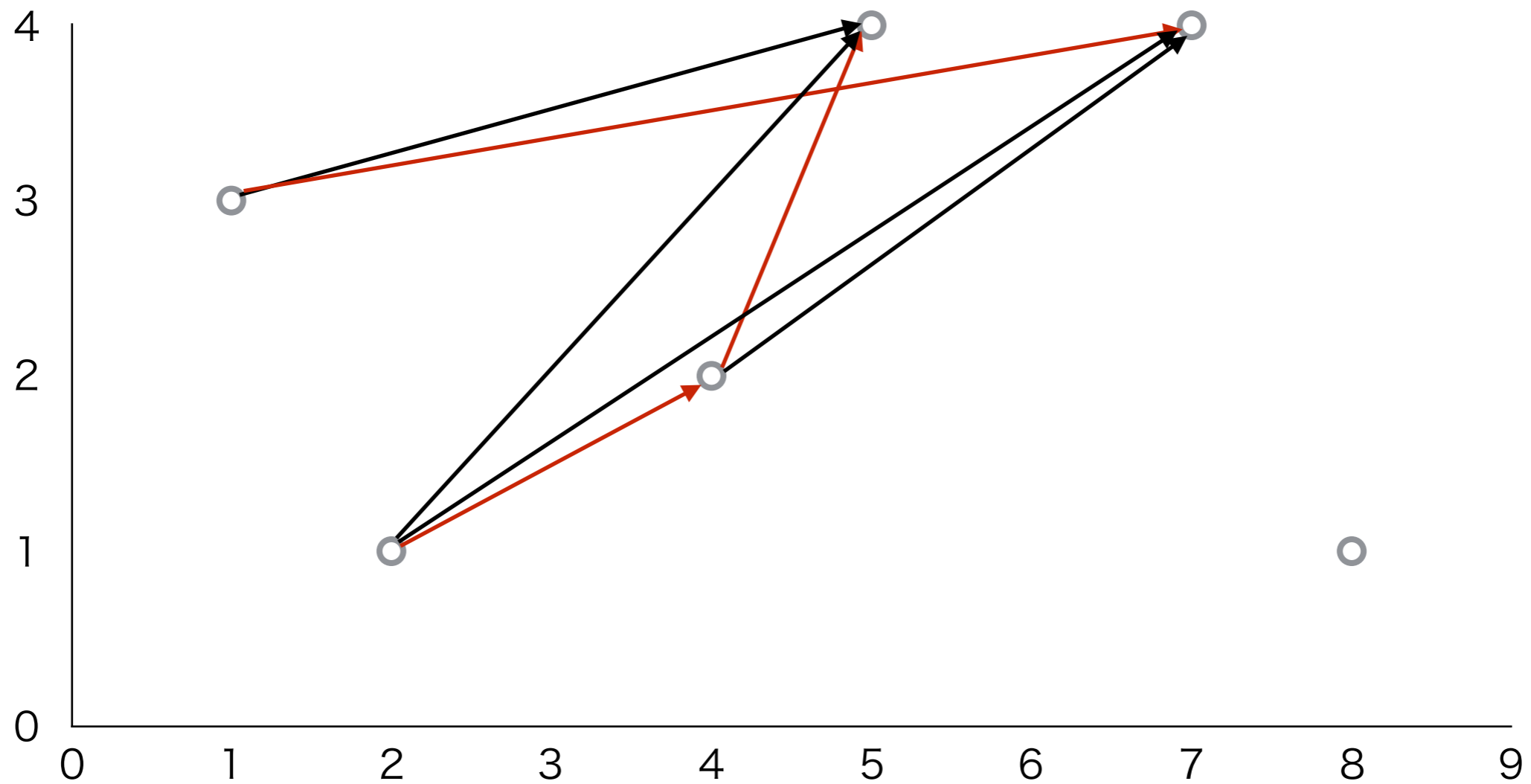
# 考察 Lv. 1

- ・ 入れ子にできるペアの関係を付け足しました



# 考察 Lv. 1

- 「例」で入れ子にした関係を赤く塗りました



# 考察 Lv.1

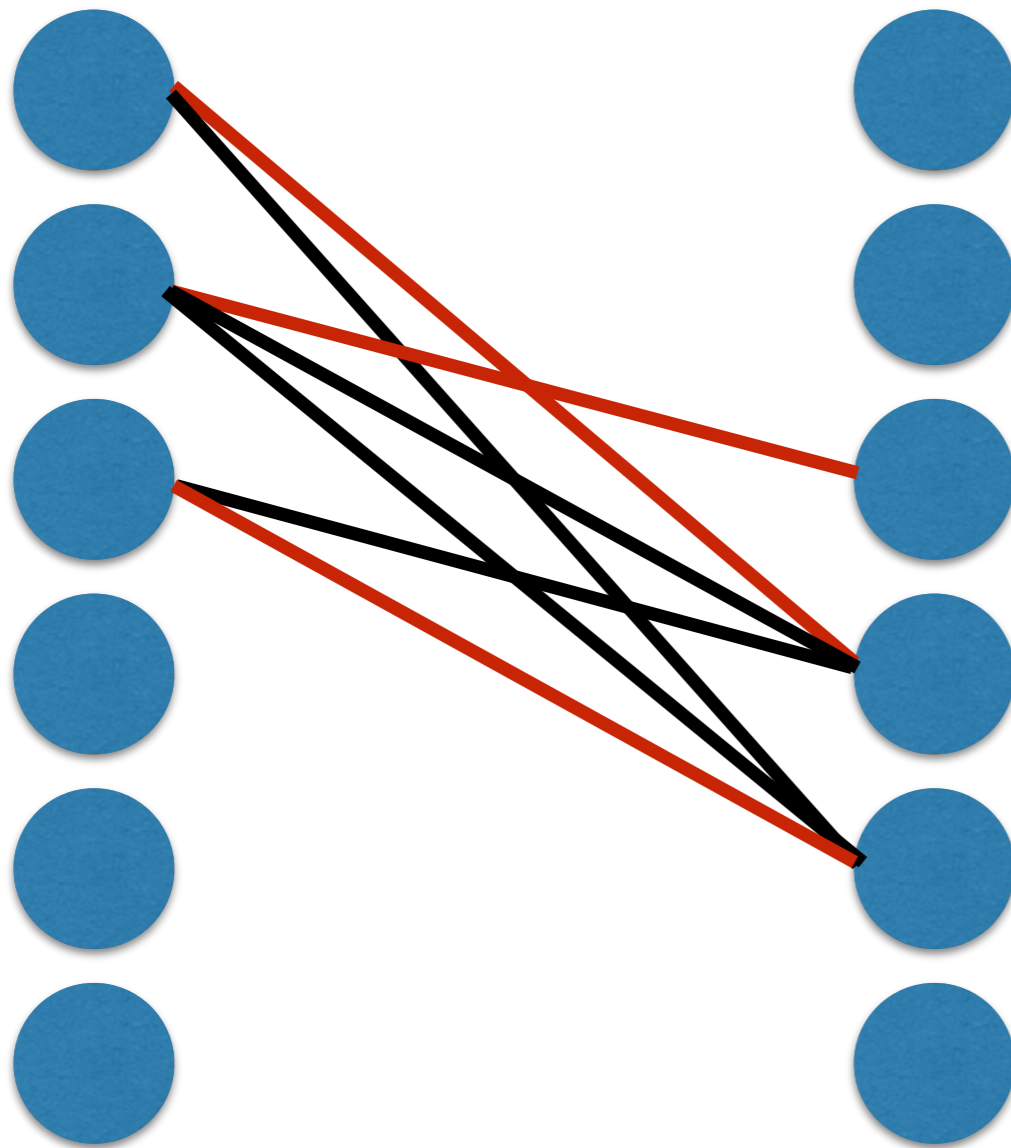
- ・ 矢印を描いてみてわかったこと
  - ・ 矢印の関係には閉路がない (= DAG )
    - ・ (矢印はどれも右上を向いているので)
- ・ 求める答えは最小いくつのパスですべての点に訪れられるかに等しい (= 最小パス被覆)



# DAG の最小パス被覆

- ・ 実は DAG の最小パス被覆は有名問題です
- ・  $N$  頂点 -  $N$  頂点の二部グラフを用意します
- ・ DAG で  $i \rightarrow j$  という辺があるなら、左の  $i$  と 右の  $j$  の間に辺を張ります
- ・ 求める答えは ( $N$  - 最大マッチング) となります

# 例



先ほどの入力例で作った二部グラフです。

最大マッチングの大きさは 3 です。

求める答えは  $6 - 3 = 3$  となります。



# フロー

- ・ 二部グラフの最大マッチングを求める必要が出てきたので、ここでフローが必要になります
- ・ 昨日の諸注意にもあったように、JOI 合宿でフローが出題される可能性もあるんですね。

# 小課題 2

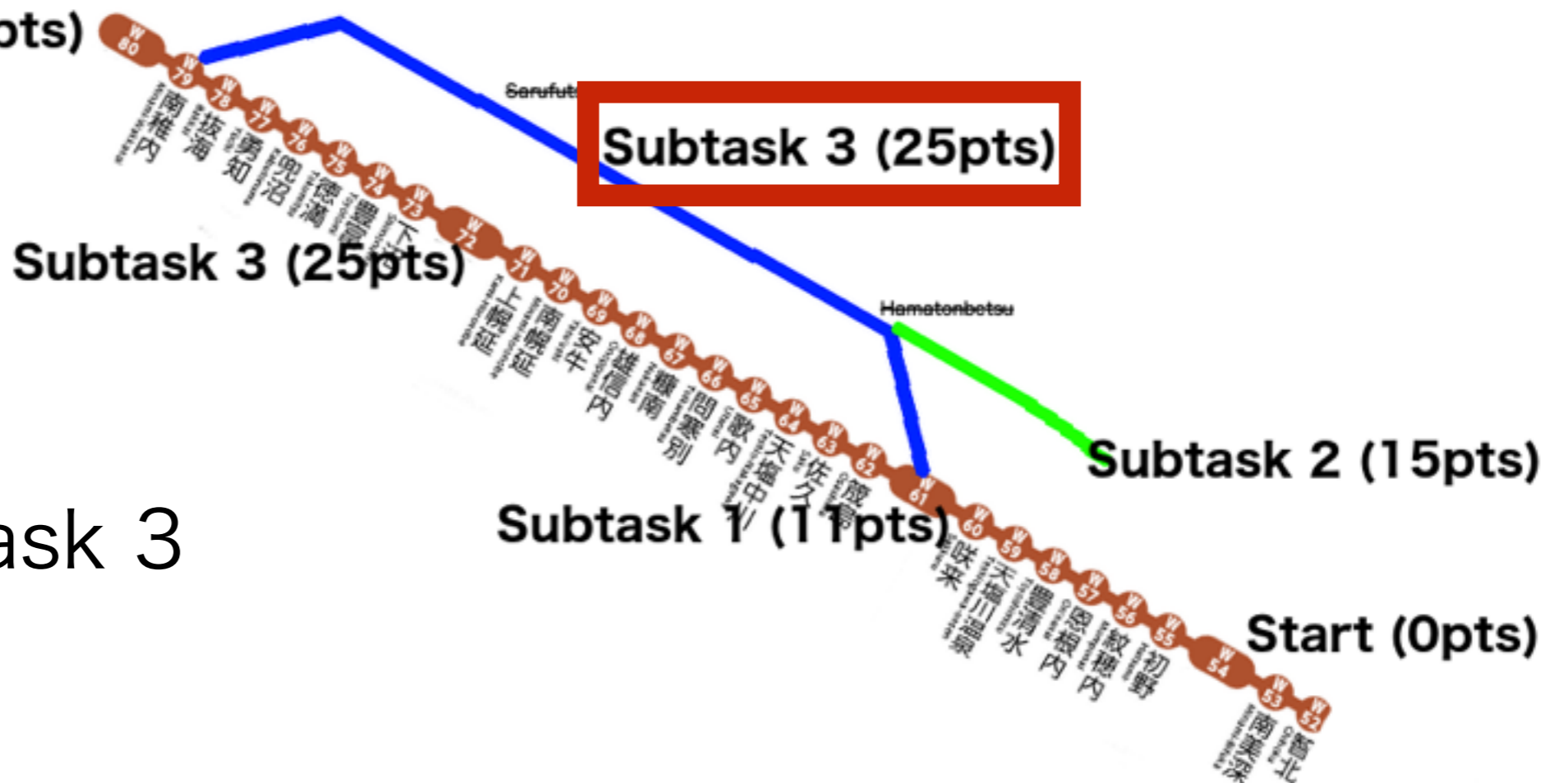
- ・ DAG の 最小パス被覆をフローで求めると、頂点数  $O(N)$ 、辺数  $O(N^2)$  となるので、計算量は  $O(N^3)$
- ・ 小課題 2 が解けます (15点)

# 満点解法への道のり



.....

Subtask 4 (49pts)



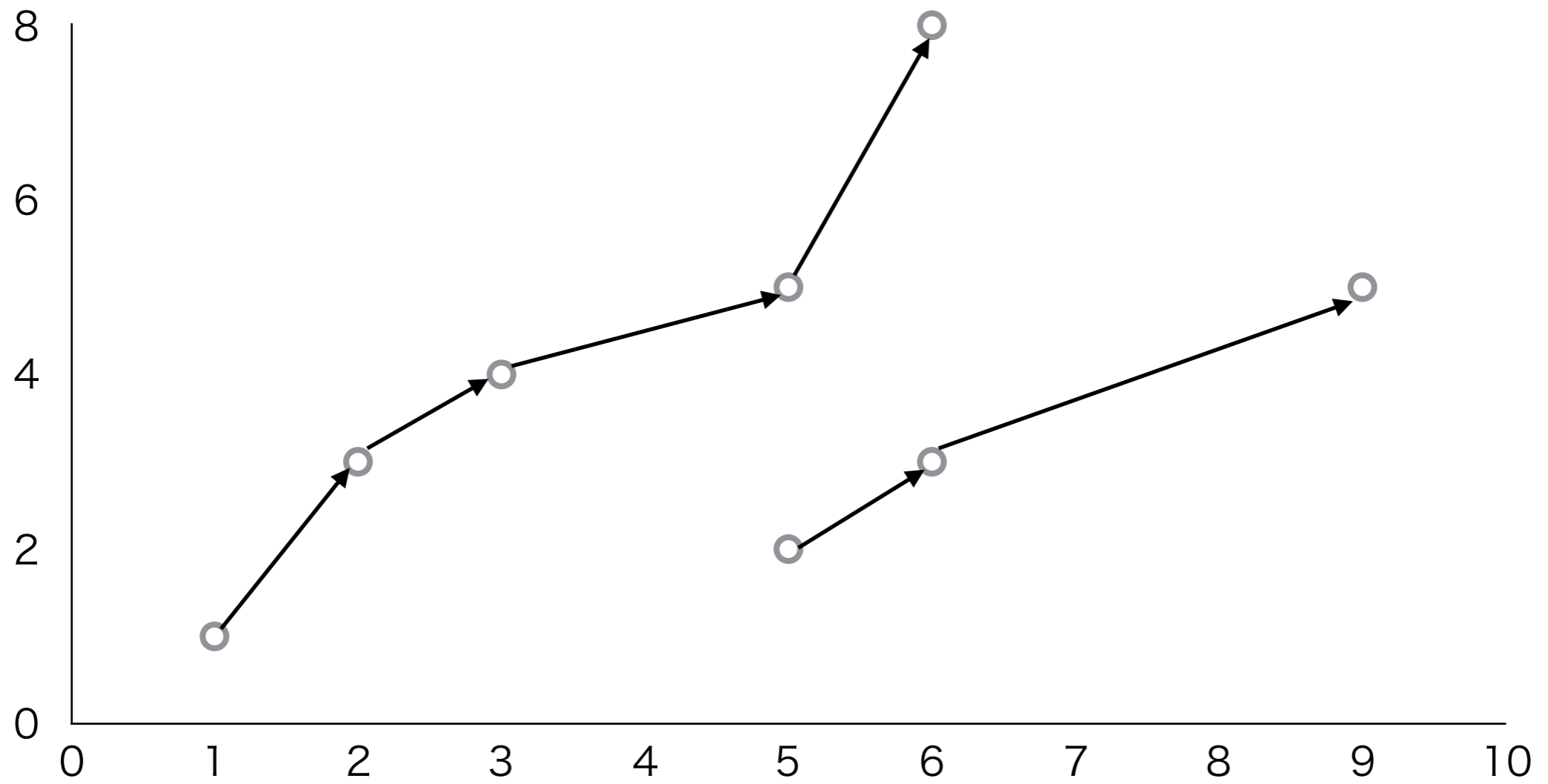
Next: Subtask 3

# 考察 Lv.2

- ・ パス被覆の個数がどうなるかをいろいろ点を描いて試してみましよう

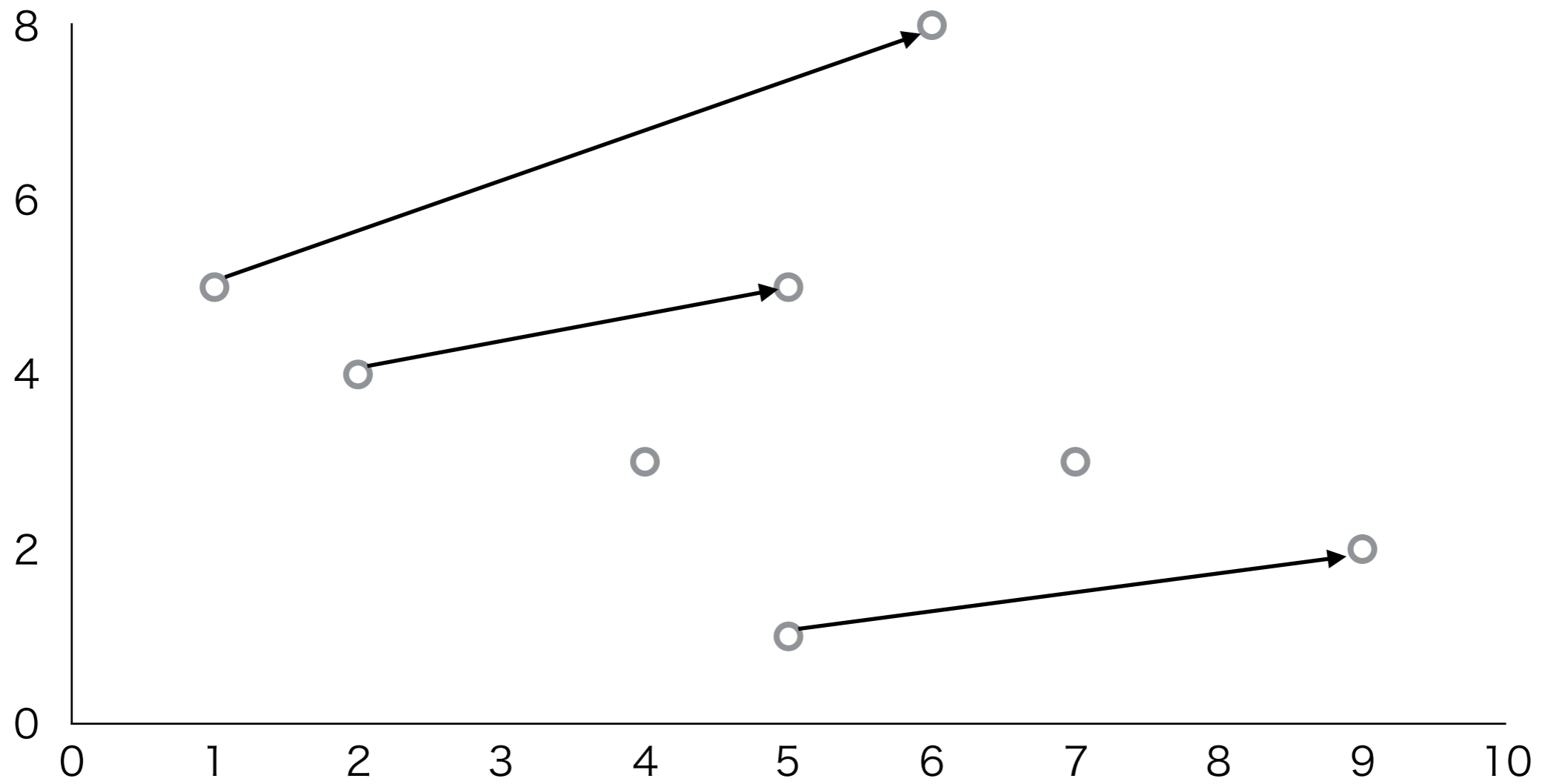
# 考察 Lv. 2

・なるほど



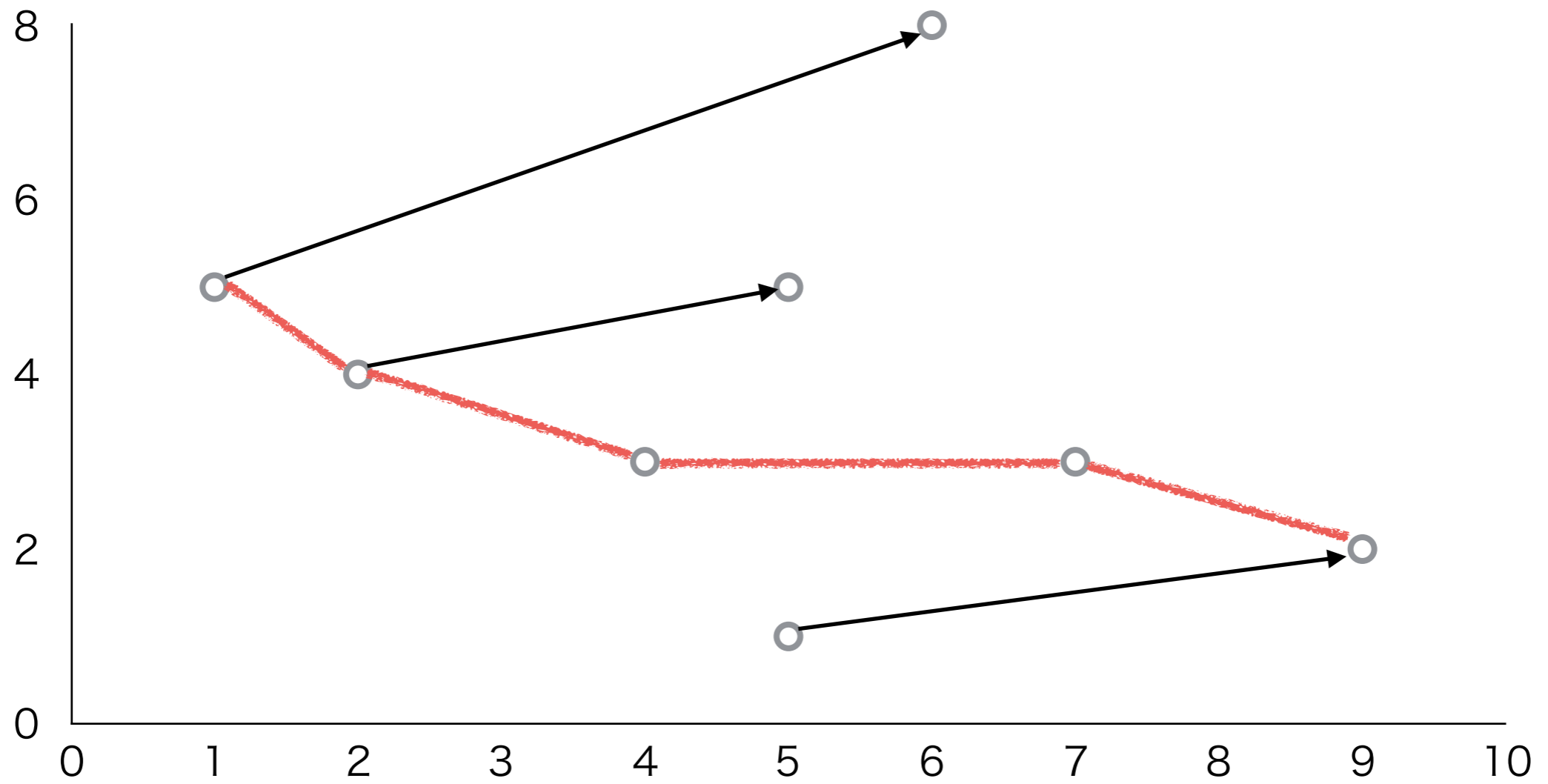
# 考察 Lv. 2

- ・ なんか落書きしてみようかな



# 考察 Lv. 2

- ・ 意味ありげな線が引けたぞ! ?



# 思ったこととまとめ

- ・ 右下から左上に向かって線を引くと、その線上の点はそれぞれ別のパスで被覆されるので、その折れ線上にある点の個数は最低でも必要っぽい
- ・ これの最大個数より多くのパスが必要な場合はあるかな……なさそうだな……それ本当かな……？



# 思ったこととまとめ

それ、正しいです。

・  
・  
こ  
かな…

無  
ある

?

# Dilworth の定理

- $i \rightarrow j, j \rightarrow k$  の辺がある時、必ず  $i \rightarrow k$  の辺もあるような DAG に対し、最小パス被覆の本数は、任意の 2 点の組に対し辺が存在しないような頂点集合の最大サイズに等しい



Robert P. Dilworth  
(1950)

# 問題を変換

- ・ この定理を使うと、次のような問題となります
- ・ 座標上の点で、右下方向 (右、下も含む)に進んでなるべくたくさん点を通りたい。最大で何個の点を通ることができるか？
- ・ 要は                      のマイナーチェンジ

# 問題を変換

- ・ この定理を使うと、次のような問題となります
- ・ 座標上の点で、右下方向 (右、下も含む)に進んでなるべくたくさん点を通りたい。最大で何個の点を通ることができるか？
- ・ 要は **LIS** のマイナーチェンジ



# 小課題3 (1) まとめ

- ・ それぞれのクエリで以下のことを行う
- ・  $R_i$ が小さい順 ( $R_i$ が等しいときは $H_i$ が大きい順)にソート
- ・ ソートされた列においての $H_i$ のみに着目し、最長非増加部分列問題を解く ( $O(N \log N)$  で解ける)
- ・ 全体で  $O(QN \log N)$  なので、小課題 3 が解けます

# 定理知らないんだけど？

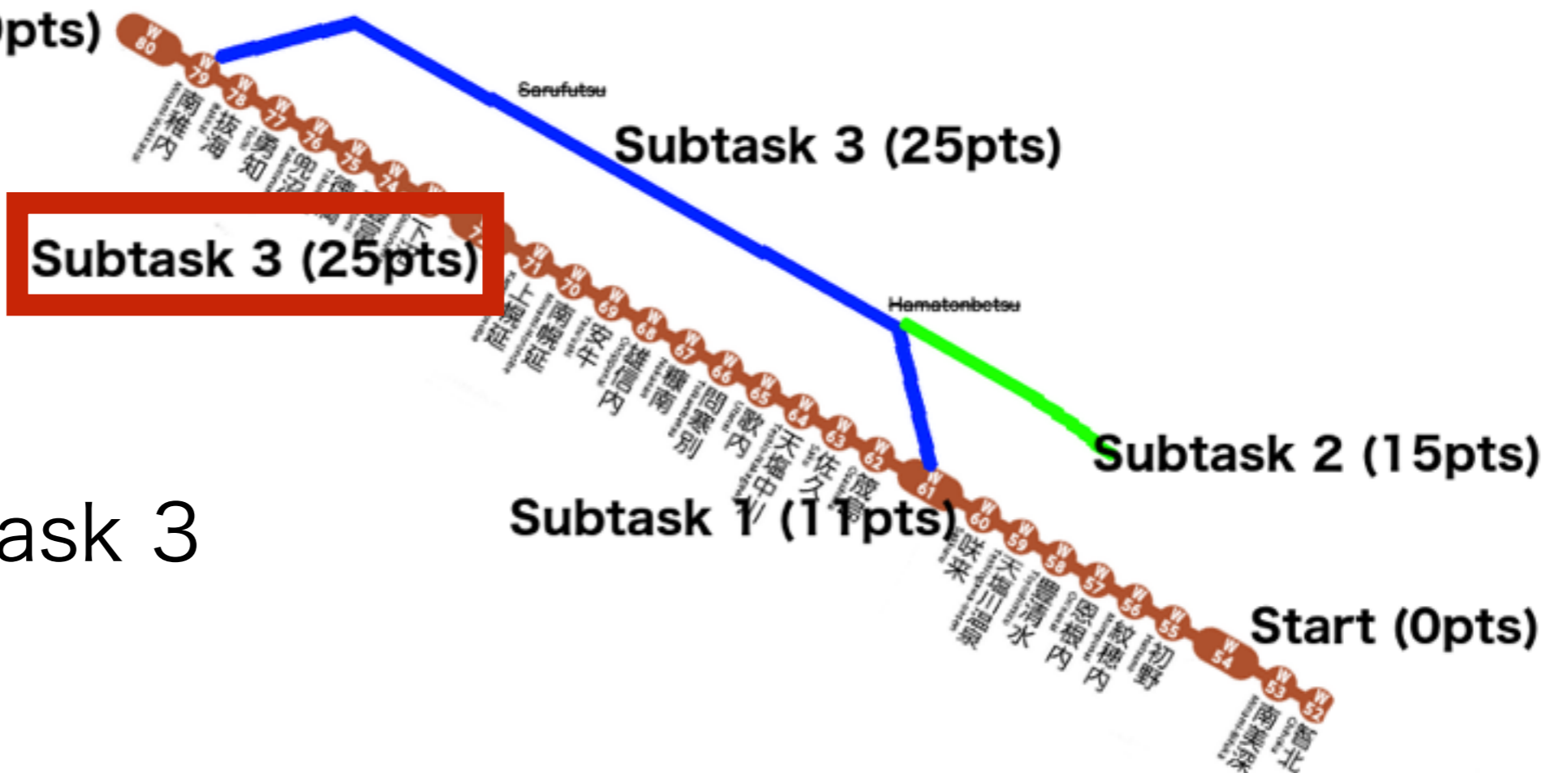
- ・ 完全フィードバックなのでとりあえず正しいと思って一度書いてもそんなに損はしない
- ・ けど
- ・ 知らなくてもちゃんと正しいと確信できる解法はないの……？

# 満点解法への道のり



.....

Subtask 4 (49pts)



Next: Subtask 3

# 小課題3 別ルート

- ・ さっきと同じ順( $R_i$  の昇順、同率は  $H_i$  の降順)で並べた状況を考えます
- ・ 並べた順に左から 1 つずつ追加していくことにします

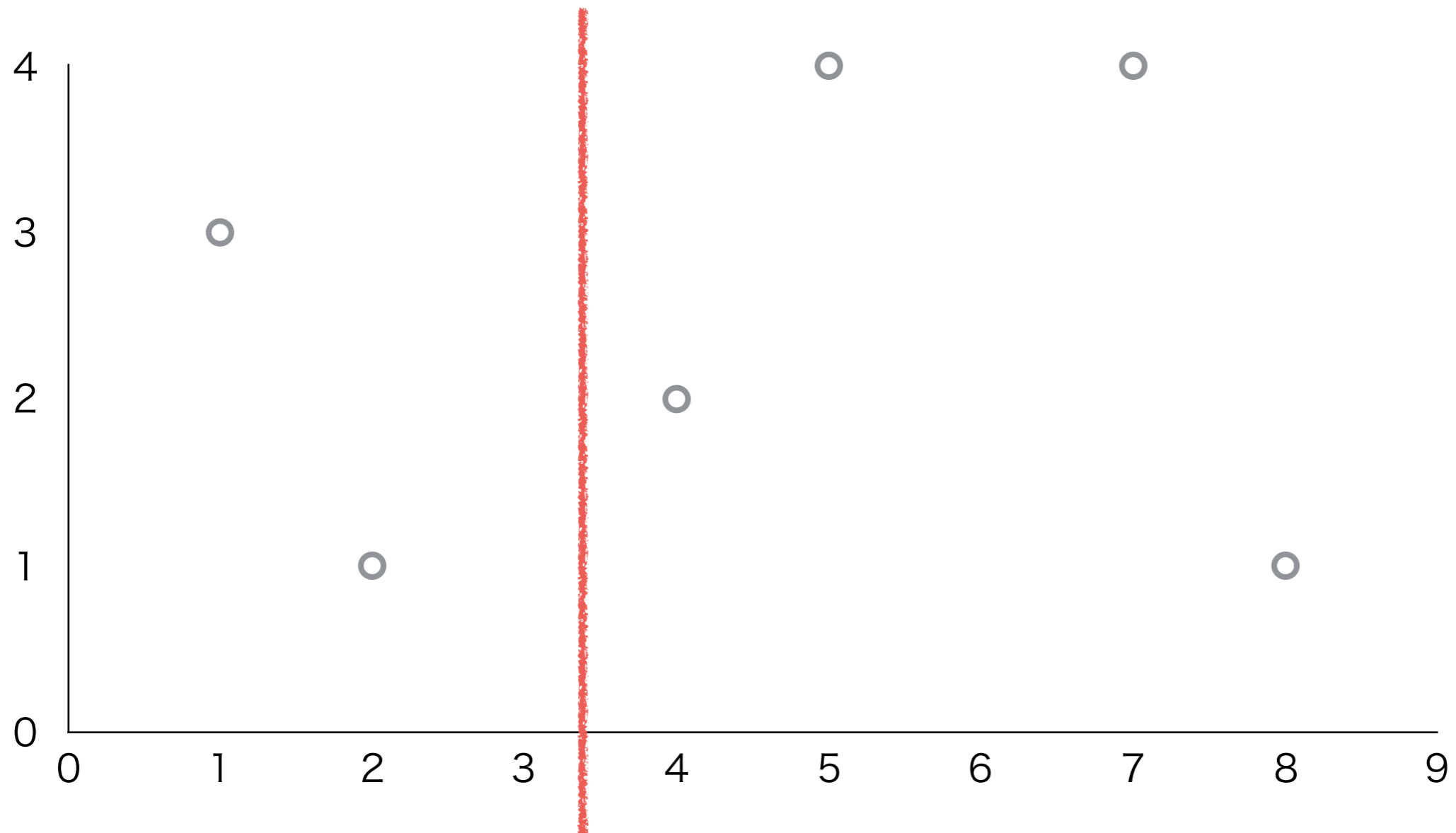


# 小課題3 別ルート

- ・ あるものを追加するとき、それに入る小さいものが今入れ子にされず残っていたらどうすればいいでしょう？
- ・ 入れない → 1 つ余計に増えるだけ、損
- ・  $H_i$  の小さいものを入れる → あとで損しそう
- ・  $H_i$  の大きいものを入れる → 損要素なし

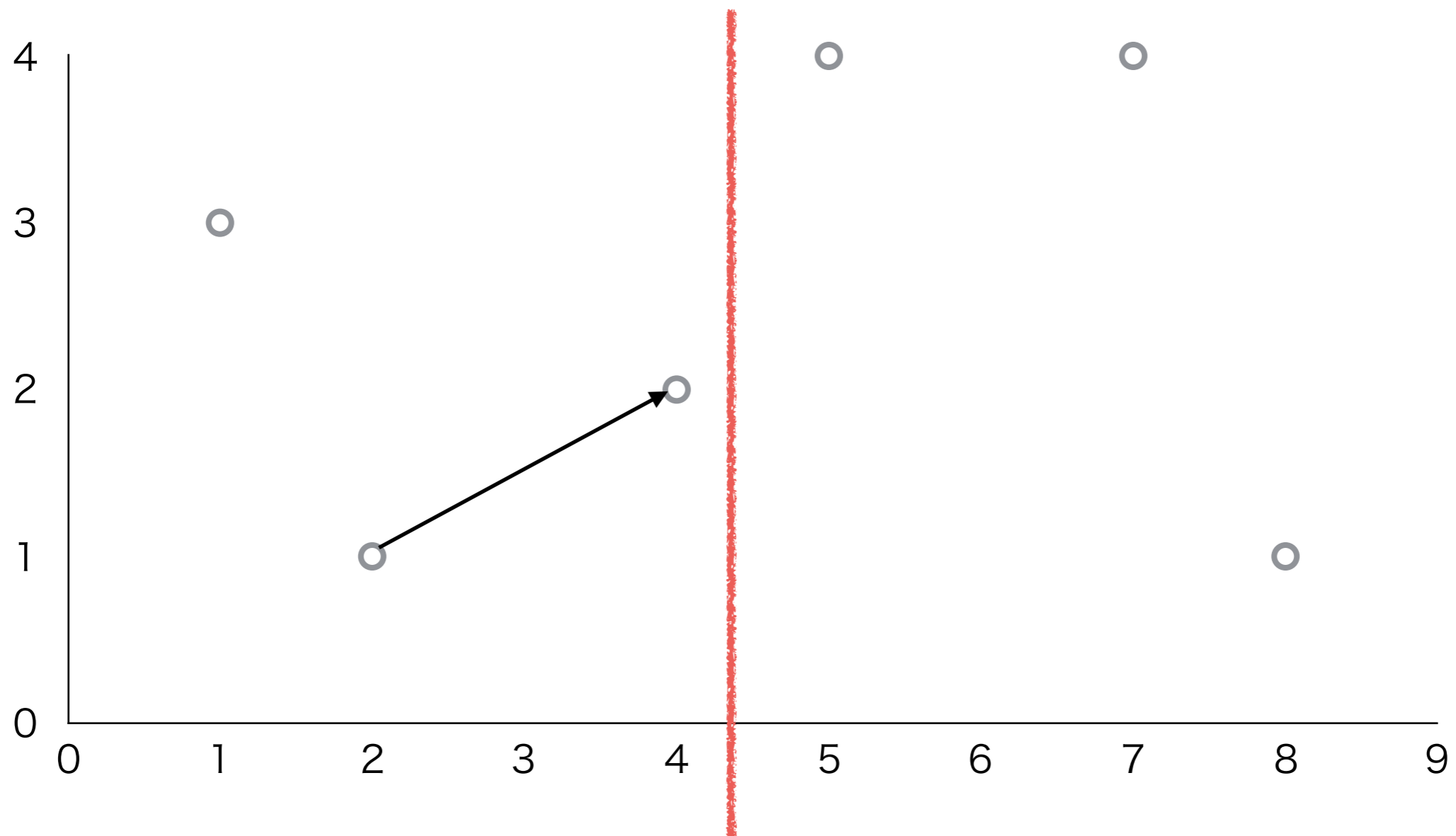
# 小課題3 例

- ・ ここまででは入れ子にできません



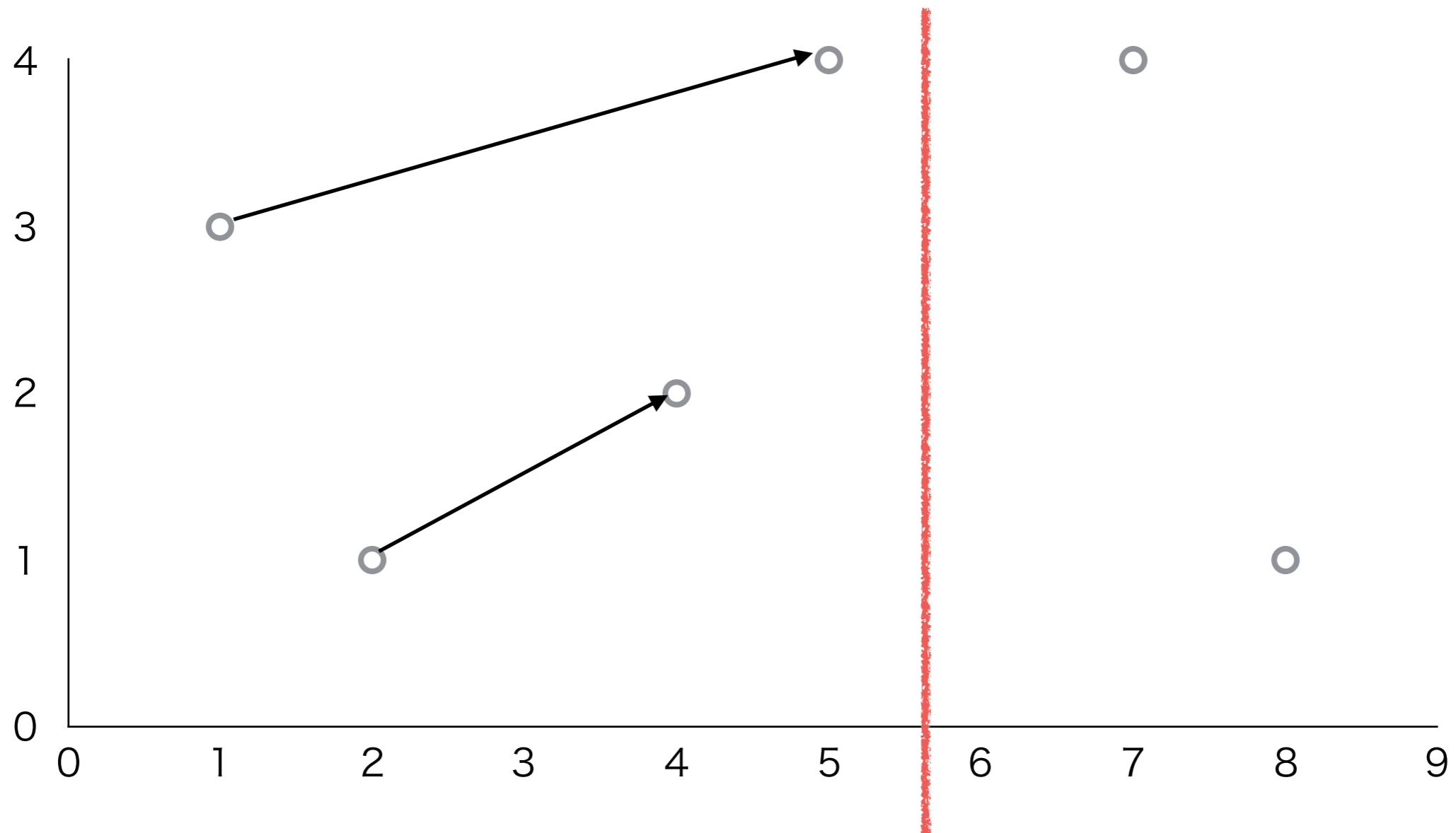
# 小課題3 例

- ・ 1つ入れられるものがあるので入れます



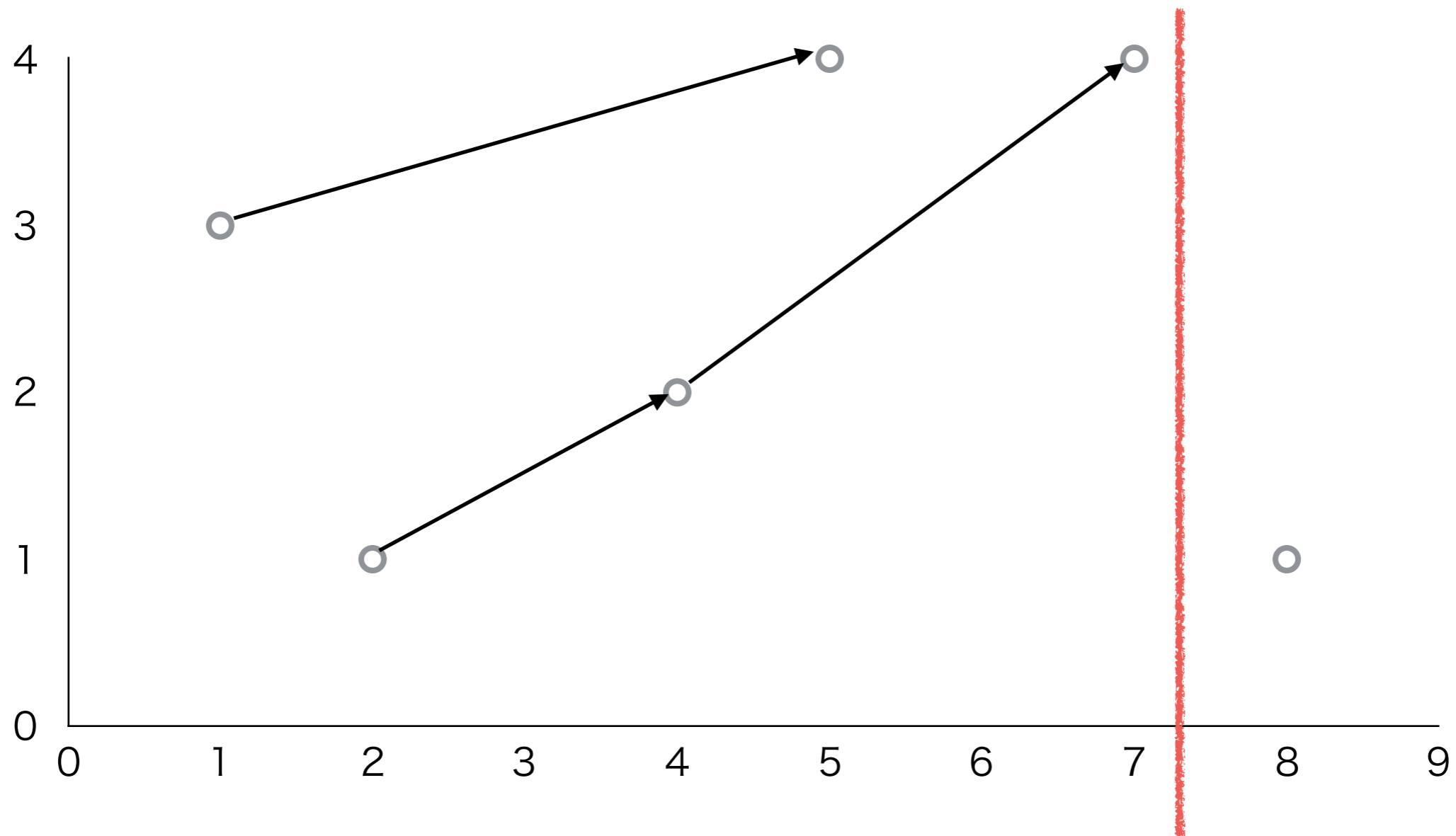
# 小課題3 例

- 2つ以上あるので、 $H_i$ の大きい方を入れます



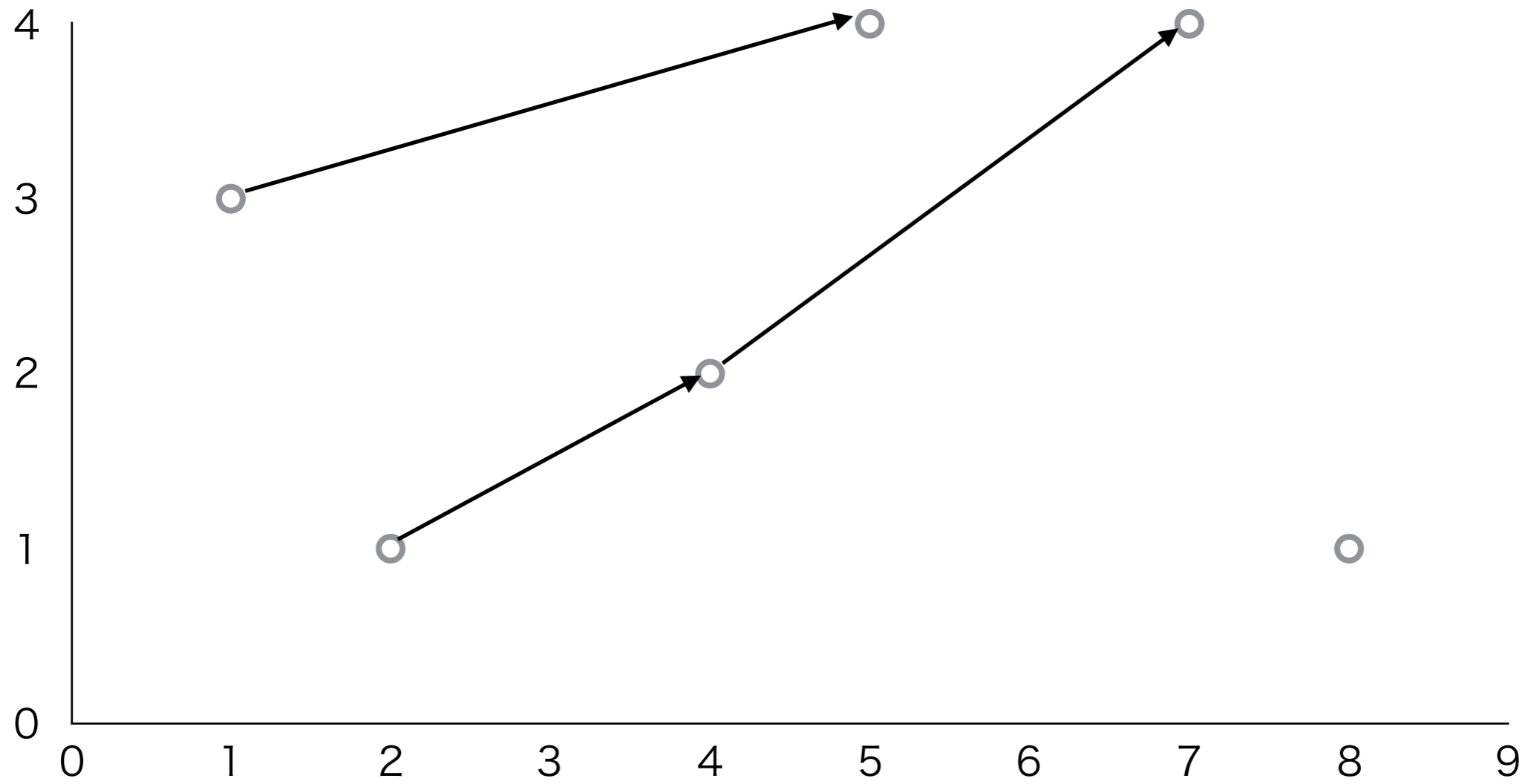
# 小課題3 例

- ・ 1つ入れられるものがあるので入れます



# 小課題3 例

- ・ 答えは 3 です



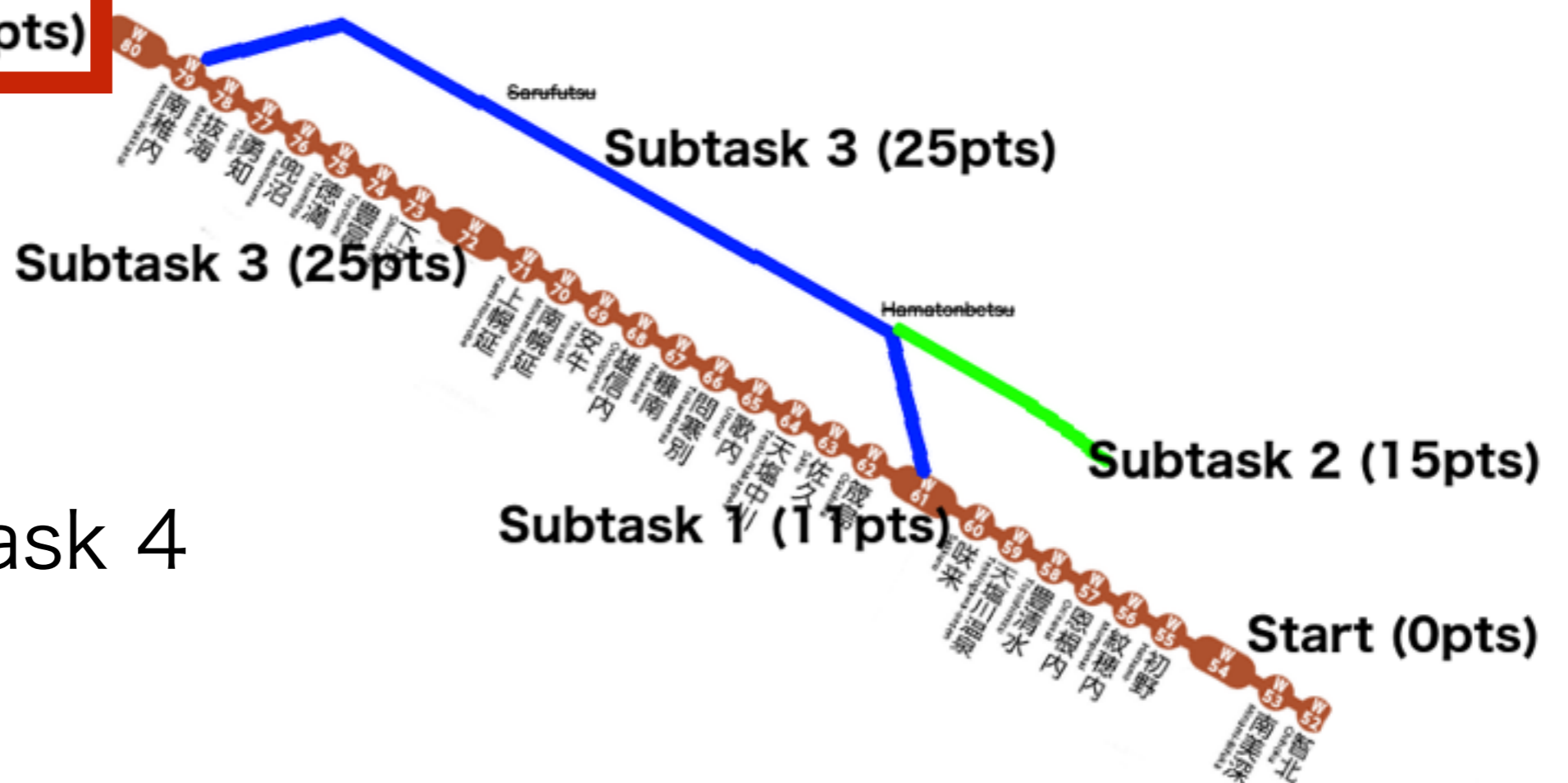
# 小課題3 (2) まとめ

- ・ 各クエリに対して以下のように求めていく
- ・ 先ほどと同じようにソートしてからGreedy
- ・ ソート列の左から追加していく
- ・ あるものを追加する時、何か入れられるものが残っていれば  $H_i$  が入れられるうち最大のものを入れる
- ・ これは `std::set` 等で実装できて、  $O(N \log N)$
- ・ 合計で  $O(QN \log N)$  で小課題 3 が解ける

# 満点解法への道のり



Subtask 4 (49pts)



Next: Subtask 4

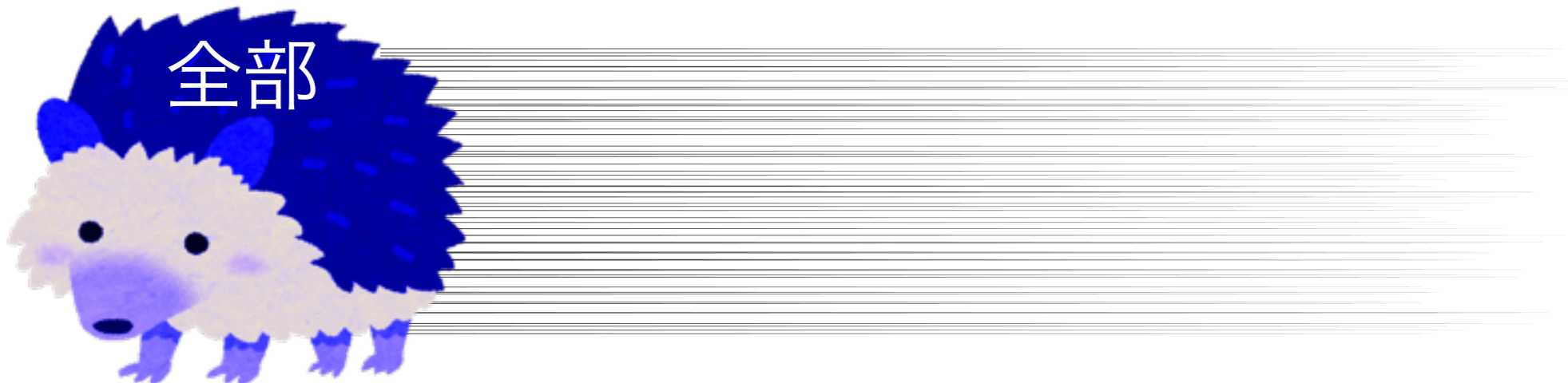


# 高速化

- 小課題 3



- 小課題 4

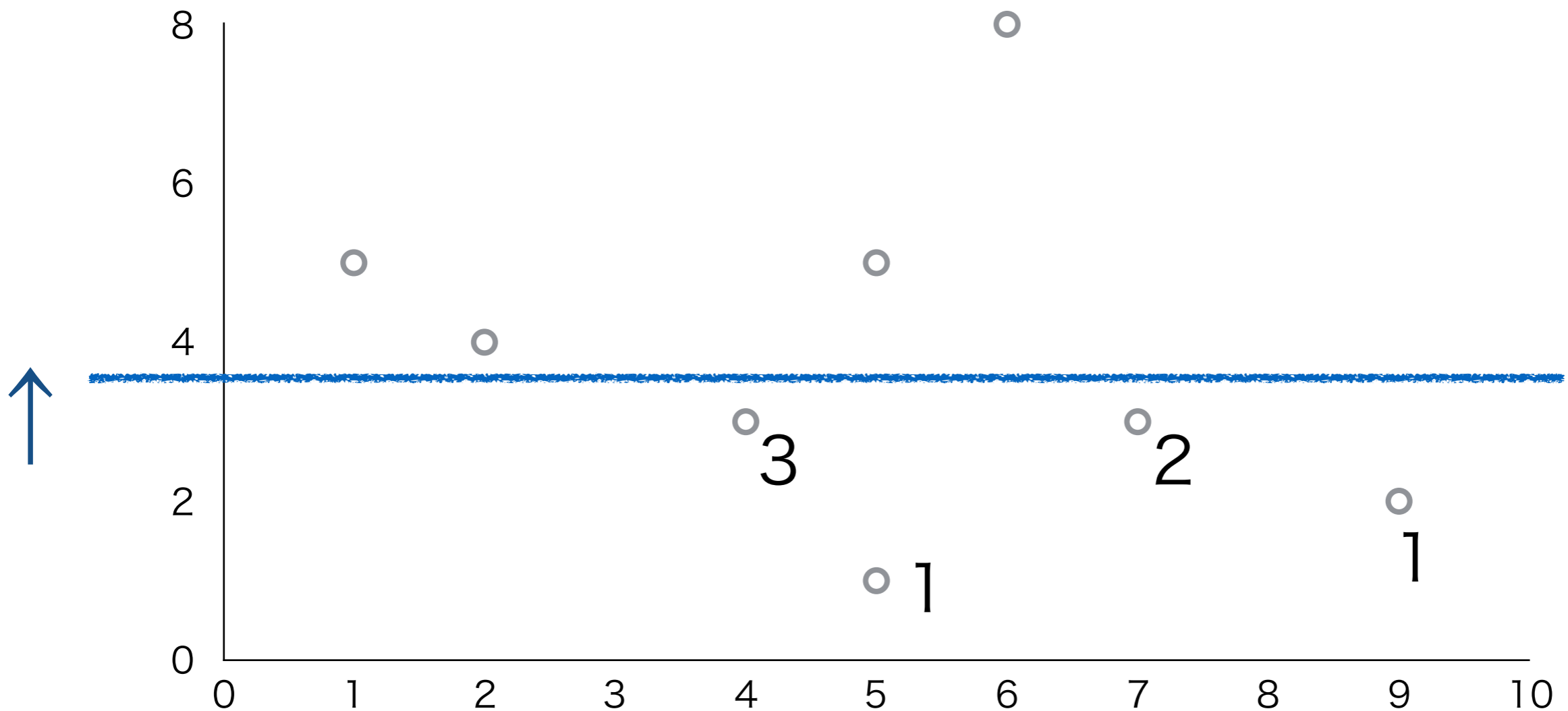


# クエリをまとめる

- ・ まずは LIS 版をまとめて求めてみましょう
- ・ クエリは  $y$  座標が小さい順にソートしておきます

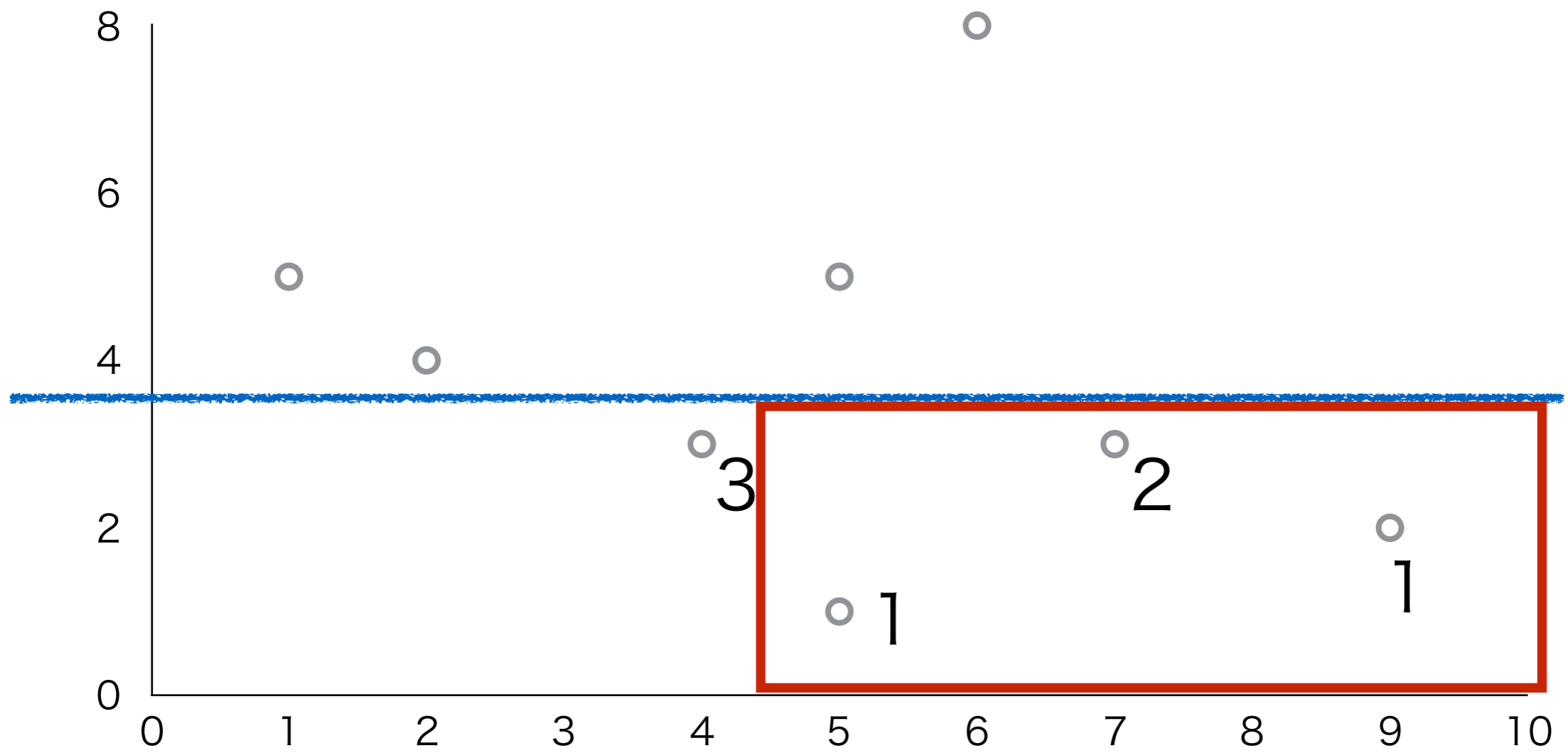
# クエリ処理の例

- 各頂点の数字はそれを左端とする LIS 最大長です



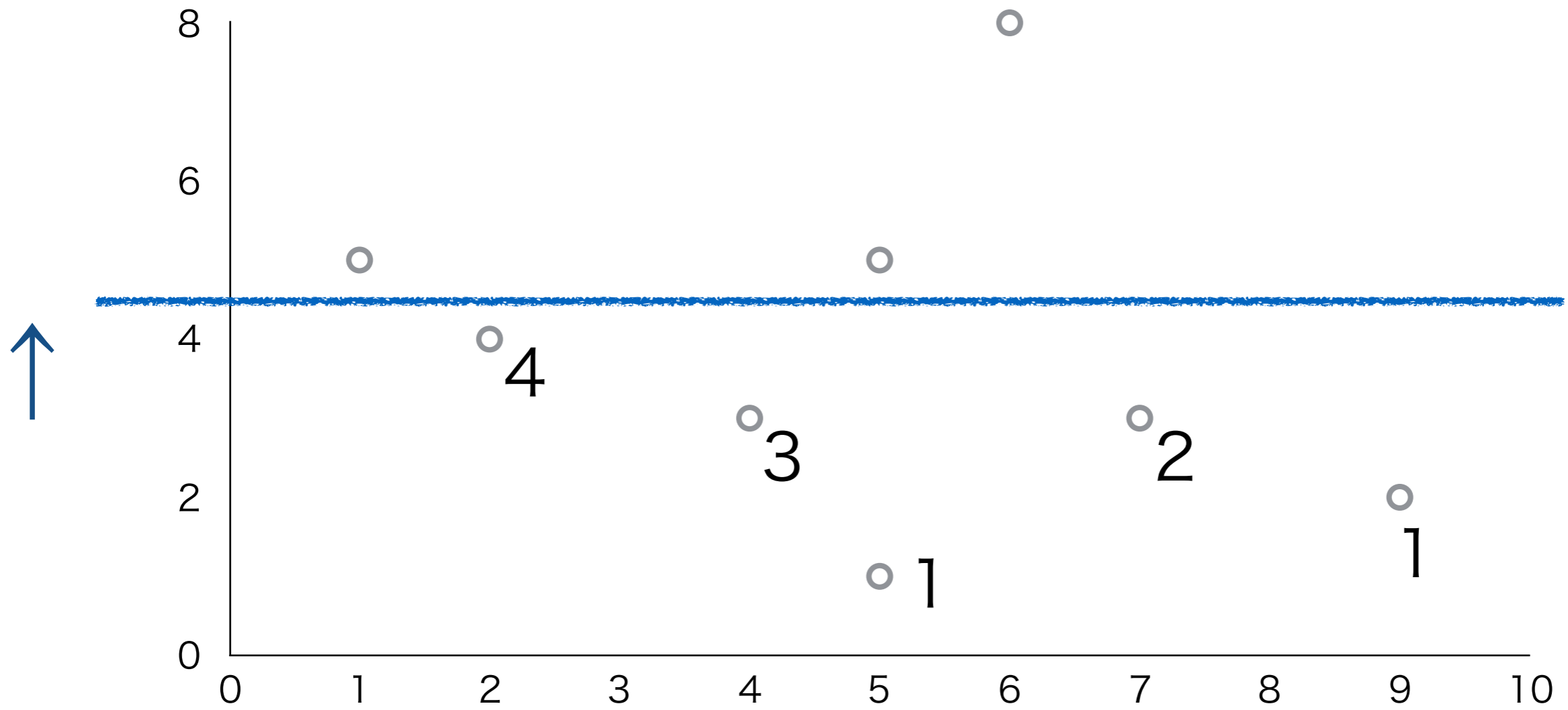
# クエリ処理の例

- 赤い範囲にクエリが来たら 2 が答えです



# クエリ処理の例

- ・ 走査線を上に動かして、新しい点を追加します



# LIS 高速化 まとめ

- ・ クエリと点の  $y$  座標が小さい順 (同じ時は点の追加が先、点は  $x$  座標が大きい順) にソートしておく
- ・ ある範囲の値の最大値を求める
- ・ ある場所を更新する
- ・ という 2 種類ができれば良い

# LIS 高速化 まとめ

- ・ クエリと点の  $y$  座標が小さい順 (同じ時は点の追加が先、点は  $x$  座標が大きい順) にソートしておく
- ・ ある範囲の値の最大値を求める
- ・ ある場所を更新する
- ・ という 2 種類ができれば良い → **座圧 Segtree**



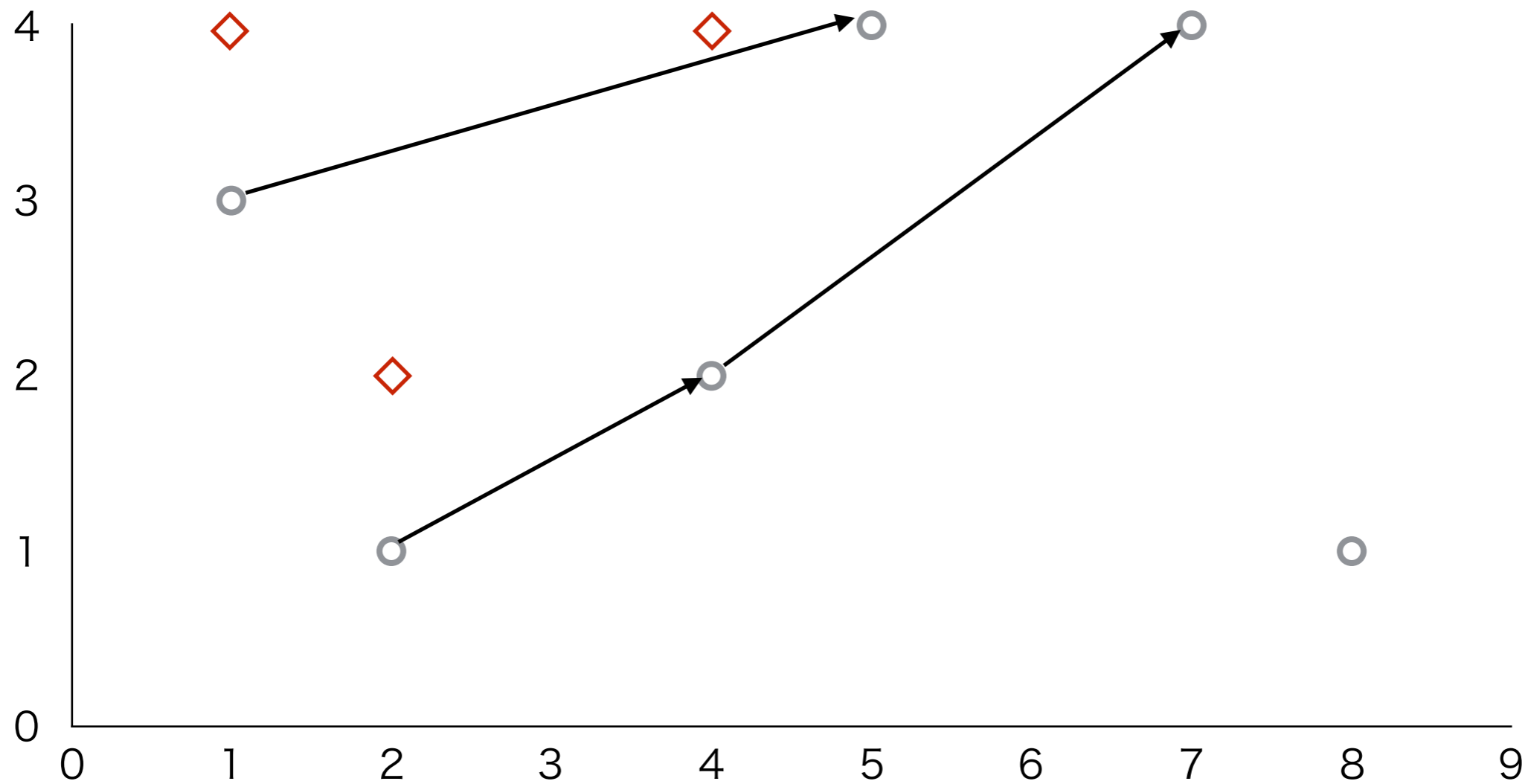
# クエリをまとめる

- ・ 次に Greedy 版をまとめて求めてみましょう
- ・ 先ほどの Greedy で入れ子になる組を事前に求めて、( $R_i$ の小さい方,  $H_i$ の大きい方) に重み  $-1$  の点を追加します
- ・ 今度は  $x$  座標が**大きい**順に走査していきます



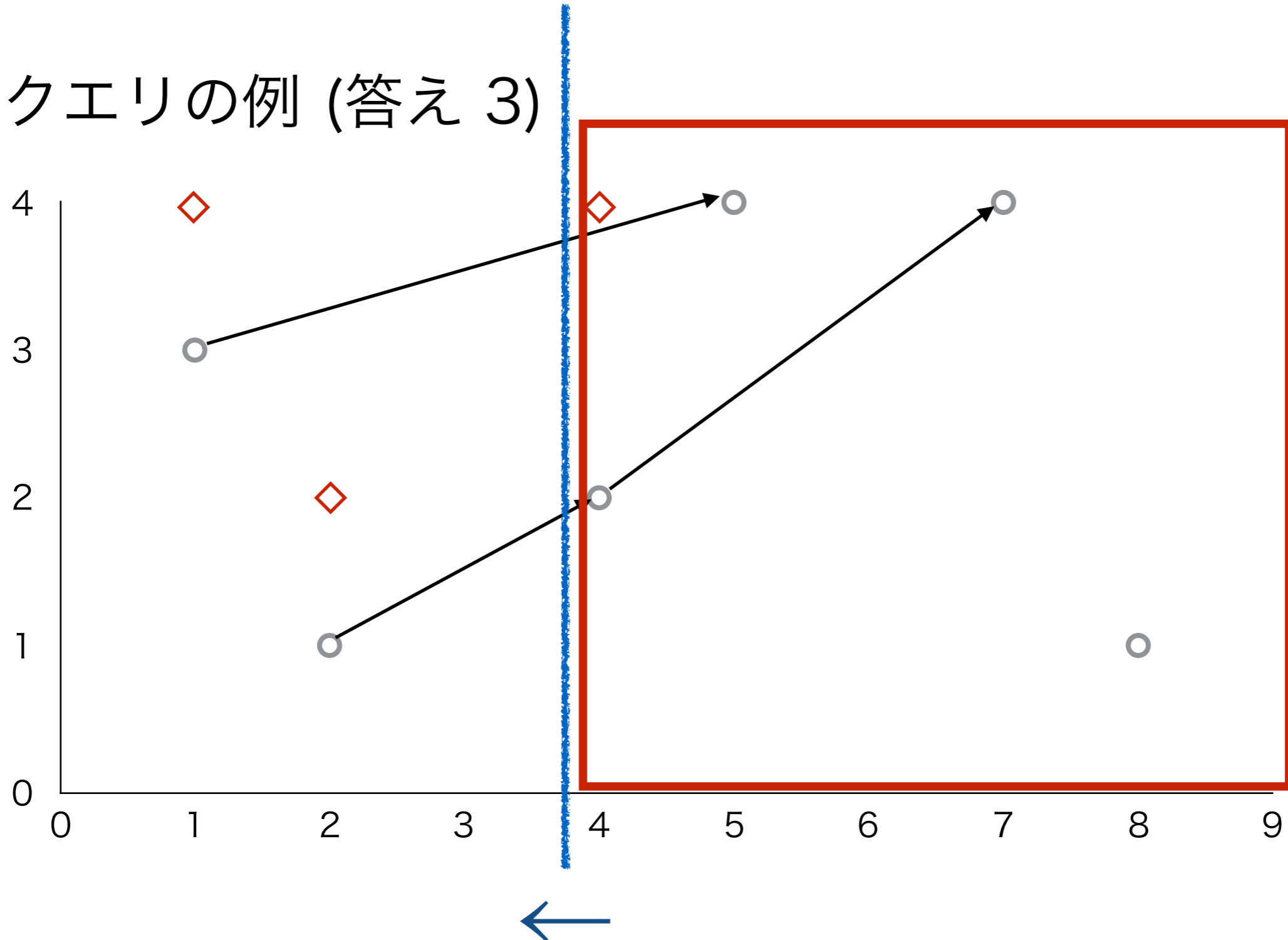
# 小課題4 例

- ・ 平面の例 (赤い点が  $-1$  の重み)



# 小課題4 例

- クエリの例 (答え 3)



# Greedy 高速化 まとめ

- ・ クエリと点の  $x$  座標が大きい順 (同じ時は点の追加が先) にソートしておく
- ・ ある範囲の値の合計を求める
- ・ ある場所に値を足す
- ・ という 2 種類ができれば良い →

# Greedy 高速化 まとめ

- ・ クエリと点の  $x$  座標が大きい順 (同じ時は点の追加が先) にソートしておく
- ・ ある範囲の値の合計を求める
- ・ ある場所に値を足す
- ・ という 2 種類ができれば良い → **座圧 BIT**



# 上手な解法

- ・ 実は LIS 解法で LIS に特有な  $O(N \log N)$  解法 (二分探索するやつ) を使うと、`std::upper_bound` だけで満点を取ることができます

クエリはソート  
します



# 得点分布

