オレンジの出荷

wafrelka

もくじ。

- 問題概要
- 簡単な解法 (方針/実装/解析/評価)
- 高速化への道 (発想/手法)
- 高速な解法 (実装/解析)
- 得点分布

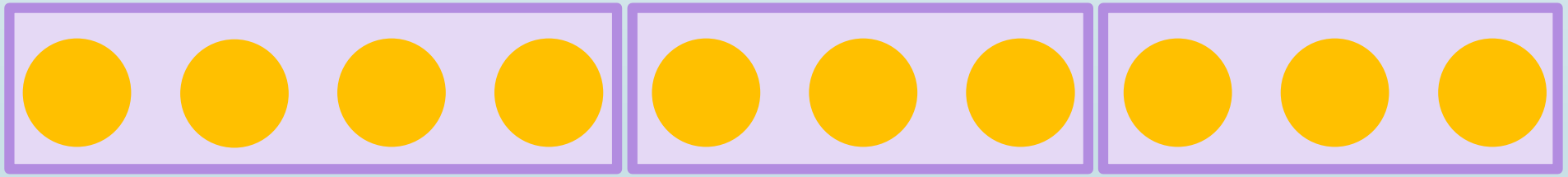
コンテストお疲れ様でした

03/15

ここに
ジューシーなオレンジの画像

問題概要

- オレンジが1列に並んでいます。
- 適当な位置で区切りを入れて箱に詰めます。
- 詰め方によってコストが決まるので
  コストを最小化してください。
- オレンジの個数: N, 箱に入る最大のオレンジ: M



簡単な解法 (方針)

- 考えられる詰め方を列挙してコストを計算。
- 最小値がそのまま答え。

簡単な解法 (実装)

$dfs(a) :=$ a 個目以降を詰めるときの最小コスト

$$dfs(a) = \min\{\, dfs(b) + (b-a) \times (MAX - MIN) + K \,\}$$

$$(a < b \leq a + M)$$

簡単な解法 (解析)

- $dfs(a)$ の呼び出しは全体で $O(2^N)$ 回
- 1回の $dfs(a)$ で $O(M^2)$ 回のループ
- 時間計算量 $O(2^N\ M^2)$

- MAX, MIN を効率よく求めると $O(2^N\ M)$

簡単な解法 (評価)

- 小課題 1 が解ける ($N \leq 20$)
- 小課題 2 以降は無理そう ($N \leq 1000$)

高速化への道 (発想)

- $dfs(a)$ を計算するときに
  $dfs(a+1)$, $dfs(a+2)$, … を計算している。

- $dfs(a+1)$ を計算するときに
  $dfs(a+2)$, $dfs(a+3)$, … を計算している。

- $dfs(a+2)$ が複数回呼ばれている。

高速化への道 (手法)

- $dfs(x)$ の結果をどこかに記録しておけば 2 回目以降の計算は一瞬で終わる。

- メモ化 (memoization) というテクニック。
- 分割統治法 + メモ化
- 動的計画法 (DP) と呼ばれる手法の一種。

高速な解法 (実装)

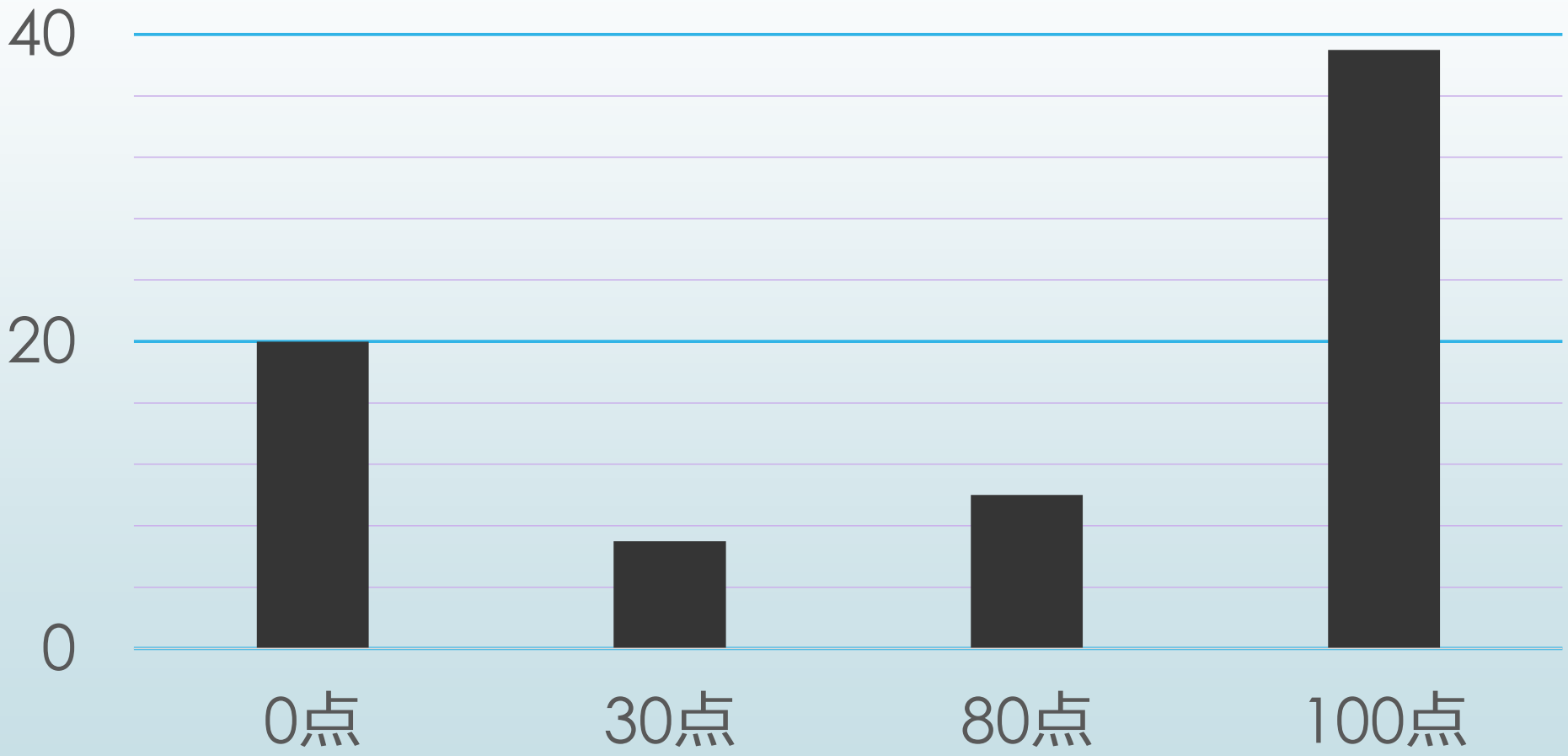
- 前述の解法にすこし実装を追加をするだけ。

高速な解法 (解析)

- 1 回目の $dfs(a)$ の計算で $O(M^2)$ かかる。
- $dfs(0)$ から $dfs(N)$ まで $O(N)$ 個ある。

- 時間計算量 $O(NM^2)$
- MAX, MIN を効率よく求めると $O(NM)$
- 小課題 2,3 が解ける。

得点分布



もくじ。(再掲)

- 問題概要
- 簡単な解法 (方針/実装/解析/評価)
- 高速化への道 (発想/手法)
- 高速な解法 (実装/解析)
- 得点分布

なにか質問があれば

第15回情報オリンピック本選 問題2 – スタンプラリー2 (Collecting Stamps 2)

解説: 劉 鴻志

問題概要

- N文字の J,O,I からなる文字列が与えられる
- 文字列中の(先頭・末尾を含む)自由な位置にJ,O,I のいずれかの文字を1文字挿入し、新たな文字列から3文字を取り出し、順番を入れ替えずに並べると"JOI"となるような3文字の取り方を最大化し、その最大値を求めよ

- 3≦N≦100,000

サンプル1

- JOIOI←元の文字列
- 1文字目の後にJを挿入すると…
- JJOIOI←このようになる。

ここから条件を満たす3文字の取り方は…

JJOIOI	JJOIOI	JJOIOI
JJOIOI	JJOIOI	JJOIOI

- の6通り

考察(前半)

- 新たな文字列から条件を満たす3文字の取り方の数を考えてみよう
- 3重ループで1文字目、2文字目、3文字目を決め、それぞれが”JOI”となっているかチェック
  - これは$O(N^3)$

考察(前半)

- 2文字目の’O’の位置を決めると、その’O’より前の’J’ならどれを取ってもいいし、その’O’より後の’I’ならどれを取ってもいい
- 逆に、上記以外のJやIを取ったら並べても”JOI”にならない
  - 3文字目の’O’を決めると…
  - それより前の’J’は2文字、それより後の’O’は2文字で、2×2通りの”JOI”となる取り方が作れることがわかる

JJOIOI	JJOIOI
JJOIOI	JJOIOI

考察(前半)

- なので、’O’の場所を決めると
  Jnum=(‘O’より前の’J’の数)
  Inum=(‘O’より後の’I’の数) とすると、
  Jnum×Inum通りの取り方がある
- Oの位置の決め方がO(N)通り、それより前の’J’の数、後の’I’の数を数えるのにO(N)で、合計$O(N^2)$

考察(前半)

- 実は、前ページの方針は簡単にO(N)に高速化できる。
- 1≦i ≦Nに対し、i文字目までの'J'の数や、i文字目までの'I'の数を覚えておく
  - i番目までの'J'の数＝i−1番目までの'J'の数＋(i番目の文字が'J'なら1;そうでなければ0)
    のように計算していくと、この処理もO(N)でできる
  - 累積和と呼ばれるものの最も基本的な形！
- 選んだ'O'の場所がj番目だとすると、j文字目までの'J'の数が Jnum と等しく、
- 全体の'I'の数−j文字目までの'I'の数が Inum と等しいため、選んだ'O'に対し定数時間で計算できる

考察(後半)

- 以上は挿入する文字と場所を決めた後の話だった
- では、挿入する文字と場所を決めるにはどうすればよいか？
- 単純な方法:挿入する文字と場所を全て試して、それぞれに対して3文字の取り方を求め最大値を出す
  - O(N)通り

考察(後半)

- 1文字挿入した後の文字列での3文字の選び方は、(元の文字列での3文字の選び方)+(挿入した文字を使った3文字の選び方) と等しいことに注目
- (元の文字列での3文字の選び方) は、考察前半の方法のうちいずれかで求めておく

考察(後半)

- 挿入した文字を使い、3文字選ぶ場合を考えよう。2重ループで残りの2文字を決めると$O(N^2)$で、挿入する場所・文字が$O(N)$通りなので、$O(N^3)$

考察(後半)

- 挿入した文字が'O'のとき、考察前半と同様に、それより前の'J'の数とそれより後の'I'の数を掛ければいい
  - 挿入場所がO(N)、毎回O(N)の計算なのでO(N^2)
- 'J'を挿入するとき、文字列の先頭に入れるのが常に最適
  - 'J'を挿入して新たにできる3文字の選び方は、挿入した文字以降の"OI"となる2文字の取り方なので、'J'の入れる場所は前にすれば2文字の取り方は増えていく

考察(後半)

ex) JOIOI の先頭に'J'を挿入したとき、それ以後で"OI"となる2文字の取り方は…

JOIOI	JOIOI	JOIOI

JOIOIの2文字目の後に'J'を挿入すると…

IOI

↑のみが挿入した'J'の後で"OI"を作る2文字の取り方となる

- 'I'も、同様の理由で末尾に挿入するのが最適
- よって、挿入場所は定数個・2文字の取り方は$O(N^2)$で計算できる
- J,O,Iどの場合も$O(N^2)$

考察(後半)

- 挿入する文字が’O’のとき、考察前半と同様に、累積和を使い高速化できる
  - これでO(N)
- 挿入する文字が’J’の場合はどうか？
- 先頭に挿入することが決まると、元の文字列で”OI”となる2文字の取り方の数を求めればいい

考察(後半)

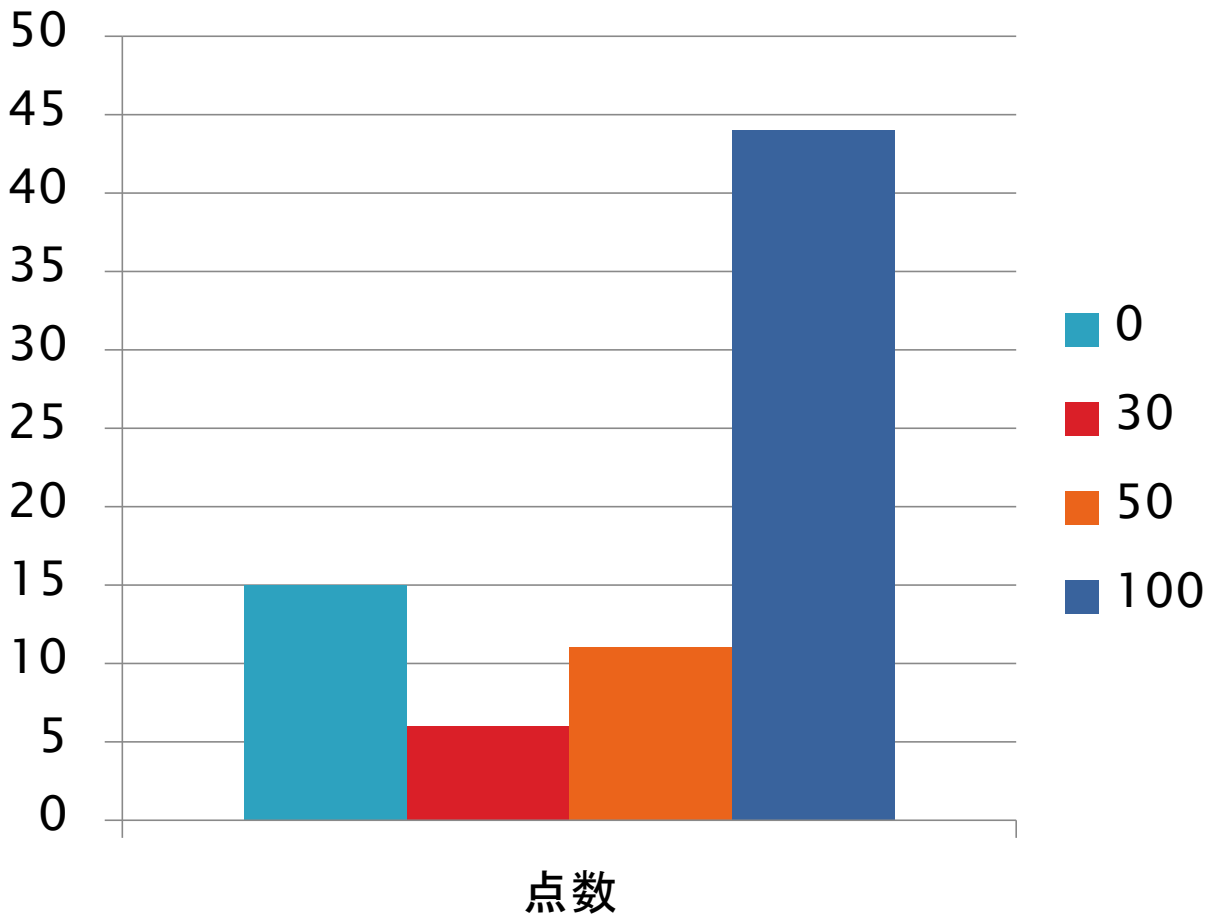
- これは考察前半で考えた問題の簡単版で、'O'の場所を決め、それ以後の'I'の数を累積和で求め、その総和を取ればOK
- 挿入する文字が'I'の場合も同様
- O(N)

まとめ

考察2＼考察1	$O(N^3)$	$O(N^2)$	$O(N)$
N回計算	$O(N^4)$	$O(N^3)$	$O(N^2)$
変更分のみ計算、$O(N^3)$	$O(N^3)$	$O(N^3)$	$O(N^3)$
変更分のみ計算、$O(N^2)$	$O(N^3)$	$O(N^2)$	$O(N^2)$
変更分のみ計算、$O(N)$	$O(N^3)$	$O(N^2)$	$O(N)$

- 計算量は最も遅いところに左右されることに注意しましょう
- $O(N^3)$で30点、$O(N^2)$で50点、$O(N)$で100点です
- ちなみに他にもいろんなDP解法もあります

得点分布



JOI 2015-2016 本選 問題 3 鉄道運賃 (Train Fare) 解説

三谷庸

問題概要

* N 個の駅、M 個の鉄道路線がある
* Q 年間にわたる値上げ計画がある
* 値上げ計画の i 年目には、路線 R_i の運賃を値上げする
* etc

* グラフ理論の言葉で言い換えます

問題概要

* N 頂点 M 辺の無向グラフがある
* はじめ、すべての辺のコストは 1 である
* クエリが Q 回くる
* i 番目のクエリでは辺 R_i のコストを 2 にする
* 各クエリに対して、その時点で以下の条件を満たす頂点 v の個数を出力する
  * 頂点 v から頂点 1 への最短距離がはじめの状態よりも長くなっている

小課題 1

* N <= 100
* Q <= 30

* とにかく問題文に書いてある通りにやる
* 各頂点 v について、「頂点 v から頂点 1 への最短距離」を求めたい

小課題 1

* 最短距離を求めるには、Warshall-Floyd 法を用いればよい
* N 頂点のグラフでは、全点対最短距離が $O(N^3)$ で求まる
* クエリごとに全点対最短距離を求める
* 全体の計算量は $O(QN^3)$ となる

考察: 始点をひとつに

* 求めたい最短距離は N - 1 種類あり、始点は全て異なるが、終点は共通
* これを生かして速くできないか

考察: 始点をひとつに

* 求めたい最短距離は N - 1 種類あり、始点は全て異なるが、終点は共通
* これを生かして速くできないか

* 無向グラフでは、s-t 最短距離と t-s 最短距離は等しい
  * s-t 最短路を逆からたどると t-s 最短路になる

考察: 始点をひとつに

* したがって、今回の問題は次のように言い換えられる
  * 頂点 1 から頂点 v への最短距離が初めの状態より長くなっている v の個数を求めよ

考察: 始点をひとつに

* したがって、今回の問題は次のように言い換えられる
  * 頂点 1 から頂点 v への最短距離が初めの状態より長くなっている v の個数を求めよ
* 頂点 1 からほかのすべての頂点への最短距離を求めるには、Dijkstra 法を用いればよい

小課題 2

* N <= 100,000
* M <= 200,000
* Q <= 30
* クエリごとに、頂点 1 を始点として Dijkstra 法を行う
* 前ページのように問題を言い換えて、条件を満たす v を数える
* 全部で O(Q M log N) 時間

考察: コスト 2 の辺

* 頂点 1 から頂点 v への経路で、以下の条件を満たすものを「よい経路」と呼ぶことにする
  * コストが初めの状態での最短距離と等しい
* よい経路にはコスト 2 の辺は現れない

考察: コスト 2 の辺

* 頂点 1 から頂点 v への経路で、以下の条件を満たすものを「よい経路」と呼ぶことにする
  * コストが初めの状態での最短距離と等しい
* よい経路にはコスト 2 の辺は現れない
  * もし現れていたとすると、初めの状態では、全く同じように辺をたどってより小さいコストで頂点 1 から v へ移動できる

小課題 2 (別解)

* 各クエリにおいて、指定された辺のコストを 2 にするかわりに、その辺を削除する
* この場合、(残っている)すべての辺のコストが 1 となるので、最短距離を求めるのに幅優先探索 (BFS) を用いることができる
* 計算量は O(QM)

小課題 3

* N <= 100,000
* Q <= M <= 200,000
* 「正しい出力にあらわれる整数は 50 種類以下」

小課題 3

* N <= 100,000
* Q <= M <= 200,000
* 「正しい出力にあらわれる整数は 50 種類以下」

* クエリごとにグラフ全体を見ていては間に合わない
* 最後の条件をどう使うか
* 「答えが変わった」ことを検出したい

考察: 最短路に現れる辺

* 前の考察で、コスト 2 になった辺は考えなくてよいことが分かった
* 他の辺は全て考慮すべきだろうか

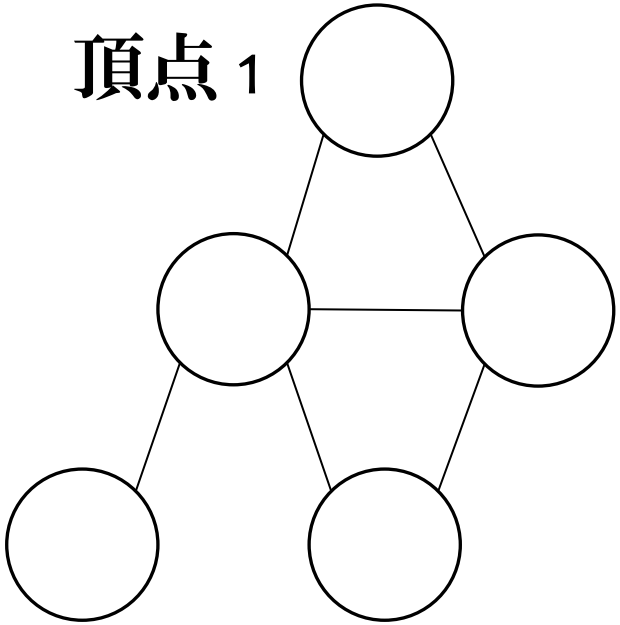
考察: 最短路に現れる辺

* 初めの状態のグラフで、各辺が頂点 1 からの最短路でどのように使われるか考える

* N 個の頂点を頂点 1 からの最短距離で分類する
  * 辺はどのような頂点を結んでいるだろうか？
  * それらの辺は最短経路でどのように使われるだろうか？

考察: 最短路に現れる辺

* 入力例 1 のグラフでやってみる



最短距離 0

最短距離 1

最短距離 2

考察: 最短路に現れる辺

* 以下のことがわかる
  * 辺の両端点の頂点の頂点 1 からの最短距離は (d, d) か (d, d + 1) のいずれか
  * (d, d) の辺はどの最短路にも使われない
  * (d, d + 1) の辺は d -> d + 1 の方向でのみ最短路に使われる

考察: 最短路に現れる辺

* 前述の観察の証明(3 つ順に)
  * 例えば (d, d + 2) の辺 (u, v) があれば、u まで距離 d で行った後この辺を使うことで v まで距離 d + 1 で行けてしまう
  * (d, d) の辺 (u, v) が最短路に u -> v の向きで使われたとすると、v までは距離 d、したがって u までは距離 d - 1 で行けてしまう
  * 3 つめも同じように考えると分かります

考察: 最短路に現れる辺

* いくつかの辺のコストが 2 になっても、よい経路(最短距離が初めと同じに保たれている最短路)で使われる辺は前の形だけ
* 初めのグラフで 1 回だけ BFS することで、最短路に使われる辺を列挙できる
* どの最短路にも現れない辺を削除するクエリは単に無視すればよい

考察: 最短路に現れる辺

* 最短路に現れる辺以外は無視する
* また、残った辺も最短路に使われる向きに限定する

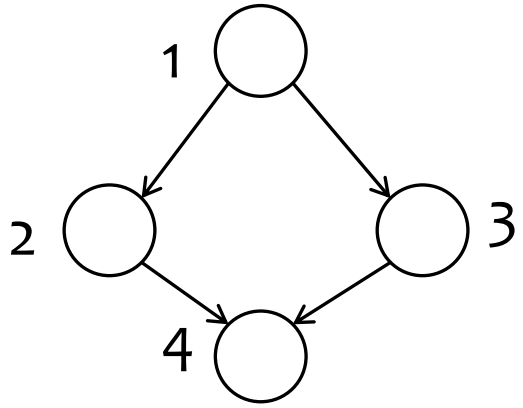
* このグラフは DAG (Directed Acyclic Graph, 閉路がない有向グラフ) となる

考察: 最短路に現れる辺

* このグラフの辺を消していくと考える
* 頂点 1 から頂点 v へのよい経路 (コストが初めの状態での最短距離と等しい経路) が残っていることと、このグラフ上で頂点 1 から頂点 v への経路が残っていることは同値

考察: 最短路に現れる辺

* この DAG の辺を消すことと、クエリに対する答えが変わることは同値ではない！！
* 下のグラフで辺 (3, 4) (右下の辺)を消しても経路が無くなる頂点は存在しない



考察: 最短路が消えるとき

* この DAG 上で辺を消してゆく
* 辺 u -> v を消すと、頂点 1 から頂点 v への経路がなくなったとする
* これはどのような状況か？

考察: 最短路が消えるとき

* この DAG 上で辺を消してゆく
* 辺 u -> v を消すと、頂点 1 から頂点 v への経路がなくなったとする
* これはどのような状況か？

考察: 最短路が消えるとき

* この DAG 上で辺を消してゆく
* 辺 u -> v を消すと、頂点 1 から頂点 v への経路がなくなったとする
* これはどのような状況か？

* 「この辺を消すことで頂点 v に (直接) 入る辺がなくなった」ということである

考察: 最短路が消えるとき

* したがって、各頂点について入次数 (その頂点に向かう辺の数) を記録しておくと、各クエリに対して「その辺を消すことでよい経路が無くなる頂点 v が存在するか」が O(1) で判定できる

考察: 最短路が消えるとき

* 辺 u -> v を消しても頂点 v へのよい経路が無くならない場合、すべての頂点についてよい経路があるかどうかは変わらない
* 頂点 v へのよい経路が無くなった場合、v 以外にもよい経路が無くなる頂点があるかもしれない
  * 例えば、1 -> 2, 2 -> 3 の 2 辺のみのグラフで辺 1 -> 2 を消すと、頂点 3 への経路もなくなる

小課題 3

* 「正しい出力に現れる整数は 50 種類以下」という条件が使えるようになった
* 辺を消すごとによい経路が無くなる頂点があるか判定し、ある場合はグラフ全体について BFS して各頂点についてよい経路の存在を判定する

小課題 3

* もう少し詳しく説明

init_dis[v] := 元のグラフでの頂点 1 から頂点 v への最短距離

cur_dis[v] := その時点での最短距離

として、

* 元のグラフの辺 (u, v) が u -> v として DAG に含まれることは、init_dis[v] = init_dis[u] + 1 と同値

小課題 3

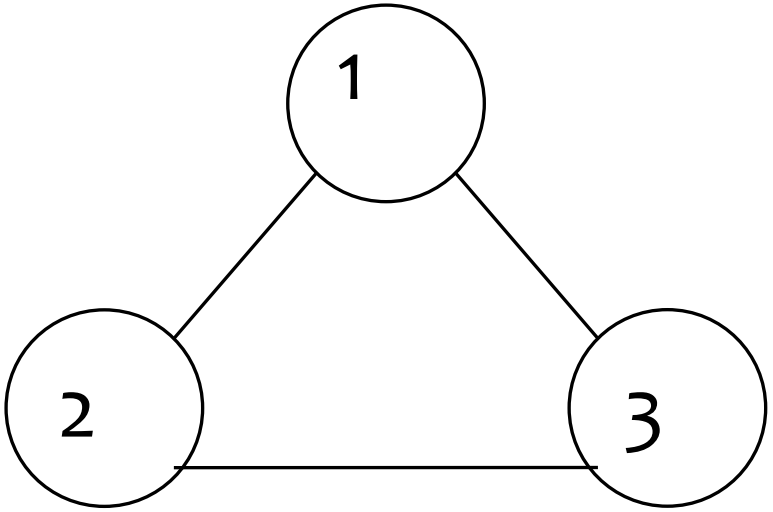
* 辺を消してよい経路の存在性が変わった時は、新たなグラフで BFS して cur_dis[v] を求める
  * DAG は明示的には持たずに、元のグラフからクエリの通りに辺を消していくだけでよい
* その後、cur_dis[v] == init_dis[v] となる v の個数を数える

小課題 3

* 新たなグラフで u -> v がよい経路に含まれることと、cur_dis[v] == cur_dis[u] + 1 は同値ではない！！
* 次のページのグラフで辺 (1, 3) を消すと、cur_dis[3] = cur_dis[2] + 1 だが、辺 (2, 3) は DAG の辺として考えてはいけない

小課題 3

* (グラフの例)



小課題 3

* 各頂点に対して入次数 indeg[v] を持っておく
* u -> v を消したときに indeg[v] -= 1 すべきかどうかの条件は、次(の両方を満たすこと)が正解
  * cur_dis[v] == init_dis[v]
  * cur_dis[v] == cur_dis[u] + 1

小課題 3

* まとめると、辺 (u, v) を消すクエリでは以下のことをすればよい

よい経路の存在性が変わるかどうかを indeg から判定

変わるときは、(O(M) かけて) すべての頂点について最短距離を計算しなおす

小課題 3

* 計算量は O(Q + TM) (= O(TM)) (Q <= M なので)
  * ただし T は出力に現れる整数の種類数

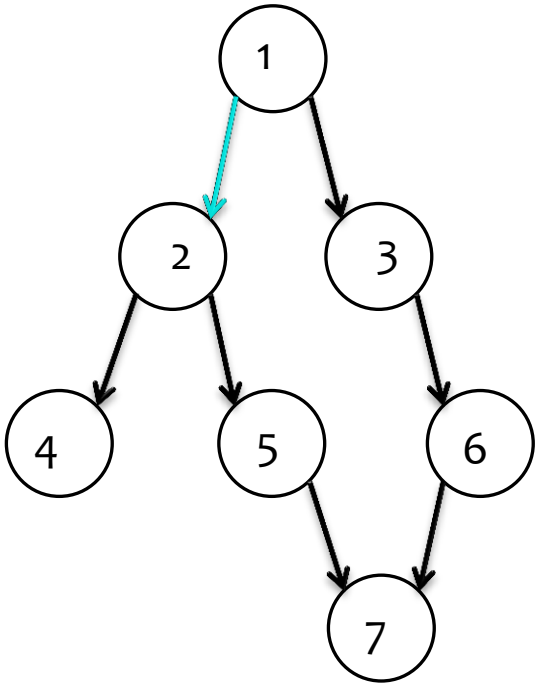
小課題 4

* Q <= M <= 200,000

* 同じ辺を何度もみていては間に合わない
* 小課題 3 の T 回の BFS では似たような計算をやっていそう
* 各頂点について「その頂点への経路が消えるのはいつか」を求めたい

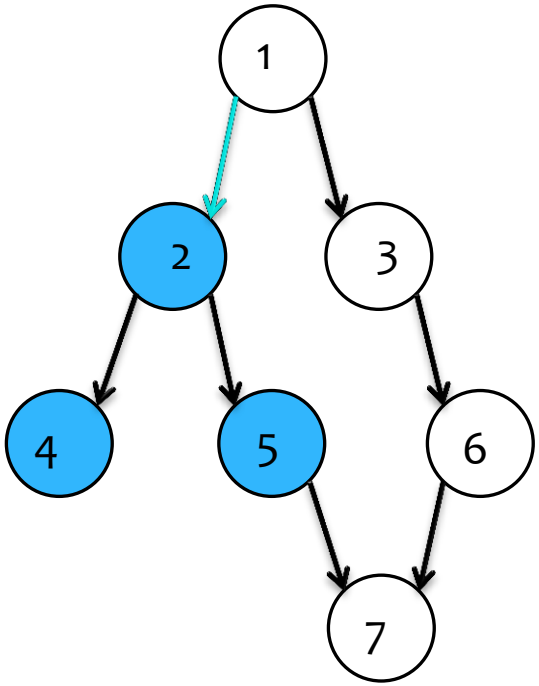
経路がなくなる頂点の列挙

* 下のグラフで、辺 1 -> 2 (左上の水色の辺) を削除するとどうなるか



経路がなくなる頂点の列挙

* 水色に塗られた頂点への経路がなくなる



経路がなくなる頂点の列挙

* 頂点 2 からいくつかたどっていける頂点が水色に塗られている

経路がなくなる頂点の列挙

* 頂点 2 からいくつかたどっていける頂点が水色に塗られている
* これらの頂点は、頂点 2 から DAG 上で BFS (DFS でもよい) すれば列挙できる
* BFS を使って経路がなくなる頂点だけを記録したい (前のグラフの場合、頂点 7 (一番下の頂点) は記録しないようにしたい)

経路がなくなる頂点の列挙

* BFS で頂点 v を選んでいるとき、次の条件が満たされるならば、v には経路がなくなったと記録せず、v から出る辺は見ない
* 逆に、満たされないならば、v を記録して、BFS を継続する
  * BFS で見ていない辺だけを使う頂点 1 から頂点 v への経路が存在する

経路がなくなる頂点の列挙

* 条件を満たすかどうかは次のようにして判定する

indeg[v] := (頂点 v へ向かう辺で、よい経路に含まれうるものの個数) となるようにする

BFS で u を選んでいるとき、辺 u -> v の処理は、

```
indeg[v] -= 1
if indeg[v] == 0
        v を経路がなくなった頂点として記録
        BFS を継続
```

小課題 4

* 以上のようにして、辺を削除した時にそれによって経路がなくなる頂点を列挙できる
  * DFS を使う場合もほぼ同じことをすればよい
* 経路がない頂点の個数をあらわす変数を持っておき、頂点への経路がなくなるたびに増やせばよい
* 計算量は？

小課題 4

* 最大 Q 回にわたる BFS (または DFS) において、辺 u -> v を見るのは、頂点 u が初めて到達不可能になった (経路がなくなった) 時の 1 回だけ
* よって、計算量は Q 回のクエリすべて合わせて O(M)

小課題 4

* BFS の初期条件や終了条件を間違えると TLE する可能性があります
* そうなったときは、「同じ辺を 2 回以上みない」アルゴリズムになっているかよく確認してみましょう

小課題４別解

* 最短路に関する考察はこれで終わりですが、少し違った方針で実装することもできます

小課題４別解

* 最短路に関する考察はこれで終わりですが、少し違った方針で実装することもできます
* 辺を「削除」するよりも「追加」する方がやりやすそう

小課題４別解

* クエリを逆から見る

* どのクエリにも現れない辺をあらかじめ追加しておく
* その後、後ろからｉ個のクエリの辺を追加した状況は、前からＱ-ｉ個のクエリの辺を削除した状況と同じになる

小課題 4 別解

* クエリの辺を逆から追加していき、各頂点についてどの時点で頂点 1 から到達可能になったかを調べることにする

小課題 4 別解

* 到達可能 (頂点 1 からの経路がある) な頂点 u と到達不可能な頂点 v について辺 u -> v を追加するとき、頂点 v は到達可能になる
* 頂点 v 以外にも到達可能になる頂点があるかもしれない
* そのような頂点は BFS (DFS でも)で列挙できる
  * 前の解法で探索を打ち切る条件を「到達不可能な頂点」から「到達可能な頂点」に変えるだけ

小課題 4 別解

* BFS の終了条件などは以下の通り

辺 u -> v の処理は

if (v への経路が存在)

　　v はこの BFS では無視 (すでに以前の BFS で記録済み)

else

　　que.push(v)

　　v への経路が存在すると記録

　　BFS を継続

小課題 4 別解

* 計算量は同じく O(M)
  * 辺 u -> v は頂点 v が初めて到達可能になるときだけみられる

小課題 4 別解

* (今回は逆から見なくても解けましたが) クエリを逆から見るという手法はとても重要です

小課題 4 別解 2

* 各辺について、その辺が削除されたのは何番目かを持っておく
* Dijkstra 法 (更新の方法を工夫する) を 1 回だけ行うことで、各頂点 v について以下の値が分かる
  * 頂点 1 から頂点 v まで、i 番目のクエリまでは残っている辺だけで最短距離でたどり着けるような最大の i
  * 最短距離の代わりに (距離, 経路が消える時間) を求めるようにすればよい
* これを使うとクエリに答えられる
* 計算量は O(M log N)

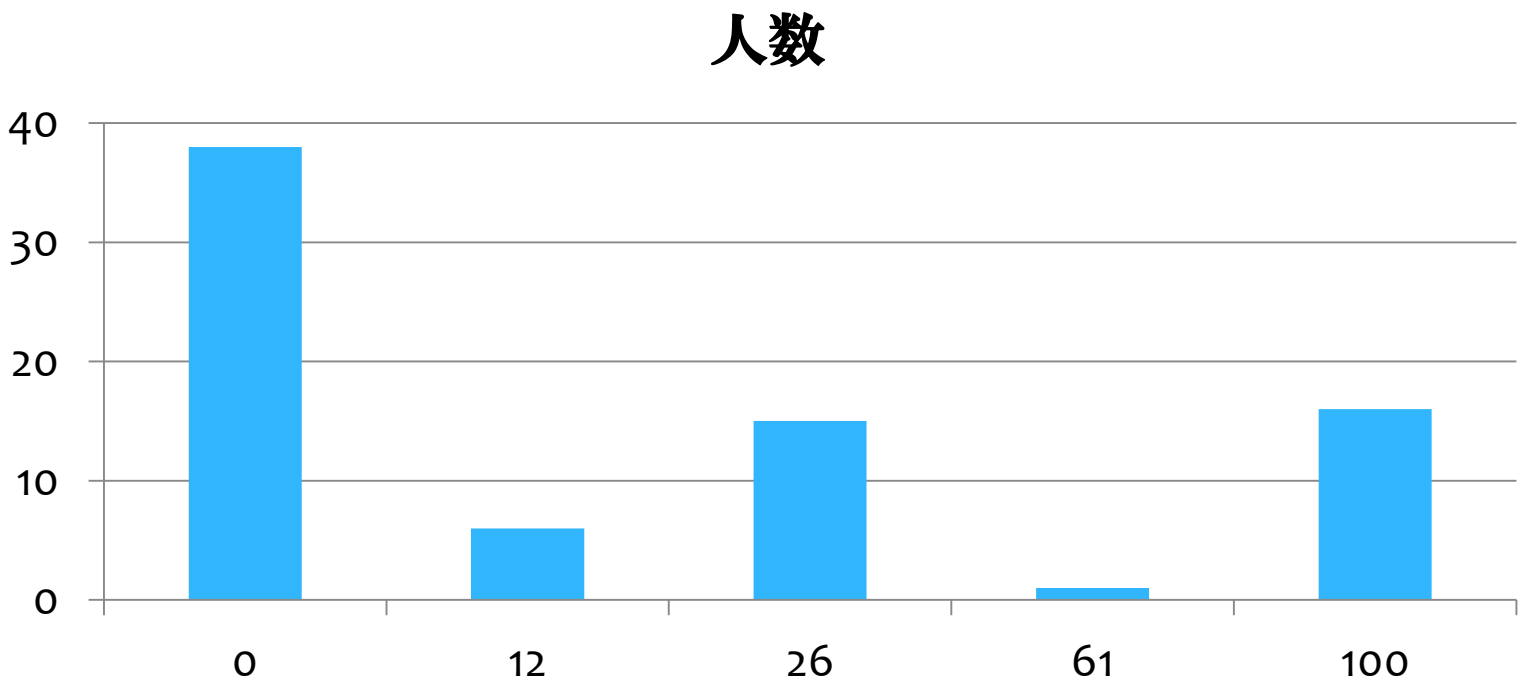
余談: 最短路問題

* プログラミングコンテストでは、最短路アルゴリズムとしては以下を覚えておくとよいでしょう
* 重みなしグラフの場合
  * 幅優先探索 (BFS)

余談: 最短路問題

* 重みつきグラフの場合
  * Warshall-Floyd 法
  * Bellman-Ford 法
  * Dijkstra 法
* 各アルゴリズムについて、計算量、適用できる条件を確認しておきましょう
  * なお、Dijkstra 法はコスト 0 の辺があっても使えます

得点分布



第 15 回
情報オリンピック本選
問題 4「縄張り(Territory)」解説

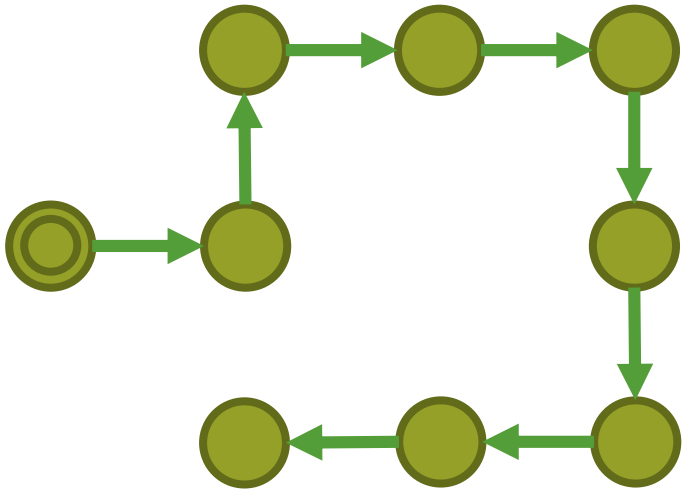
解説担当：城下慎也(phidnight) - JOI チューター

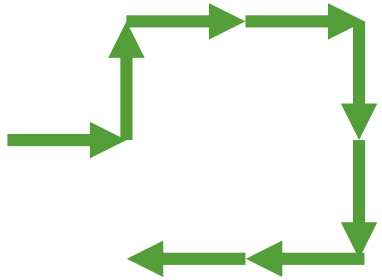
問題概要

- ジョイ君の散歩経路が与えられます。
- 同じ散歩経路を K 回反復します。
- 4 マス $(a, b), (a+1, b), (a+1, b+1), (a, b+1)$ のすべてを 1 回以上通るような (a, b) の総数を求めてください。
- $1 \leqq N \leqq 100{,}000$
- $1 \leqq K \leqq 1{,}000{,}000{,}000$

例 $(K = 3)$

- 1 日目までの移動経路

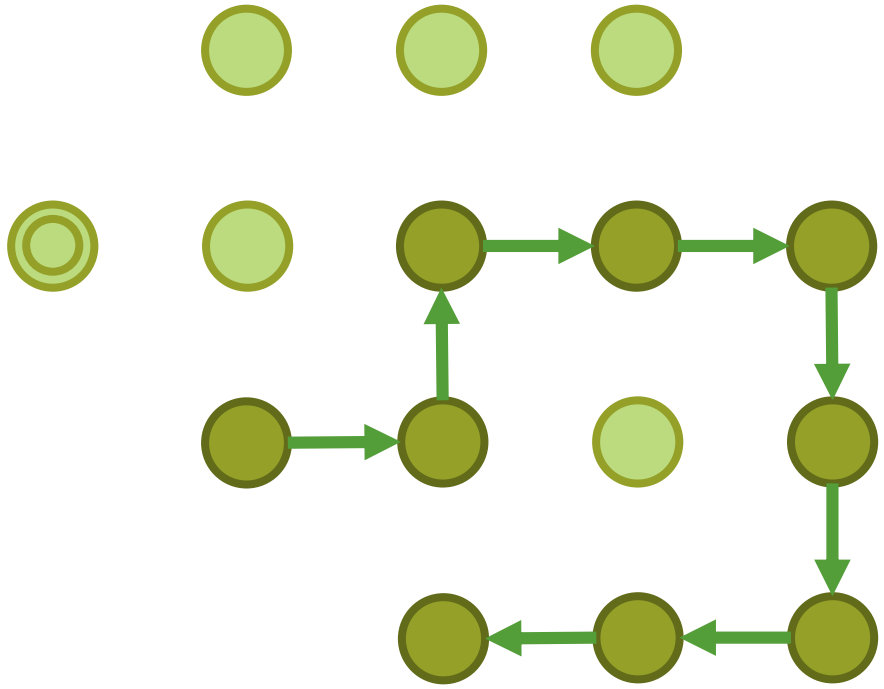


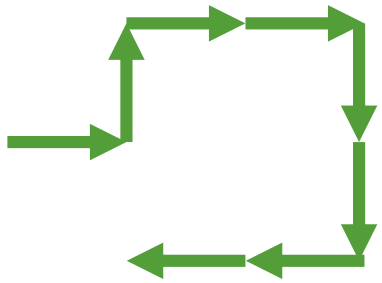


1 日の散歩経路

例 ($K = 3$)

- 2 日目までの移動経路

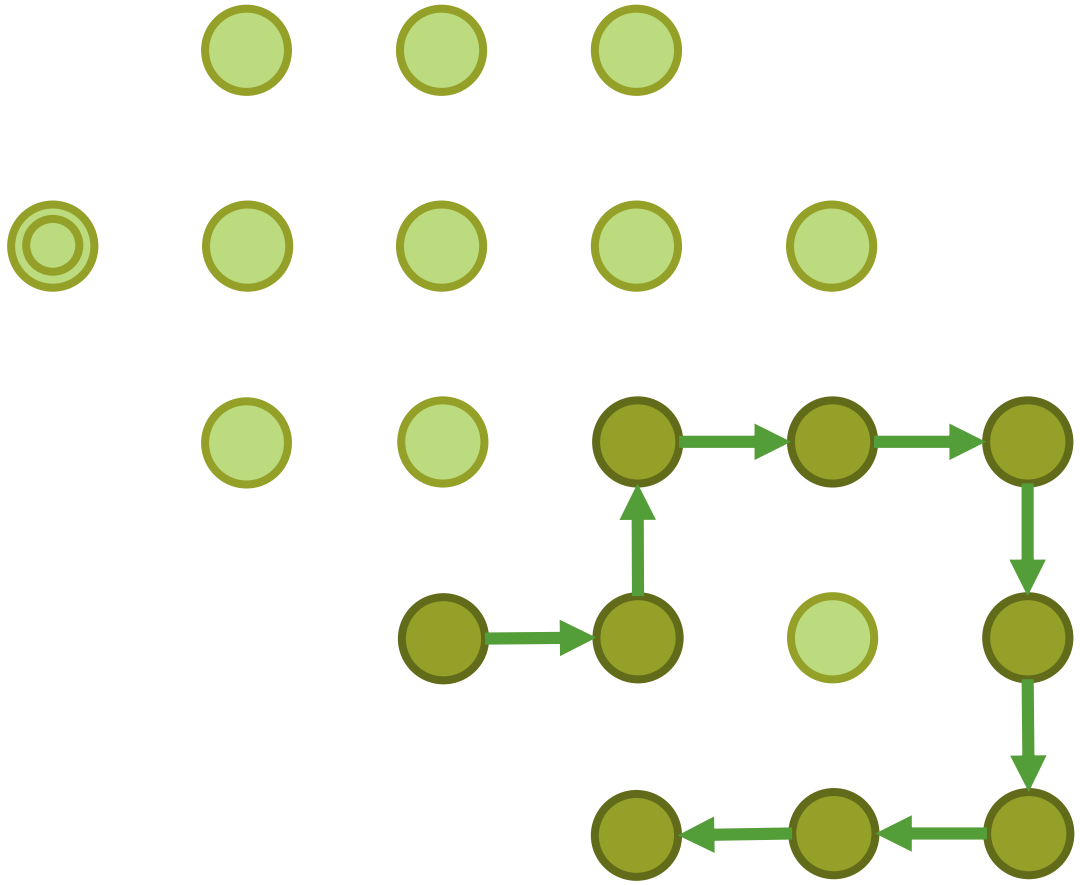


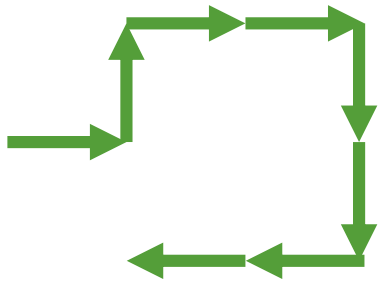


1 日の散歩経路

例 ($K = 3$)

- 3 日目までの移動経路

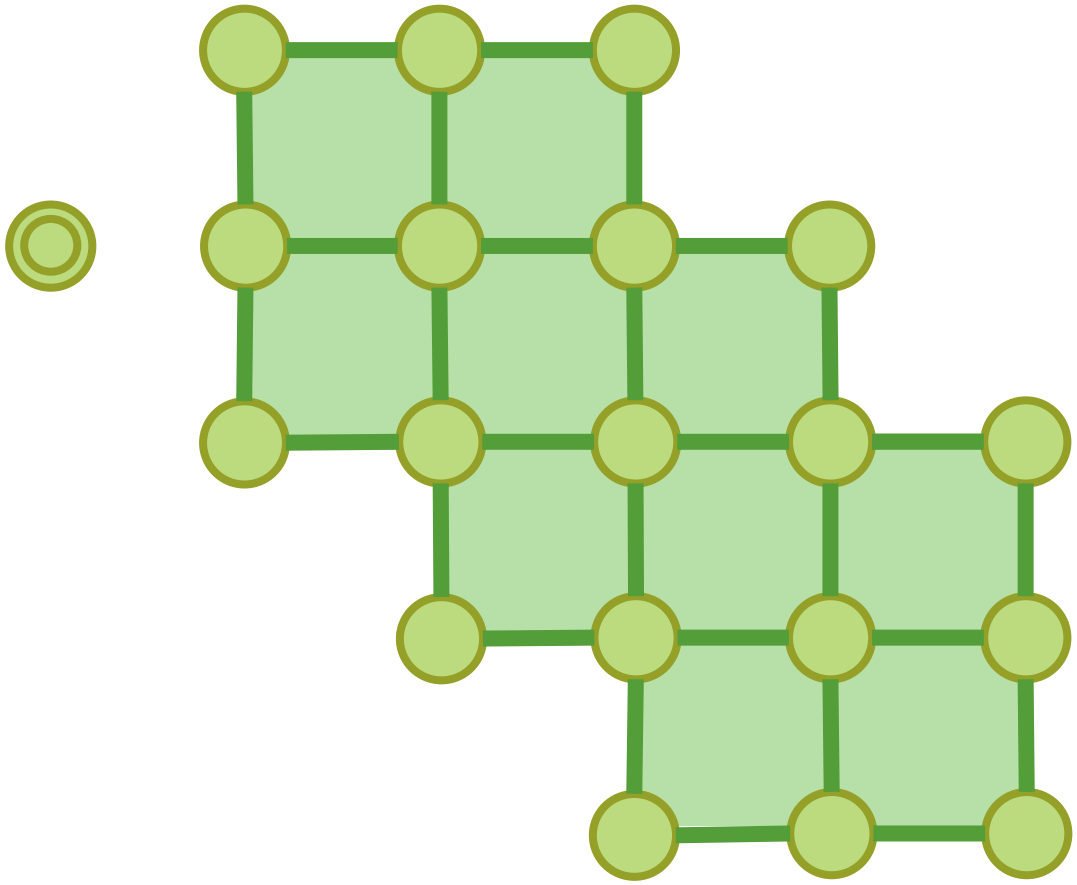


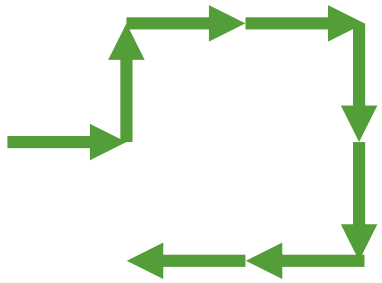


1 日の散歩経路

例 ($K = 3$)

- 3 日目までの移動経路





1 日の散歩経路

- 左図の 10 マスが縄張りに属するので、答えは 10 です。

小課題 1 解法

- $N \leqq 50$ かつ $K = 1$ です。
- 移動回数が少ないのでシミュレーションによって解を求めることができます。
- 移動範囲が、$(0,0)$ を中心とした 101×101 のマス目に収まるので、そのまま 2 次元配列上で実行できます。

小課題 2

- $K = 1$ です。
- 小課題 1 同様シミュレーションによって解を求めることができます。
- ただし、移動経路をそのまま 2 次元配列で格納しようとすると、 $\mathrm{O}(N^2)$ の空間を確保しなければなりません (メモリ制限上実行不能)。
- 実際には、先ほどの 2 次元配列内のほとんどの領域にアクセスしないことを利用します。

小課題 2 解法

- ジョイ君が通過したそれぞれの交差点を二分探索木や x 軸→ y 軸の順にソートした配列で管理します。
- 二分探索木や配列のそれぞれの要素を領域の南西端に固定して、南東端、北東端、北西端すべてについてジョイ君が通過したかどうかは、二分探索木の探索や配列への二分探索で求められます。
- 全体で $\mathrm{O}(N \log N)$ で計算できます。

小課題 3

- $N \leqq 50$ です。
- 総移動回数 NK が大きいので全移動をシミュレーションすることはできません。
- それぞれの日における移動経路が同じであることをうまく応用して考えます。

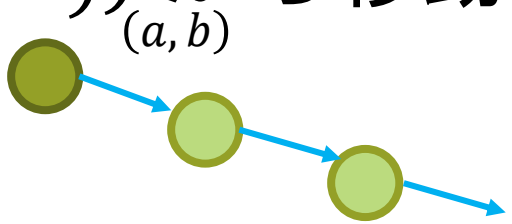
考察 1

- 初日の移動で $(0,0)$ から出発し $(0,0)$ に戻って移動が終了した場合を考えます。
- この場合、K の値にはよらず初日の移動だけを見れば良いです。

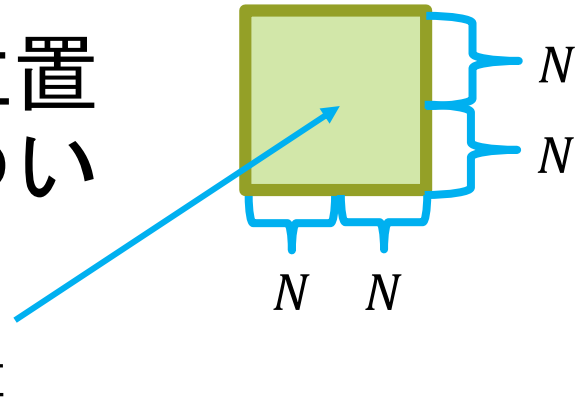
- 以降のスライドでは、このケースを例外処理したものとして扱います。

考察 2,考察 3

- 初日の移動で $(0,0)$ から $(a,b)((a,b) \neq (0,0))$ に移動して終了した場合を考えます。
- このとき、k 日目は $(a(k-1), b(k-1))$ から移動を開始することになります。



- また、それぞれの日について、初期位置から x 軸方向、y 軸方向それぞれについて N より遠ざかることはありません。

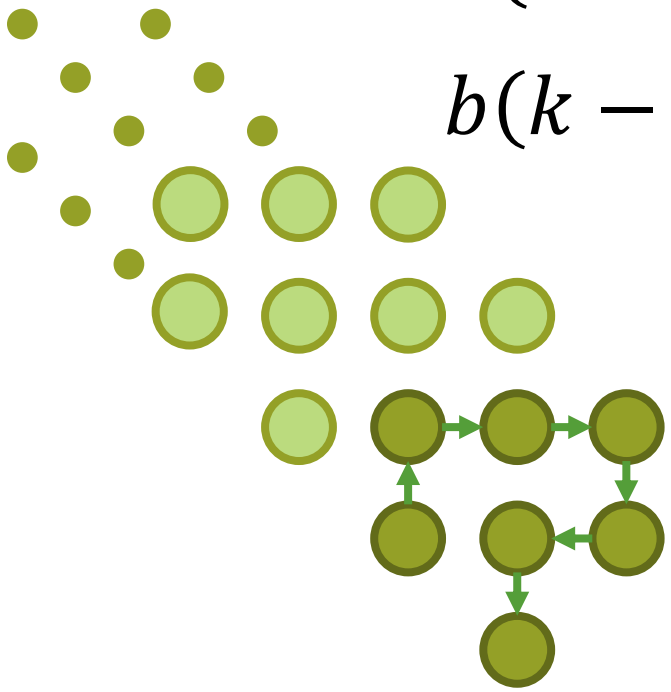


小課題 3 特有の考察

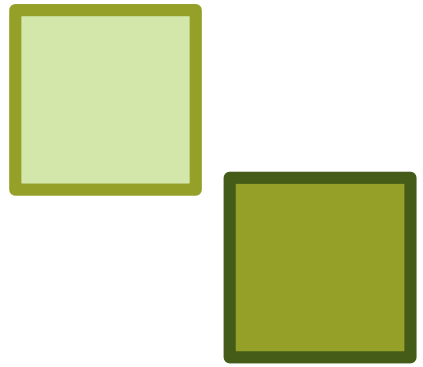
- K の変化に対して、解がどう変化するかを考えます。
- 考察 2, 3 により、k 日目に通過する交差点 (x, y) が

$$a(k-1) - N \leqq x \leqq a(k-1) + N \quad \text{および}$$

$$b(k-1) - N \leqq y \leqq b(k-1) + N \quad \text{を満たします。}$$

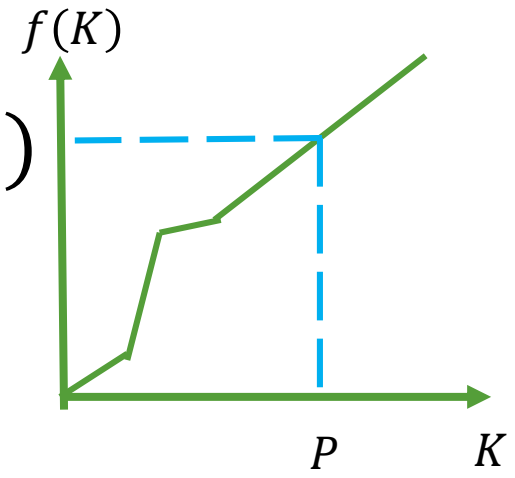


- これにより、k 日目において増える縄張りを考えた際、$k-2N-2$ 日目以前の移動による影響を受けないことがわかります。



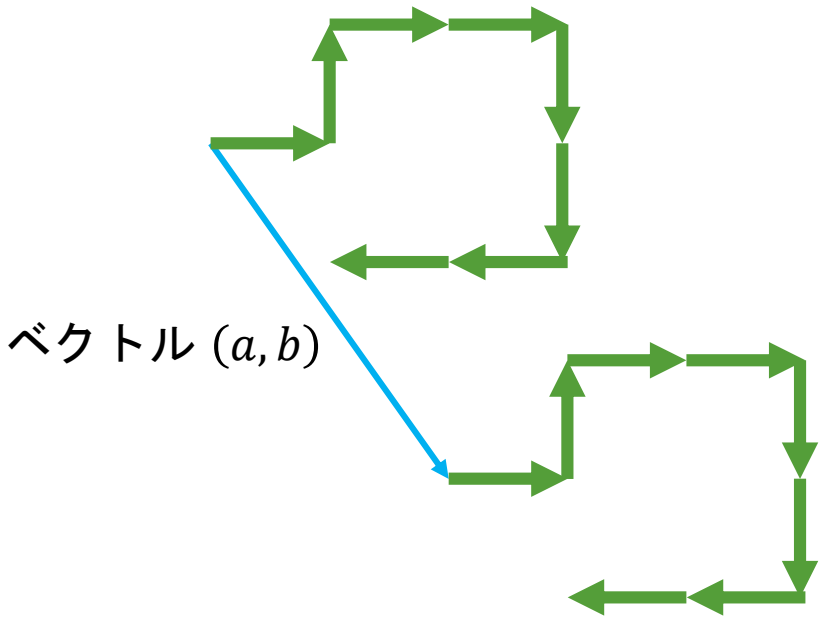
小課題 3 特有の解法

- それぞれの日における移動経路は同一なので、$2N+2$ 日目以降、K が 1 増えるごとに縄張りがどれだけ増えるかは一定になります。
- 以上より大きな K については、$K=P\ (P \geqq 2N+2)$ 時の解 $f(P)$ および $K=P+1$ 時の解 $f(P+1)$ を求めて、
$$f(K)=f(P)+(K-P)(f(P+1)-f(P))$$
から解を求めることができます。



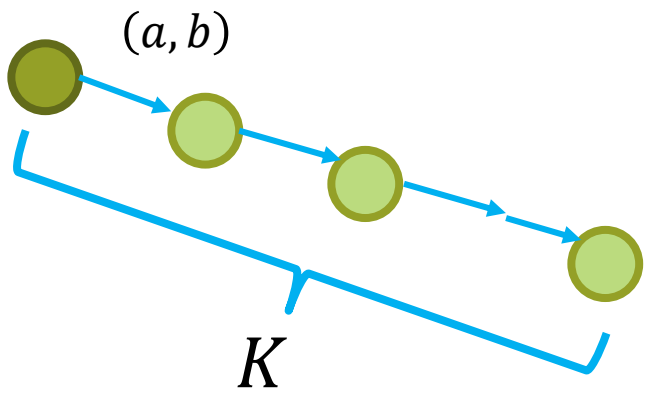
小課題 4

- 追加の制限はありません。
- 満点解法に至るためには、それぞれの日の移動経路がベクトル (a, b) による平行移動で変化していることを利用します。



考察 4

- ある交差点からベクトル (a, b) の方向にある交差点について考えます。
- すると、初日に経由する交差点から見て (ak, bk) $(0 \leqq k \leqq K-1)$ 進んだところにある交差点にも印が付けられているということがわかります。



満点解法の前に

- (a, b) は $a > 0$ および $b \geqq 0$ を満たすものと仮定しています。
- 実際の入力例ではそのような仮定を満たさないものも考えられますが、入力を適切に反転・回転させることにより上記の条件をみたすように変形できます。
- このような変形はコードの場合分けを減らし、バグ発生率を下げることもあります。

満点解法

- 印をつけた交差点 (x, y) を、

$(x, y) = (p + ar, q + br)$ ((a, b)は先程定めたベクトル)

を満たす (p, q, r) で定義します。

- ただし、 $0 \leqq p \leqq a - 1$ とします（こうすることで(x, y)に対する(p, q, r)が一意に定まります）。

満点解法

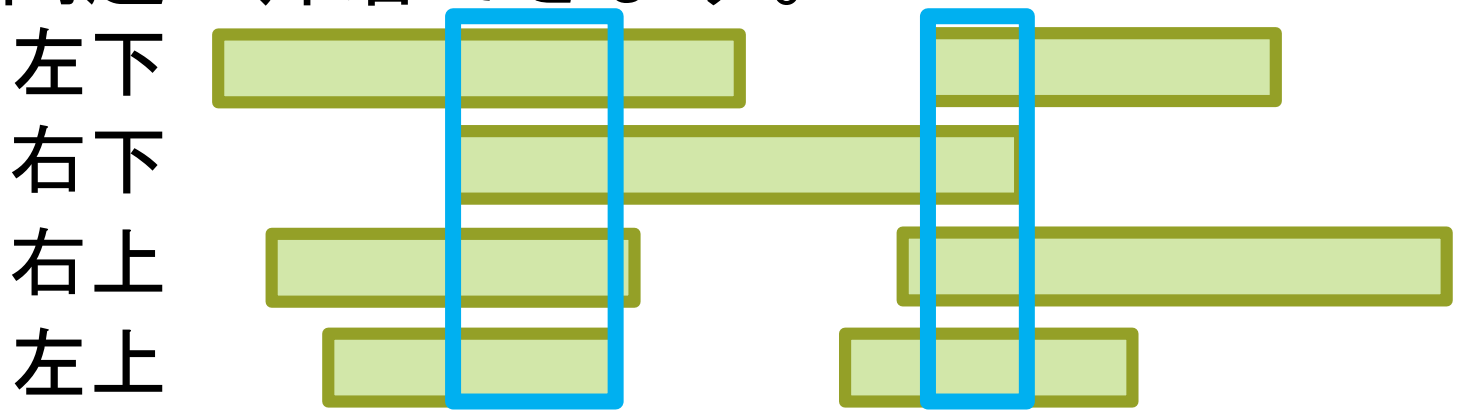
- すると、印が付けられた交差点は、以下のような長さ n $(n \leqq N)$ の組からなる配列 $(p_1, q_1, s_1, t_1), \ldots, (p_n, q_n, s_n, t_n)$ によって表現できます。

(p_i, q_i, s_i, t_i) : $s_i \leqq r \leqq t_i$ を満たす r に対し、(p_i, q_i, r) に印がある。

- 上記配列は二分探索木やソート済み配列で $\mathrm{O}(N \log N)$ で計算できます。

満点解法

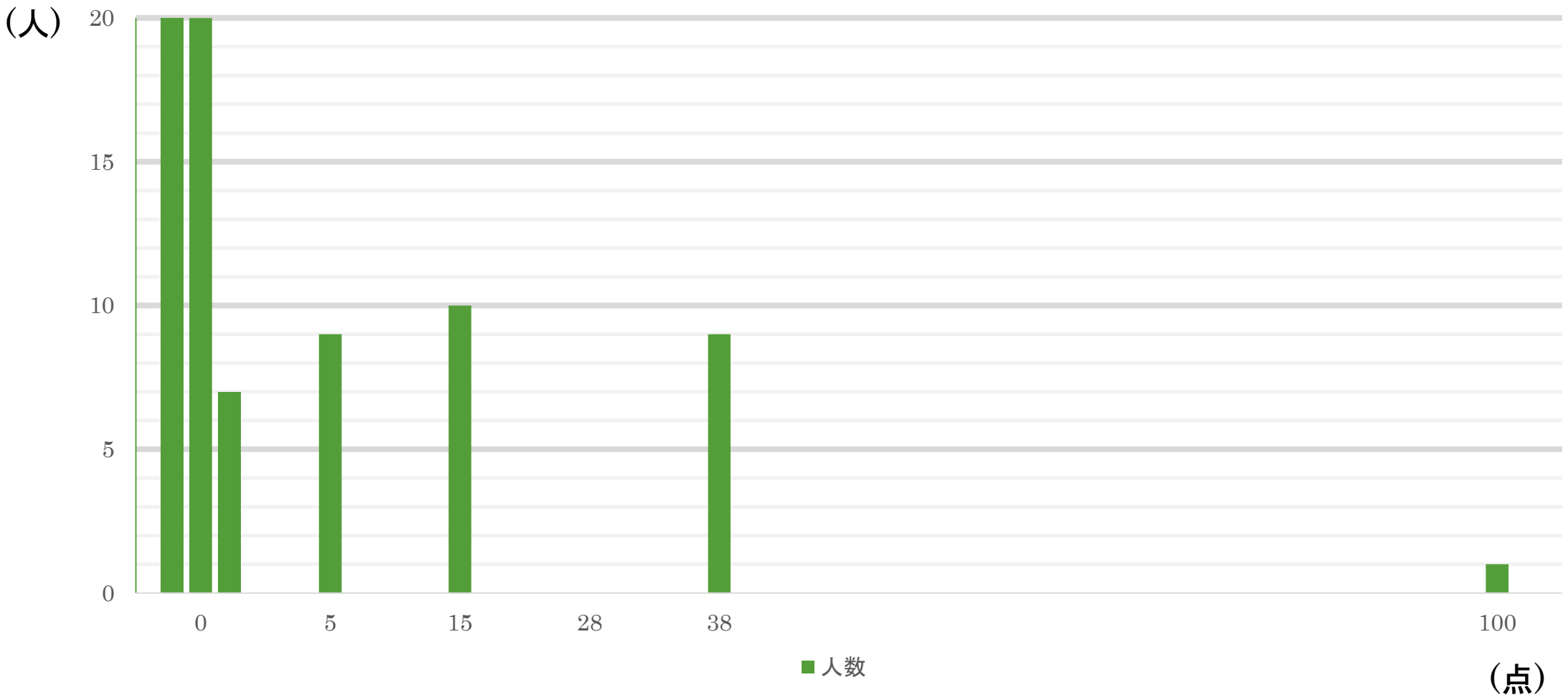
- 各 p, q に対し、左下の交差点が $(p, q, *)$ となる区画について、縄張りか否かを高速に計算したいです。
- 残り三隅における (p', q', r') のうち p', q' は一意に定まります。
- 最終的に下図のように 4 つの区間列の共通部分を求める問題に帰着できます。



満点解法

- 各左下に対して縄張りに属する個数を求めるための計算量は、その計算で登場する区間列の要素数(四隅それぞれの合計) n' に対し $\mathrm{O}(n' \log N)$ で計算できます。
- どの区間に対しても、四隅それぞれに高々 1 回ずつしか登場しないので、n' の総和は $\mathrm{O}(N)$ で抑えられます。
- 以上より $\mathrm{O}(N \log N)$ で解を求められます。

得点分布



断層(Geologic Fault) 解説

DEGwer

問題概要

- 地層がある
- 「地面が斜めに切れて上側が盛り上がる」というクエリがQ個与えられる
- Q回のクエリのあとの地表面の位置$1 \sim N$は何年前の地層か答えよ

問題概要

0

1

2

3

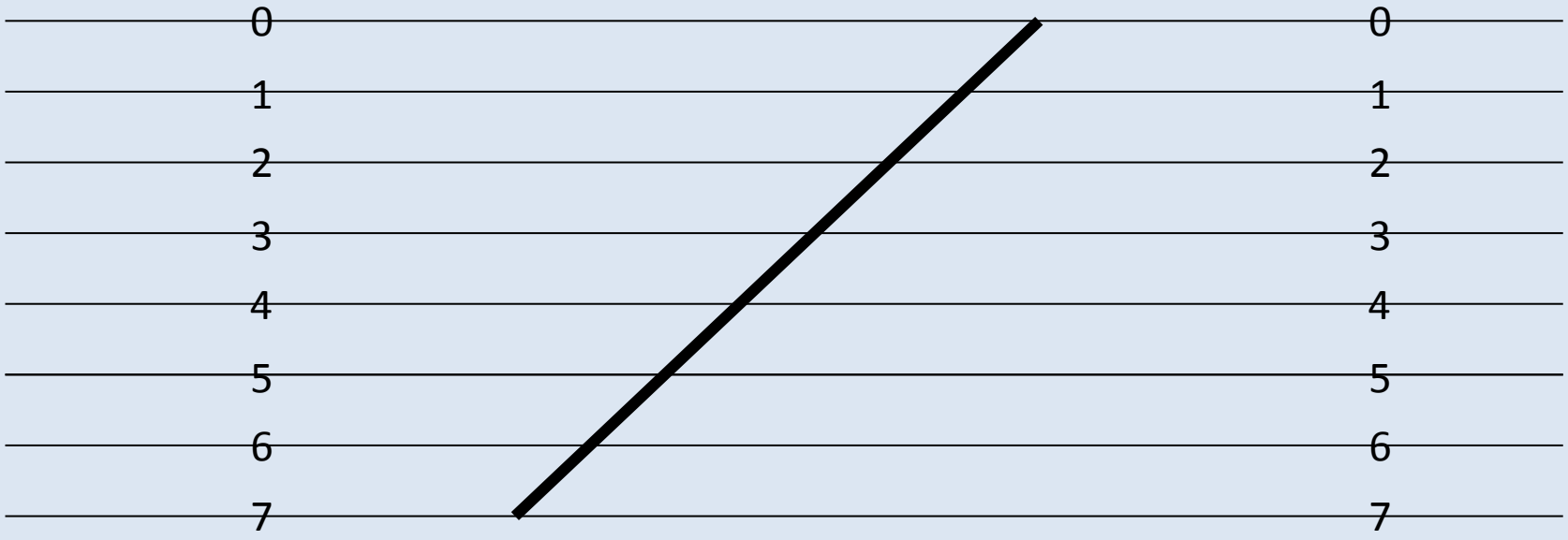
4

5

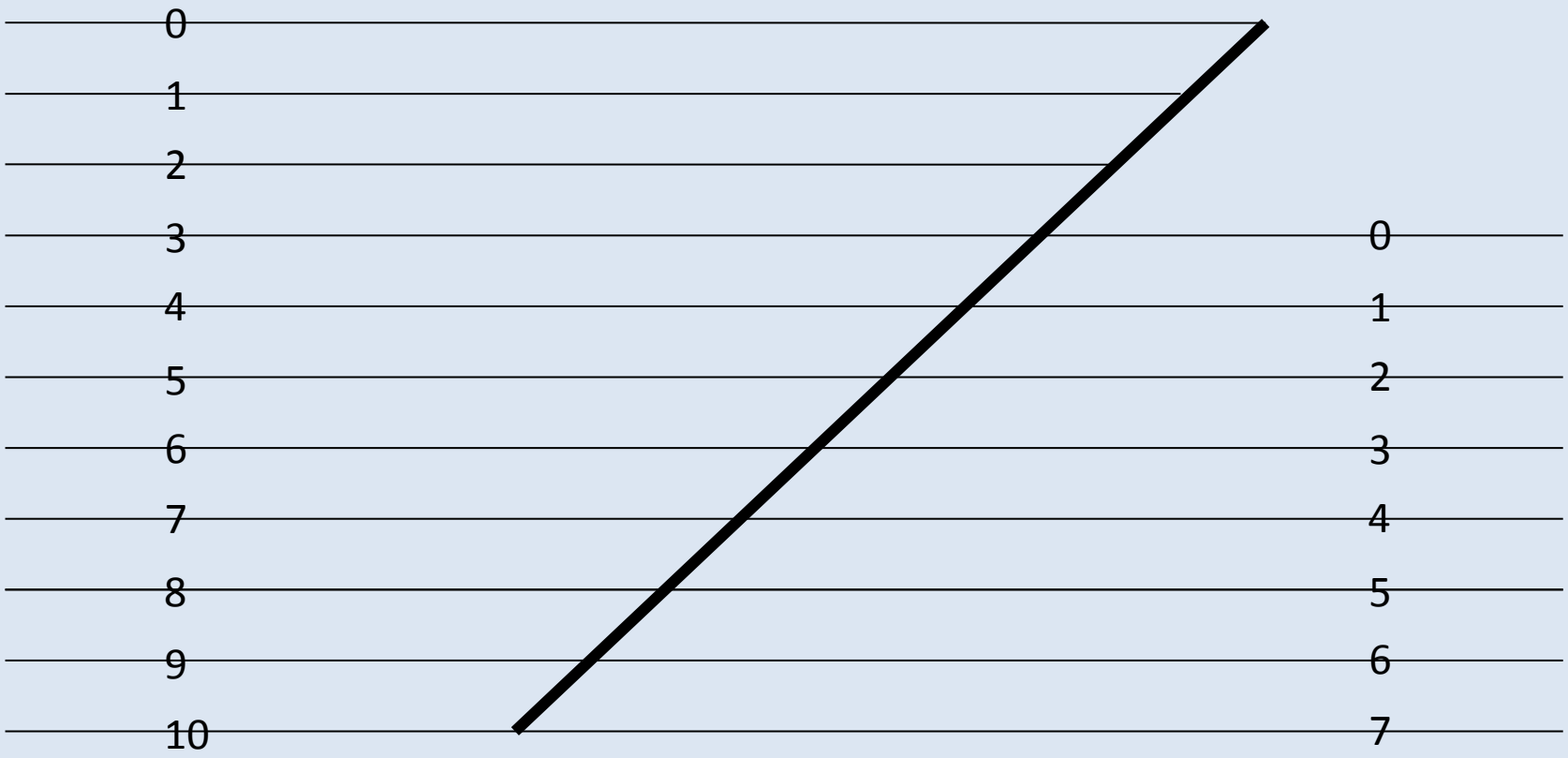
6

7

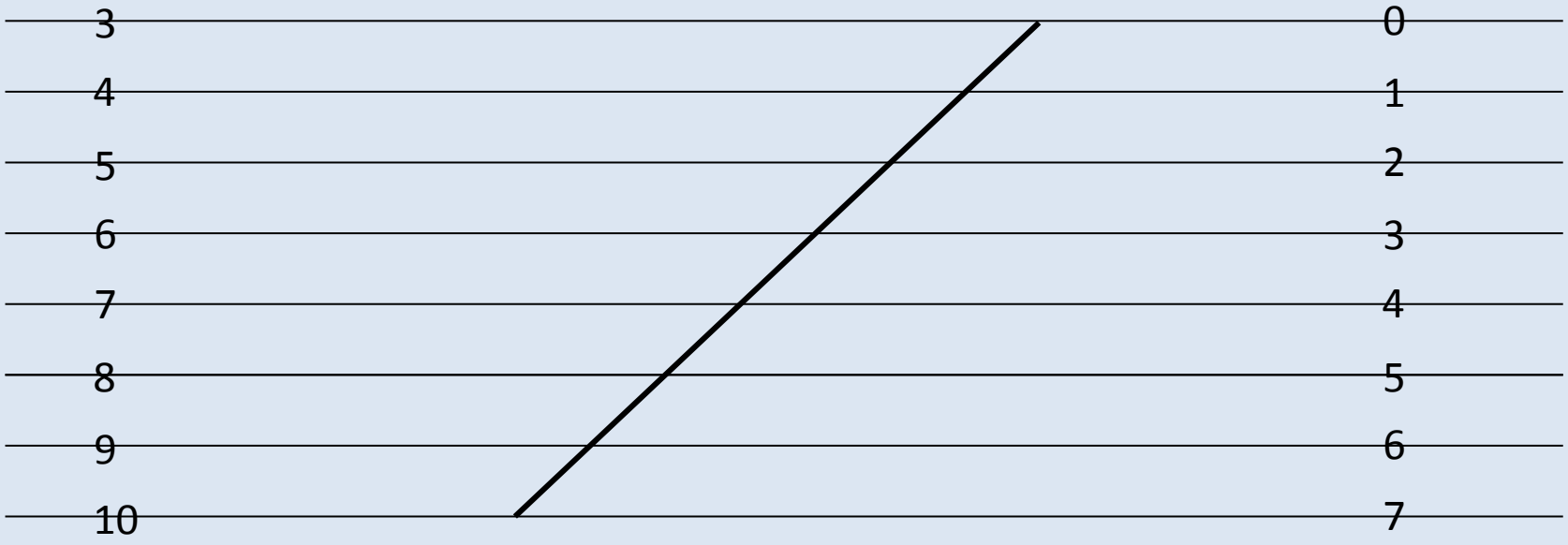
問題概要



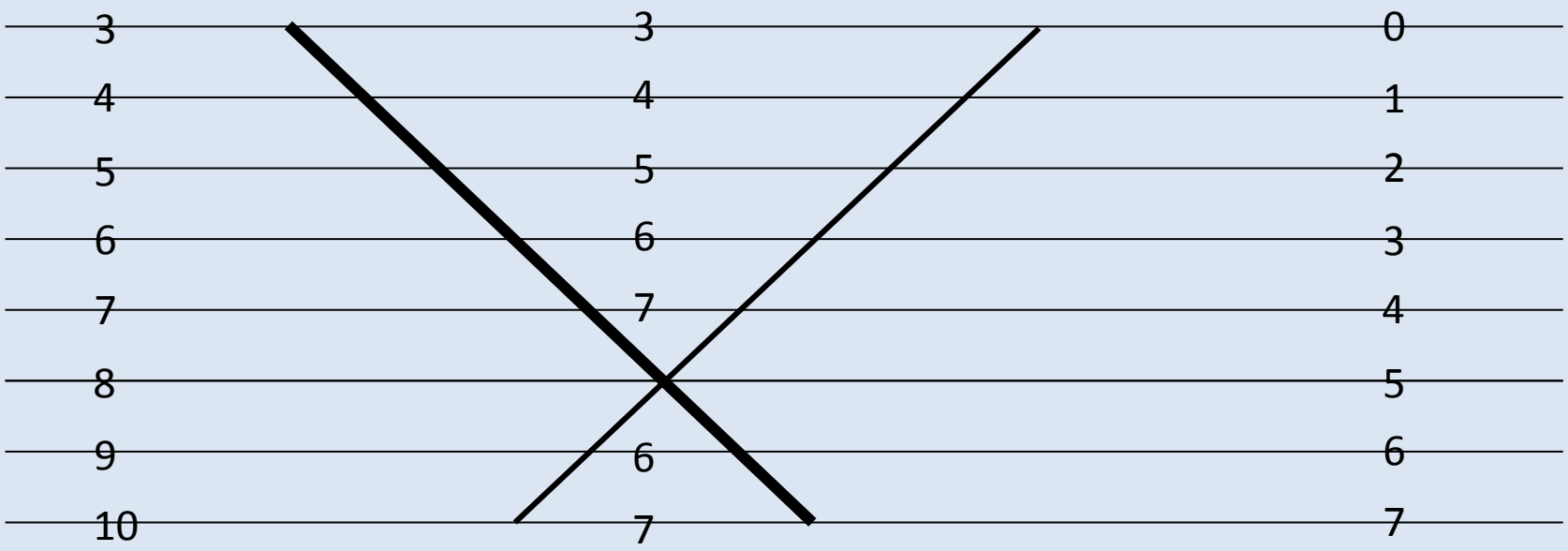
問題概要



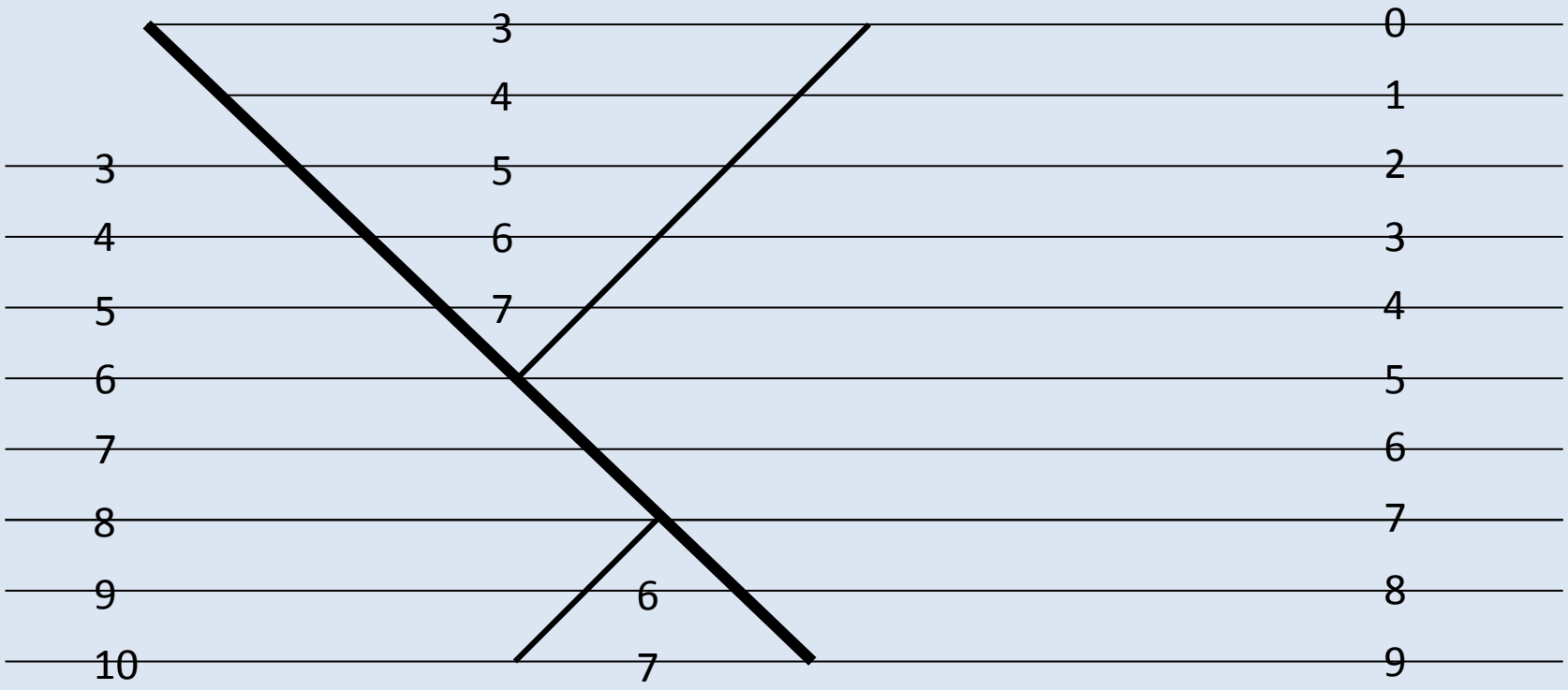
問題概要



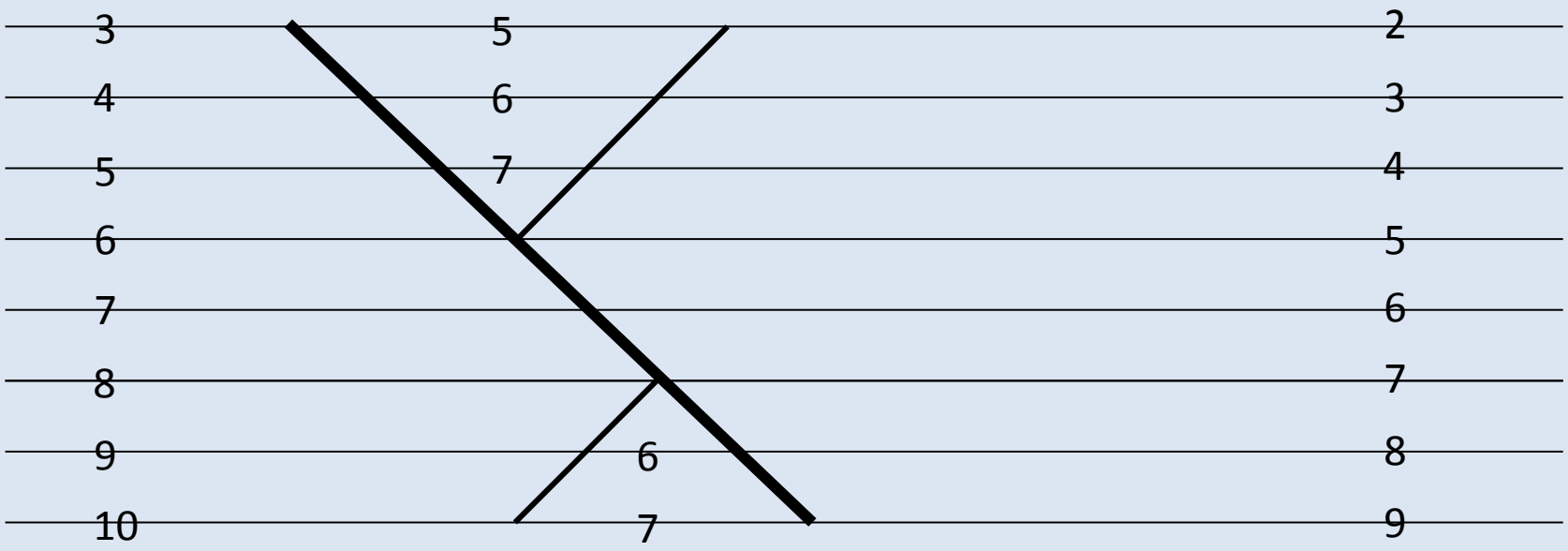
問題概要



問題概要



問題概要



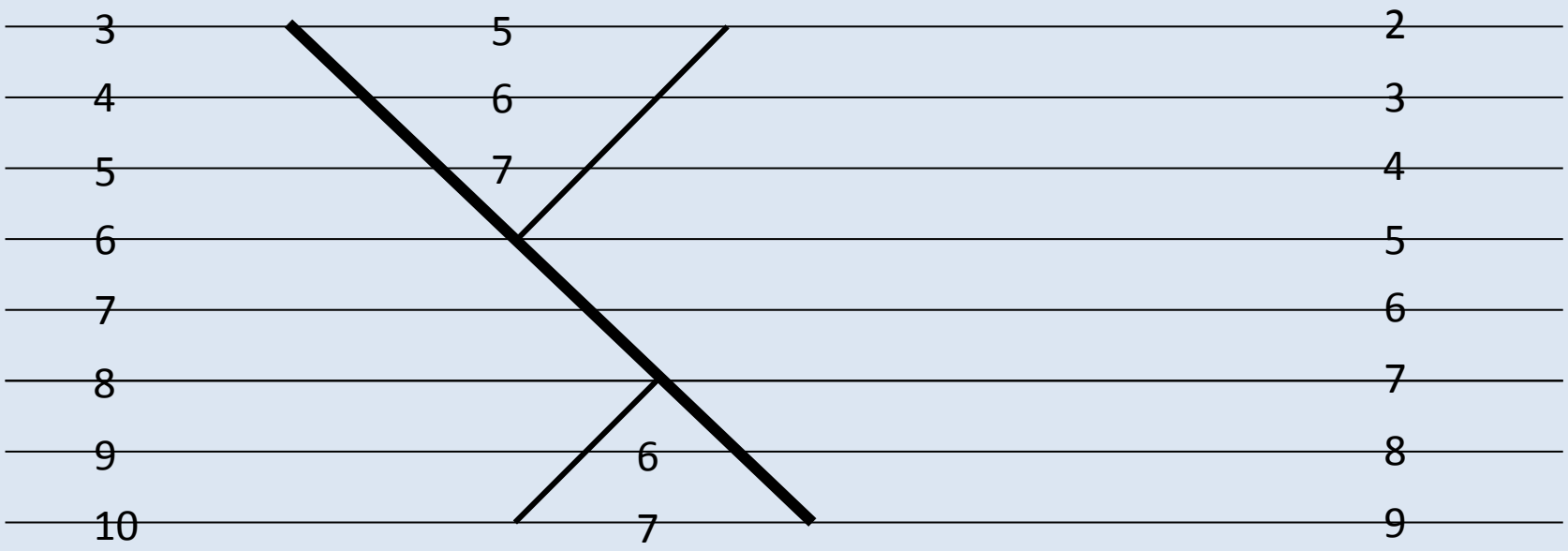
一般的テク

- ななめ45度というのは考えにくい
- こういうときは、図全体を45度回してしまうとよい

- マンハッタン距離を扱う時とかにもよく使うテク

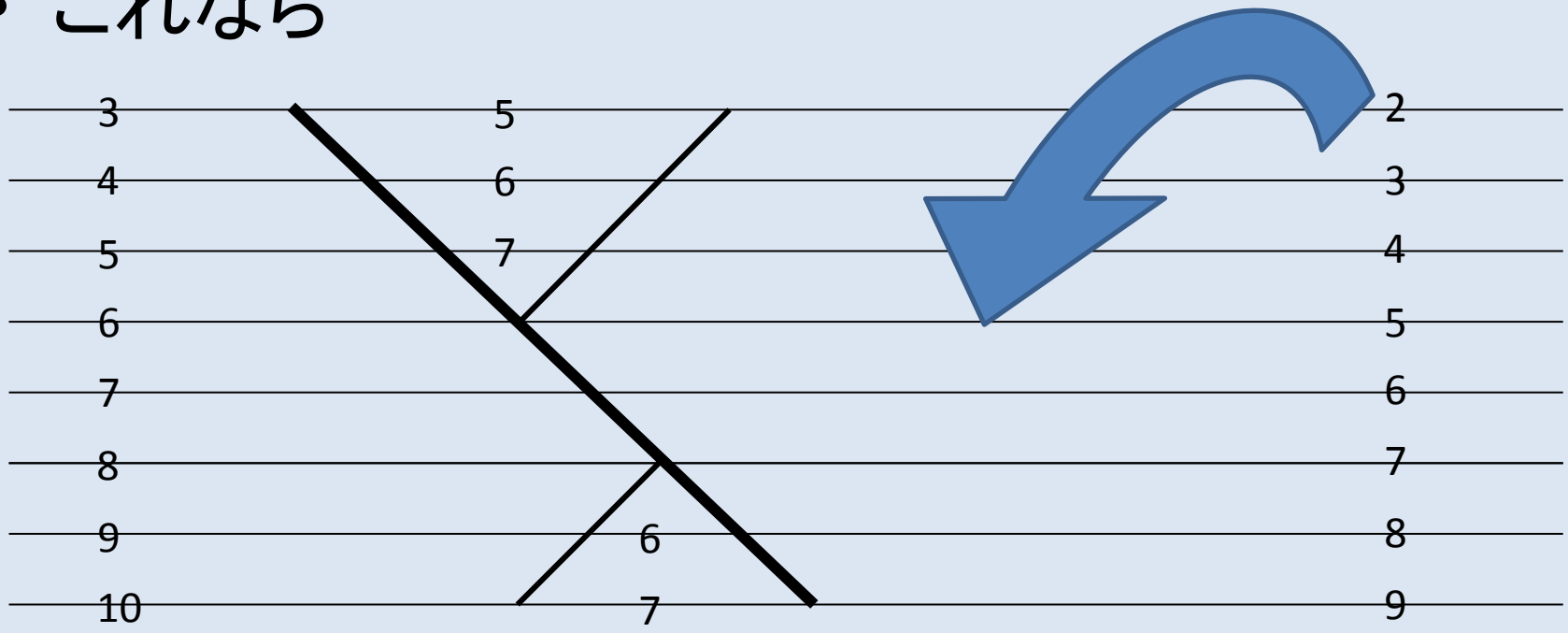
一般的テク

- これなら



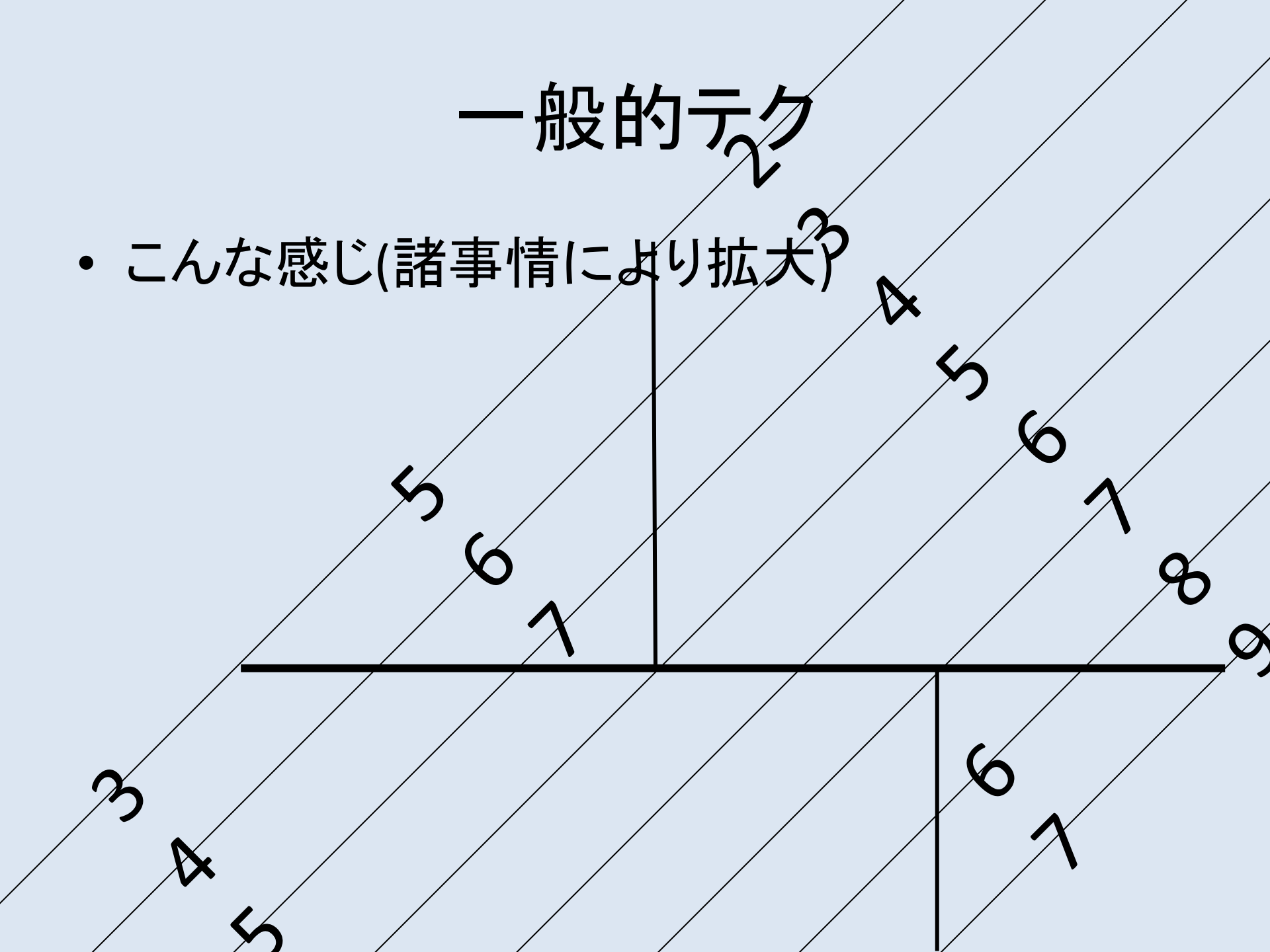
一般的テク

- これなら



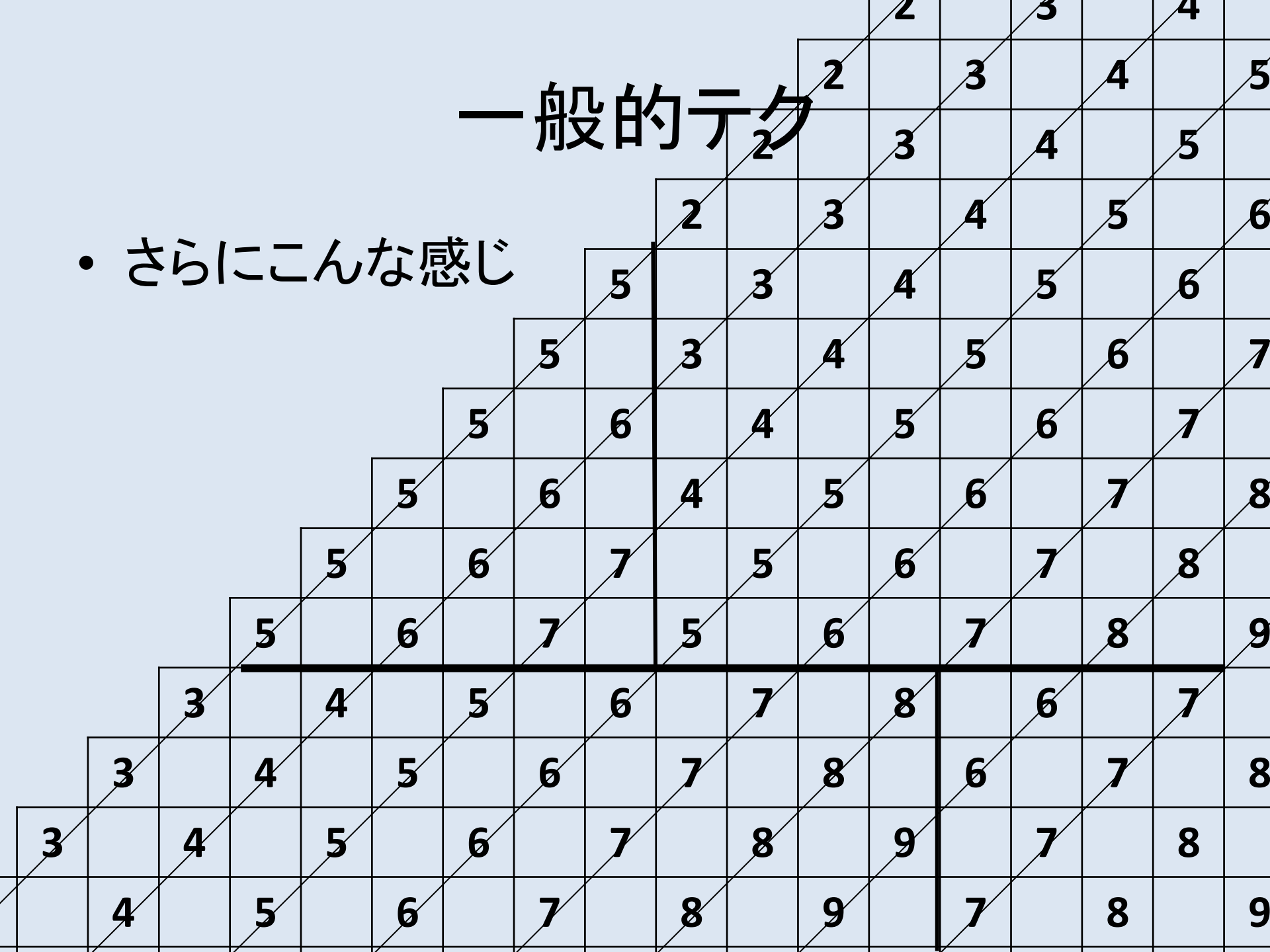
一般的テク

- こんな感じ(諸事情により拡大)



一般的テク

- さらにこんな感じ



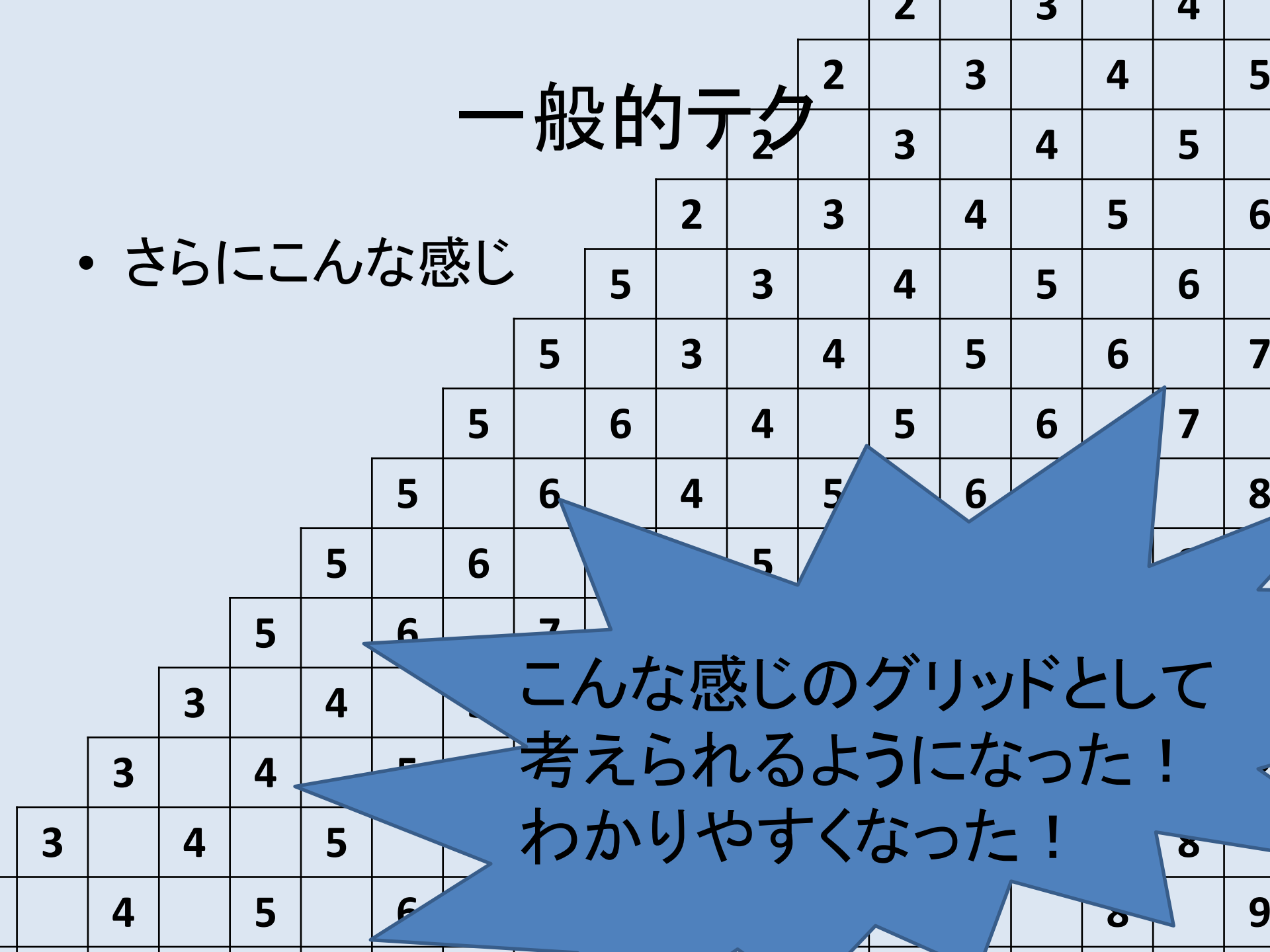
一般的テク

- さらにこんな感じ

											2		3		4		
										2		3		4		5	
									2		3		4		5		
								2		3		4		5		6	
							5		3		4		5		6		
						5		3		4		5		6		7	
					5		6		4		5		6		7		
				5		6		4		5		6		7		8	
			5		6		7		5		6		7		8		
		5		6		7		5		6		7		8		9	
	3		4		5		6		7		8		6		7		
3		4		5		6		7		8		6		7		8	
3		4		5		6		7		8		9		7		8	
	4		5		6		7		8		9		7		8		9

一般的テク

- さらにこんな感じ



一般的テク

初期状態では
こんな感じ

									0
								0	
							0		1
						0		1	
					0		1		2
				0		1		2	
			0		1		2		3
		0		1		2		3	
	0		1		2		3		4
0		1		2		3		4	

一般的テク

この領域が隆起すると?

									0
								0	
							0		1
						0		1	
					0		1		2
				0		1		2	
			0		1		2		3
		0		1		2		3	
	0		1		2		3		4
0		1		2		3		4	

一般的テク

この領域が隆起すると?

45度回したので、隆起の向きはこっち

									0
								0	
							0		1
						0		1	
					0		1		2
				0		1		2	
			0		1		2		3
		0		1		2		3	
	0		1		2		3		4
0		1		2		3		4	

一般的テク

こうなる

※高さ方向1の隆起はこのグリッドでは2マス分の動きであることに注意

							0		1
						0		1	
					0		1		2
				0		1		2	
			0		1		2		3
		0		1		2		3	
			0		1		2		3
		0		1		2		3	
	0		1		2		3		4
0		1		2		3		4	

一般的テク

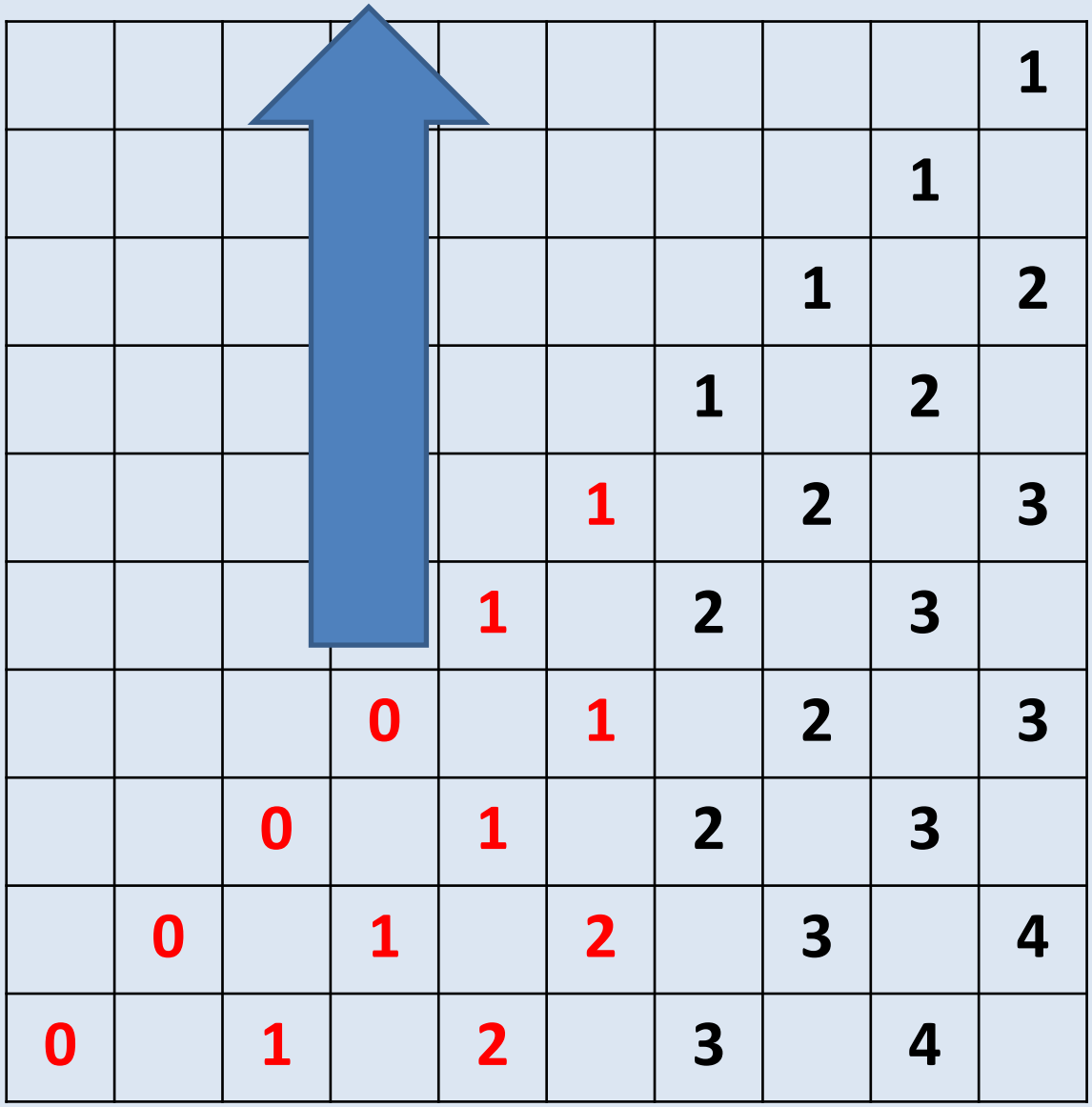
風化するとこうなる

									1
								1	
							1		2
						1		2	
					1		2		3
				1		2		3	
			0		1		2		3
		0		1		2		3	
	0		1		2		3		4
0		1		2		3		4	

一般的テク

こっちが隆起すると?

今度はこっち向き



一般的テク

こうなる

									1
								1	
					1		1		2
				1		1		2	
			0		1		2		3
		0		1		2		3	
	0		1		2		2		3
0		1		2		2		3	
	1		2		3		3		4
1		2		3		3		4	

一般的テク

風化するとこうなる

									1
								1	
							1		2
						1		2	
					1		2		3
				1		2		3	
			1		2		2		3
		1		2		2		3	
	1		2		3		3		4
1		2		3		3		4	

一般的テク

- 「ななめ45度にずれる」という操作が、グリッドの縦横への平行移動に書き換えられた！
- わかりやすくなった

小課題1(18点)

- $N \leq 100, Q \leq 100, (\text{隆起の高さ}) = 1$
- この表を作って、クエリを1個ずつ愚直にシミュレートし、最後の地表(すなわちこのグリッドの対角線)の状態を出力すればよい
  - $O(N^2)$個のマスだけ覚えておけば十分

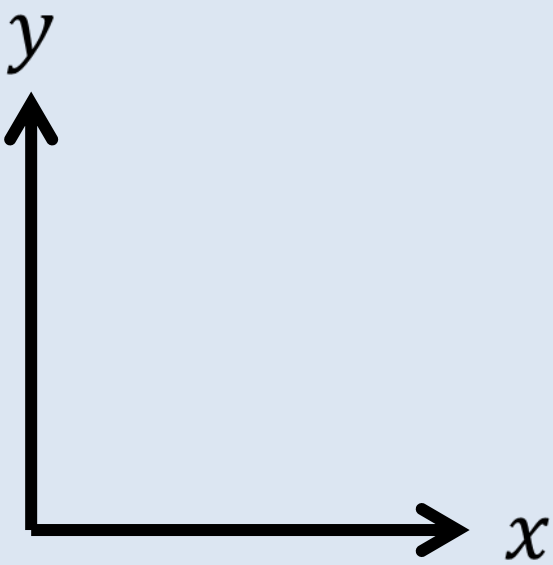
- 一回の隆起あたり$O(N^2)$で処理できる
- $O(N^2Q)$となり、18点が得られる

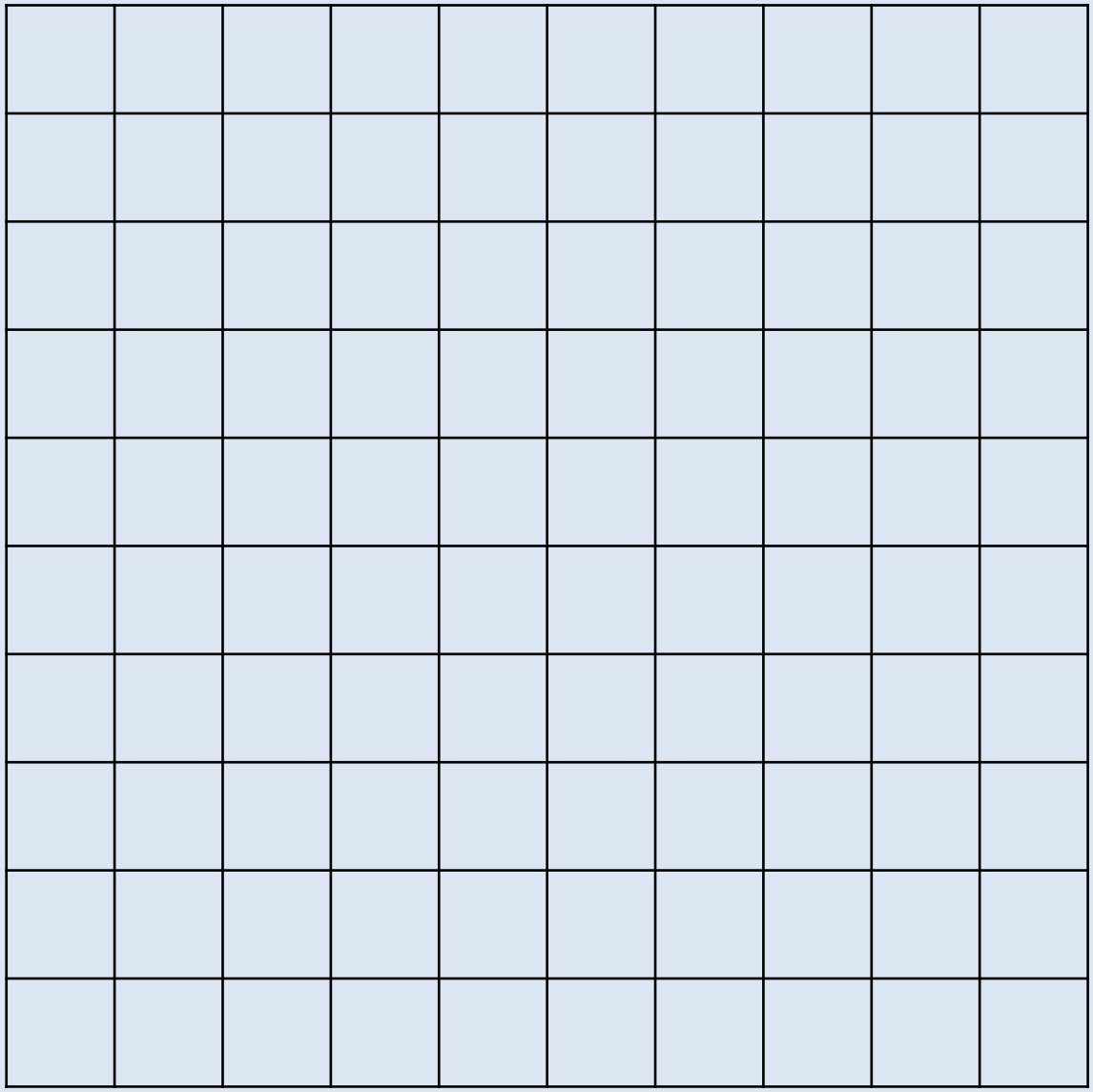
発想

- クエリを後ろから見る
- すると以下のような問題に書き換えられる
- 「沈降がQ回起こる。地表の座標1~Nの点はQ回の沈降のあとどこに行くか?」

発想

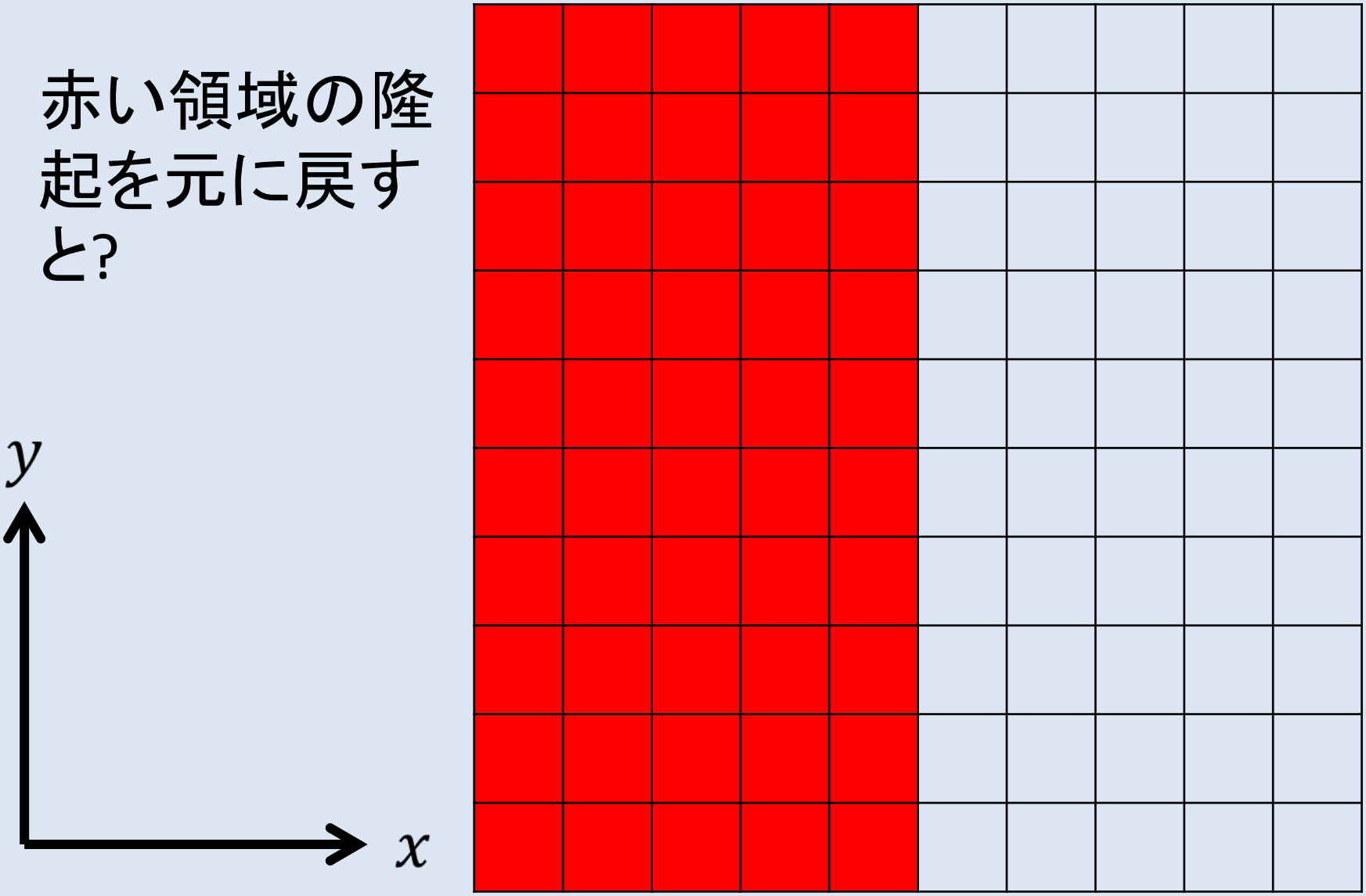
とりあえず、x軸とy軸をこのように取っておく





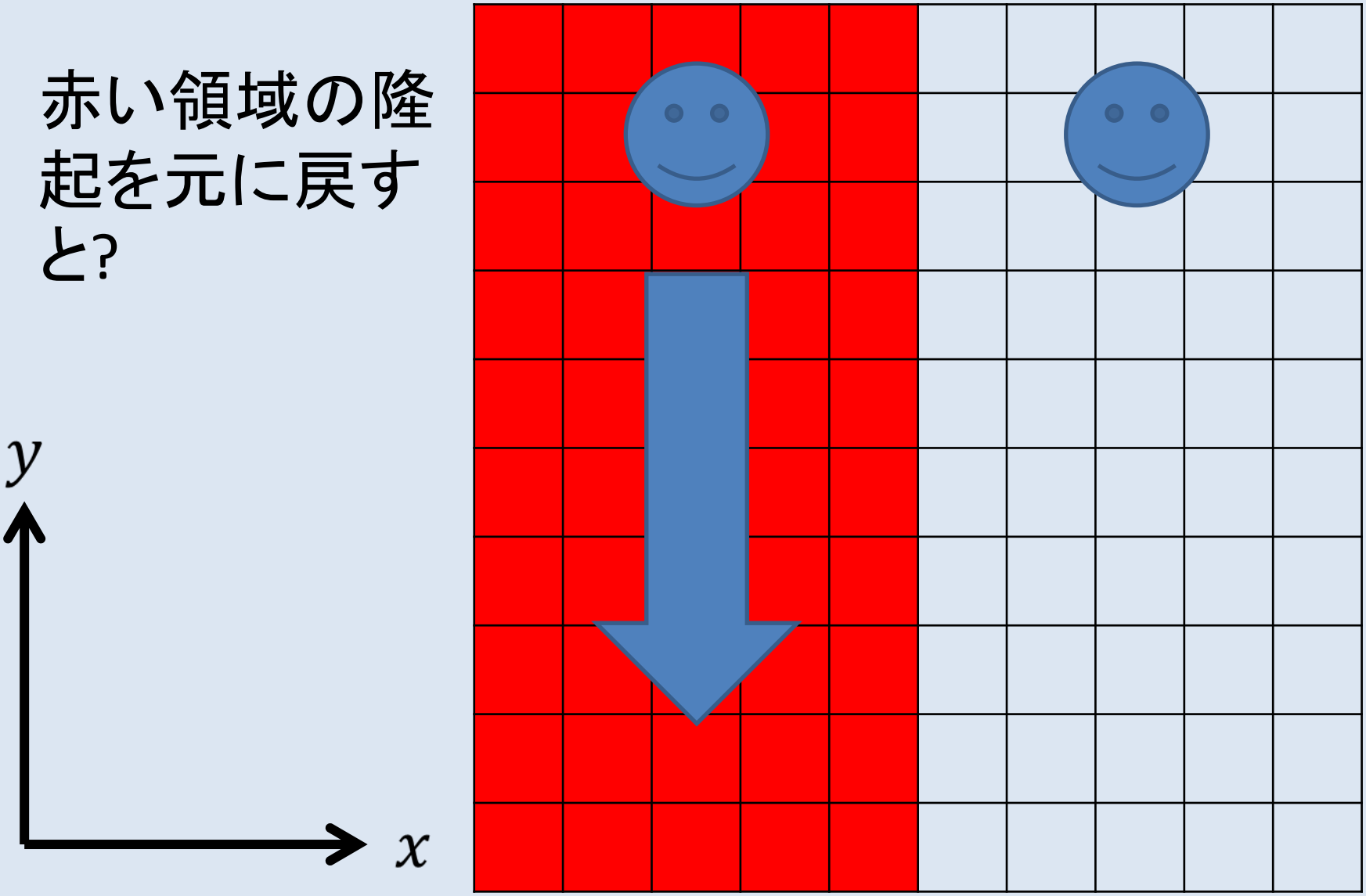
発想

赤い領域の隆起を元に戻すと?



発想

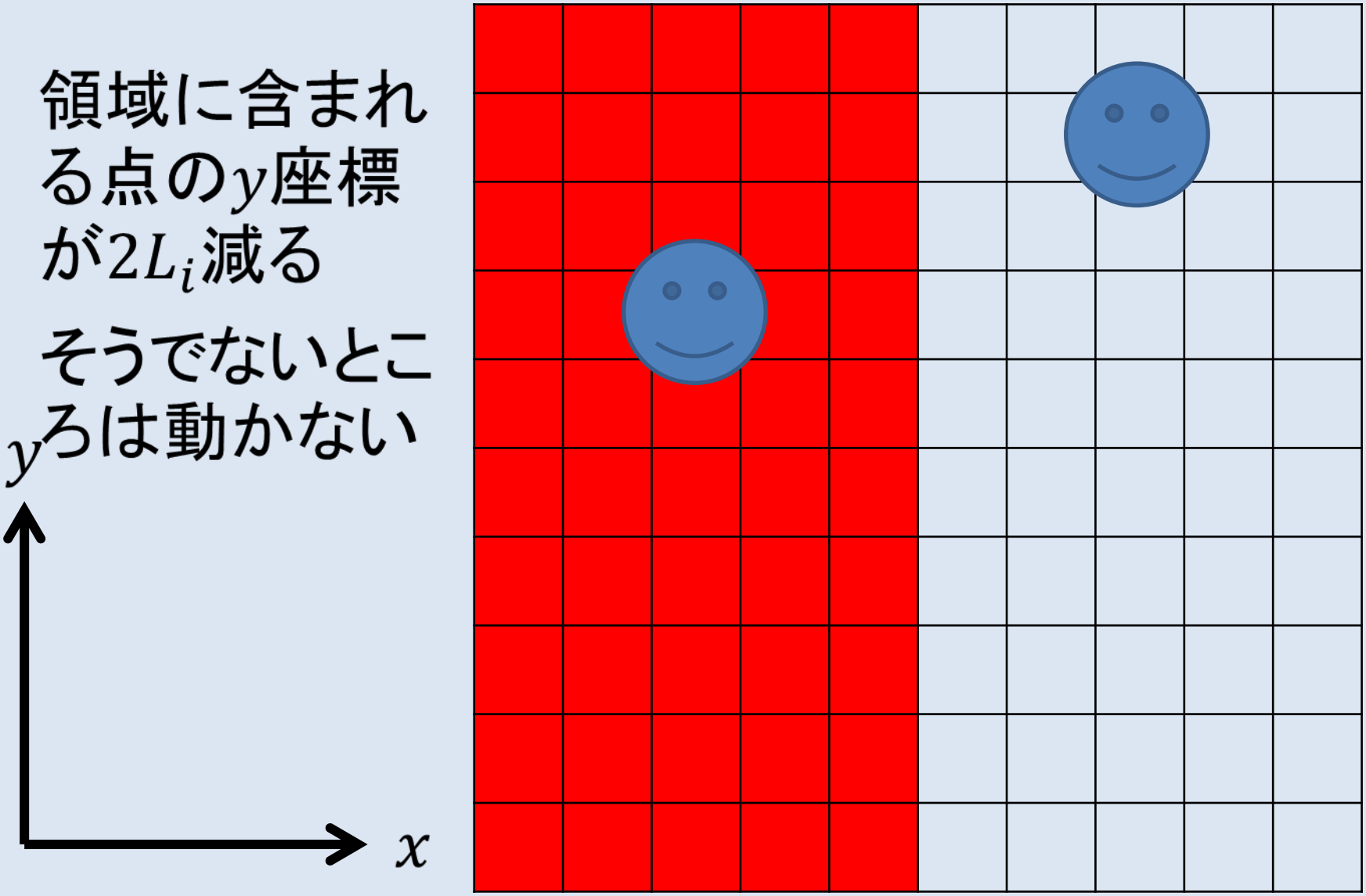
赤い領域の隆起を元に戻すと?



発想

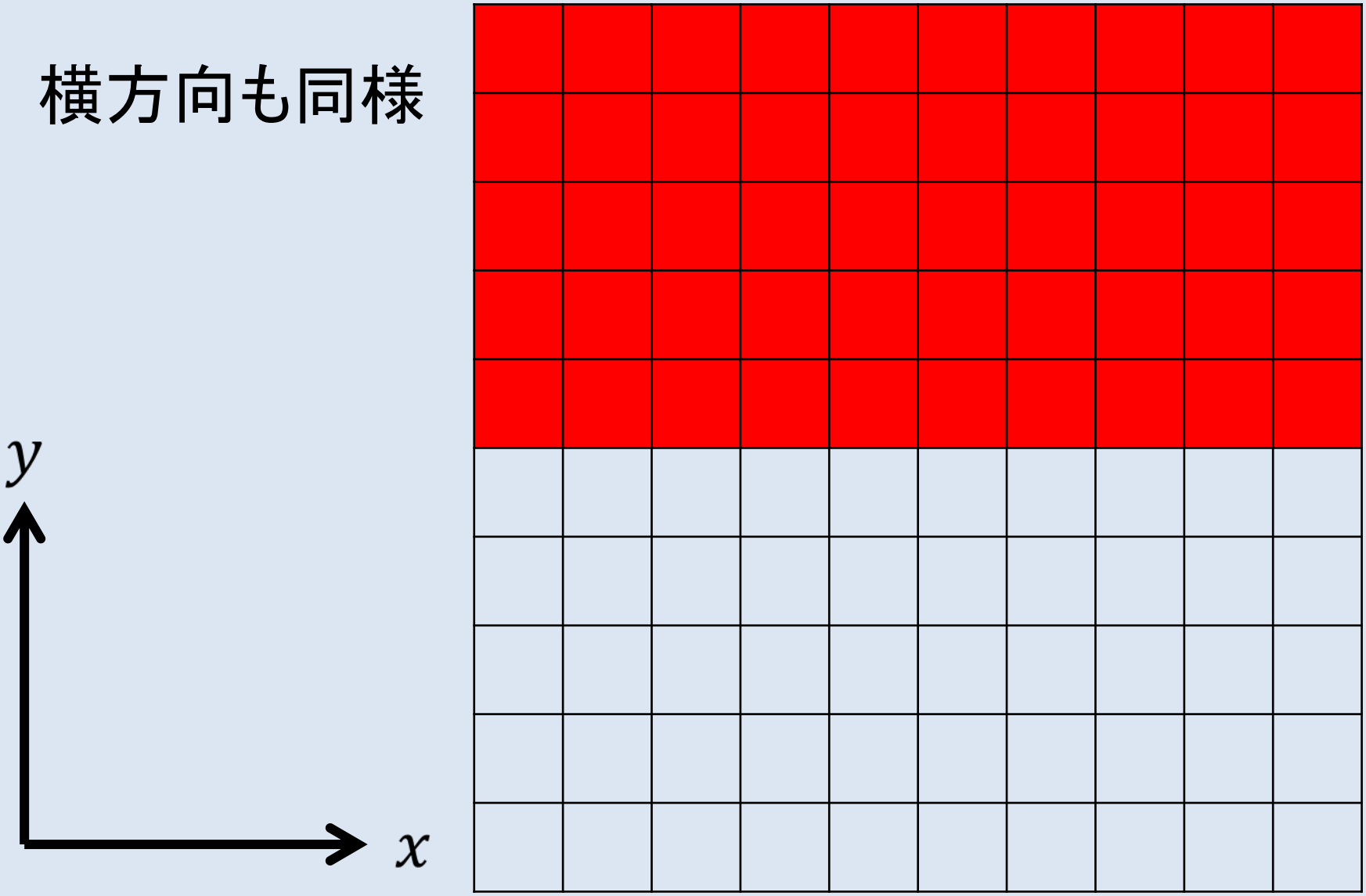
領域に含まれる点のy座標が$2L_i$減る

そうでないところは動かない



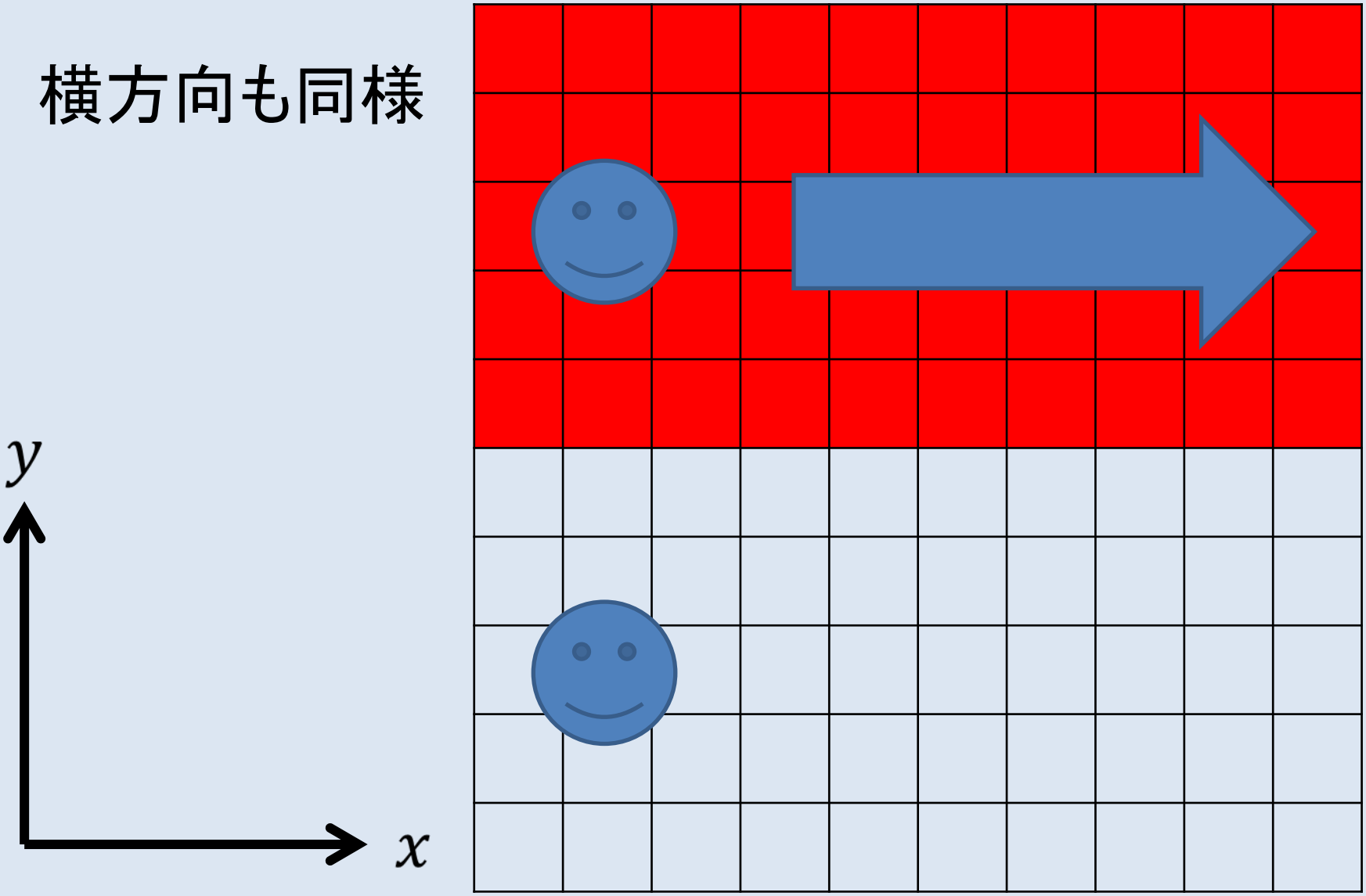
発想

横方向も同様



発想

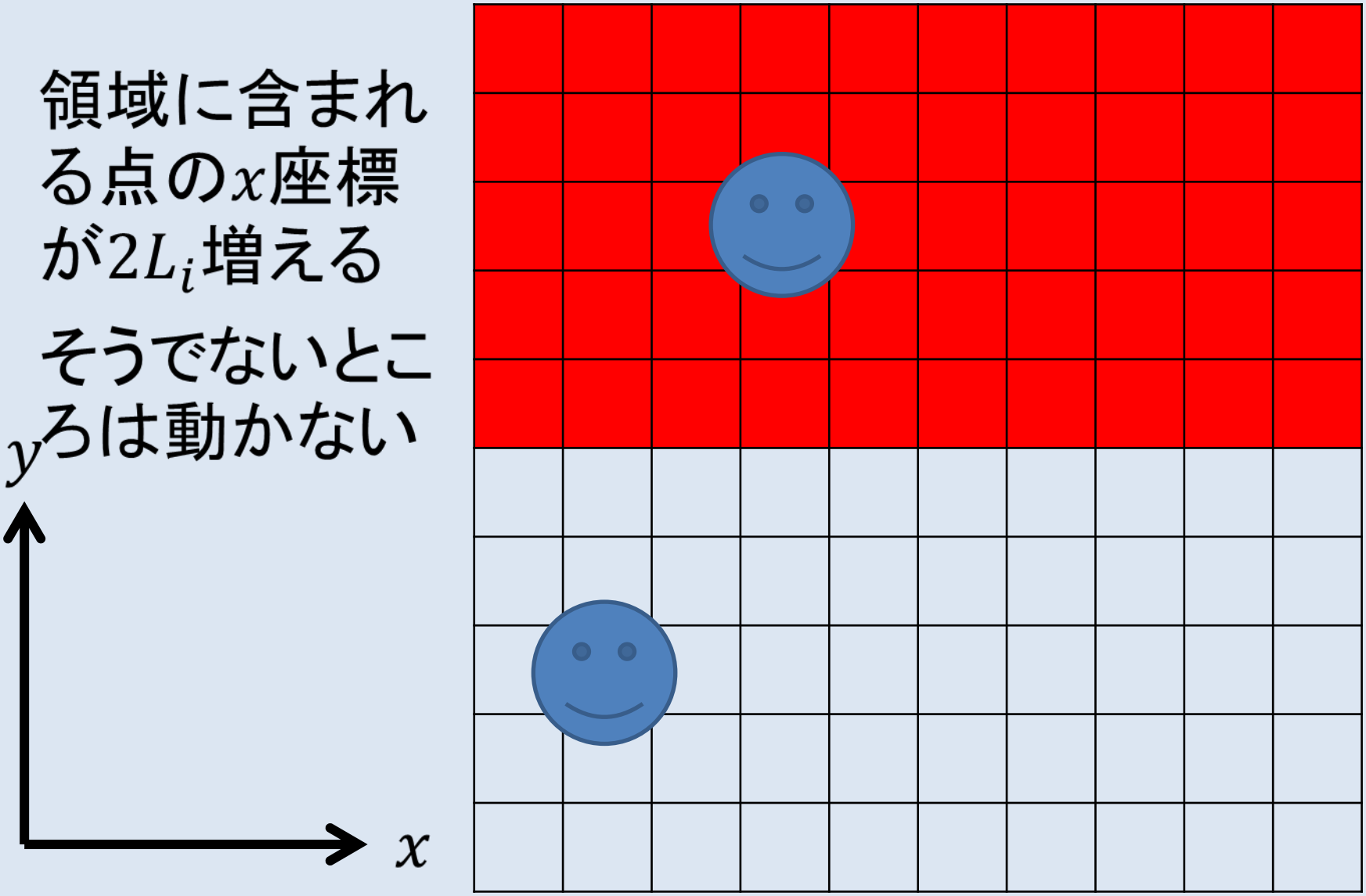
横方向も同様



発想

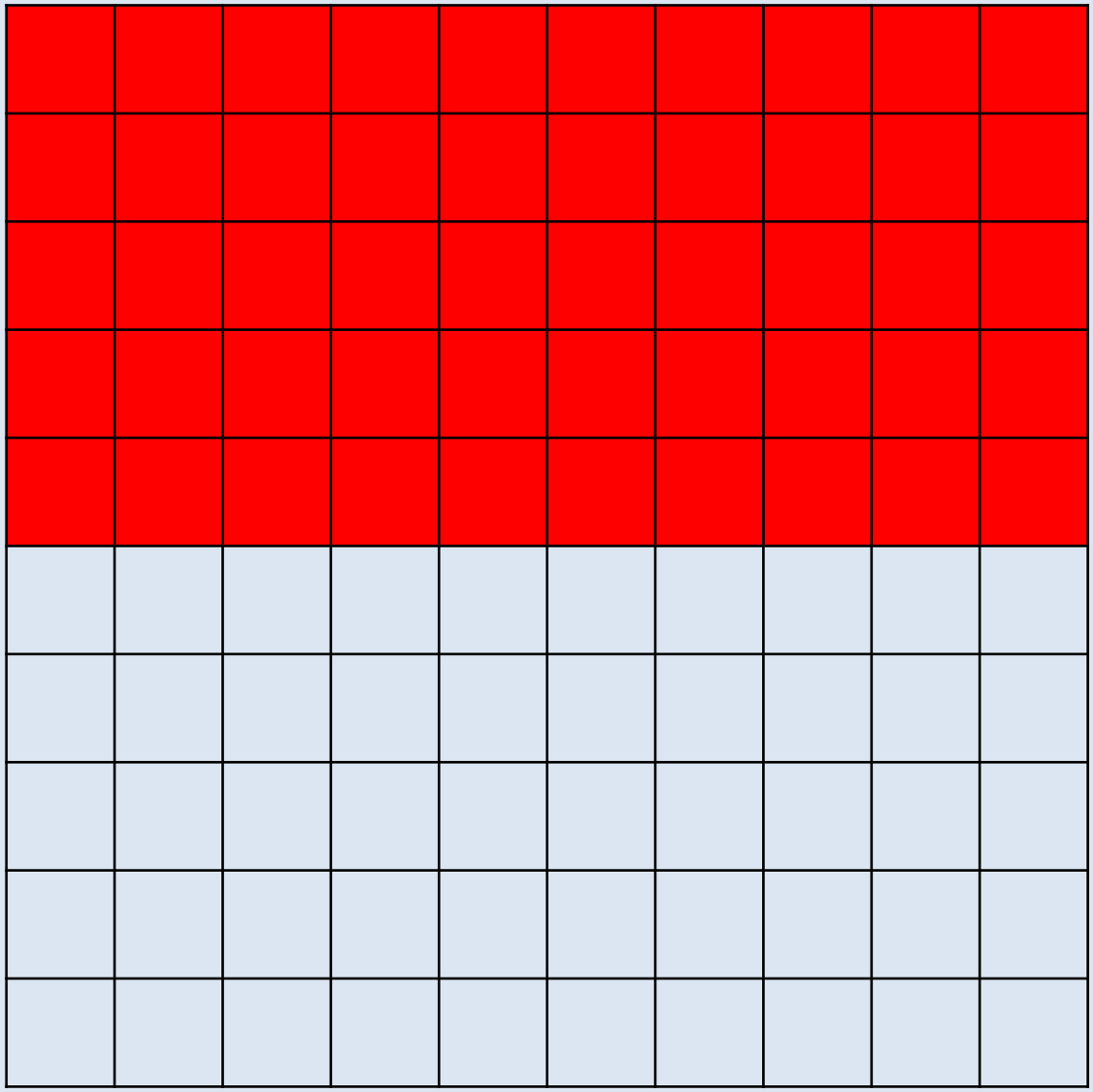
領域に含まれる点のx座標が$2L_i$増える

そうでないところは動かない



ところで

なにかに似てる



閑話休題

モナコ

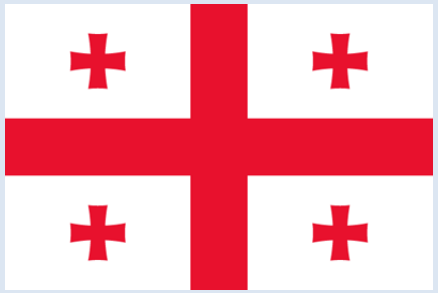
ポーランド

インドネシア

閑話休題

オーストリア

閑話休題

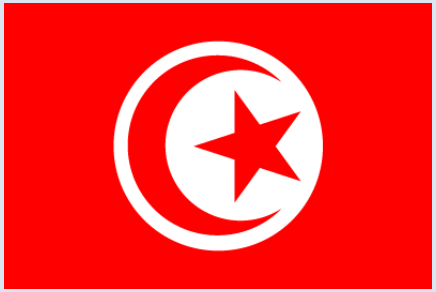


ジョージア

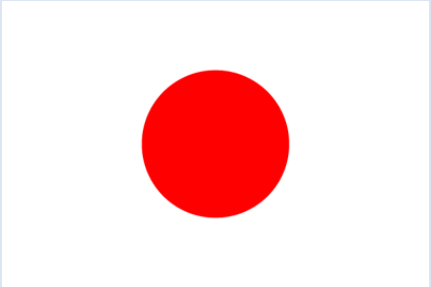
シンガポール

トルコ

バーレーン

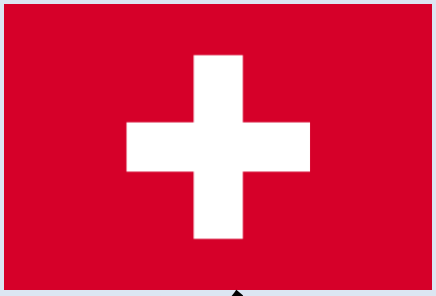


チュニジア

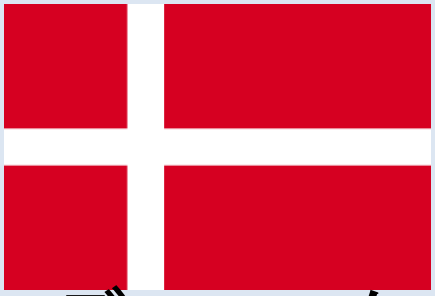


日本

カナダ



スイス



デンマーク

閑話休題

- 赤と白の国旗は意外と多い

- ちなみにまだこのスライドに表れていない赤と白の国旗の国(地域ではない)が一つあります
- 当てましょう

発想まとめ

- 隆起する領域に含まれる点が隆起の前にどこにあったかを考えると、

- 横向きの隆起の場合、その点のx座標を適切に変えた点
- 縦向きの隆起の場合、その点のy座標を適切に変えた点
- となることが分かった

小課題2(18+16点)

- $N \leq 3000, Q \leq 3000$
- 点ごとに独立に解を求める
- クエリを後ろから見て、各クエリごとにその点の現在の位置が隆起した領域に含まれるか判定し、含まれるなら座標を動かす
- 各点に対し、クエリごとに$O(1)$で処理ができるので、全体で$O(NQ)$となり、小課題1と併せて34点が得られる

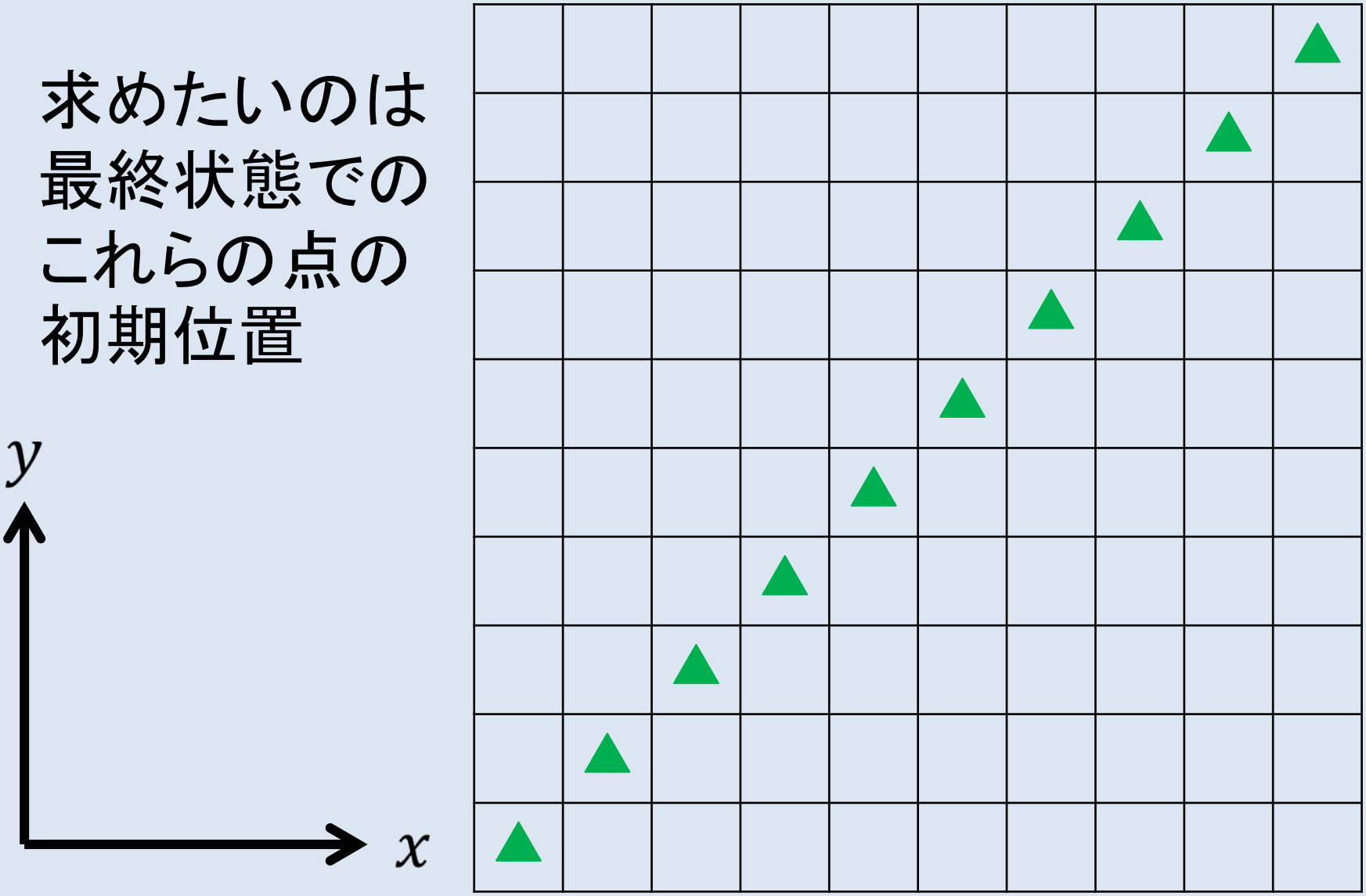
さらに考察

- 横向きの隆起の場合、その点のx座標を適切に変えた点
- 縦向きの隆起の場合、その点のy座標を適切に変えた点

- これをもう少し真面目に考えてみる

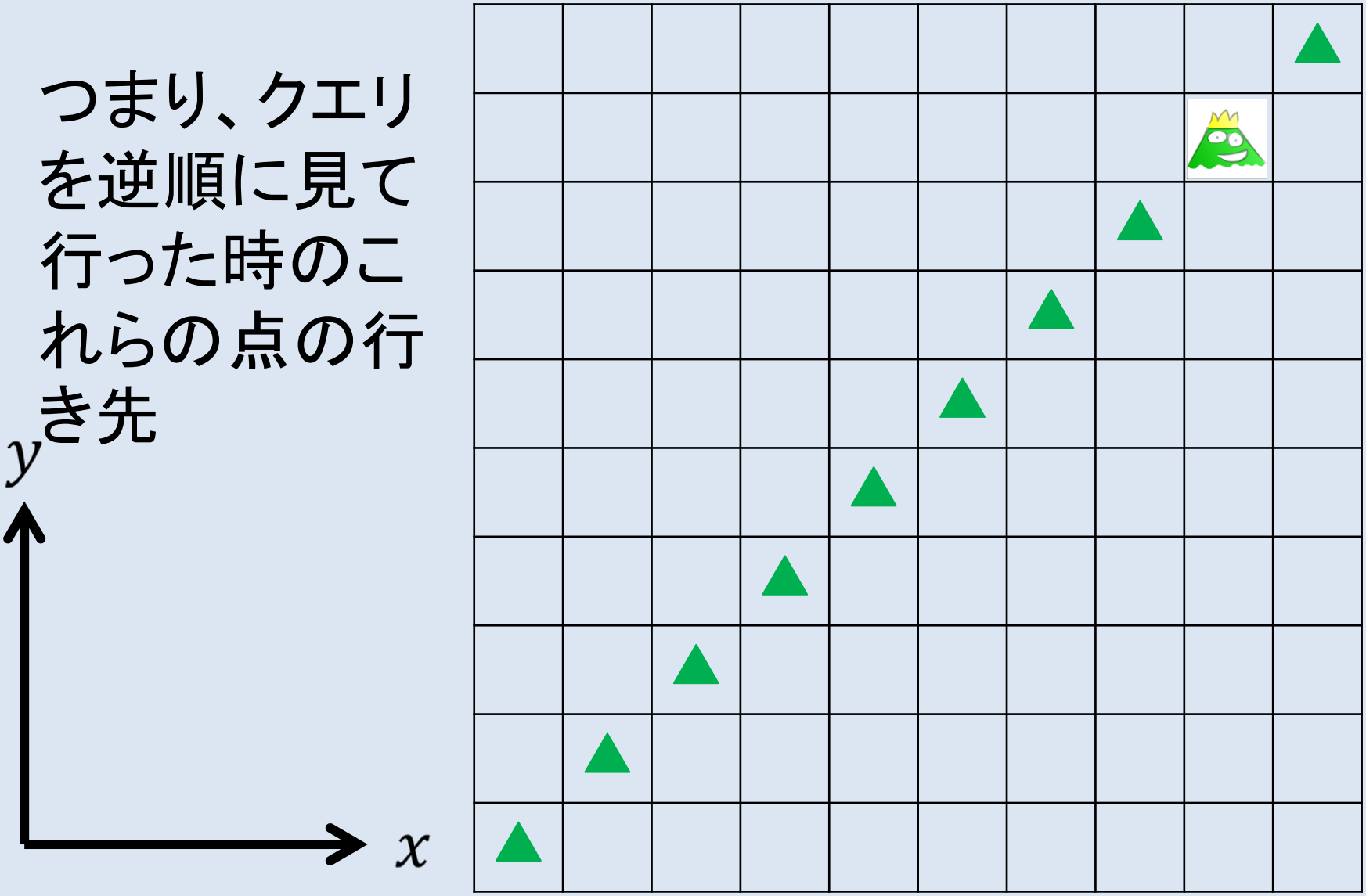
さらに考察

求めたいのは
最終状態での
これらの点の
初期位置



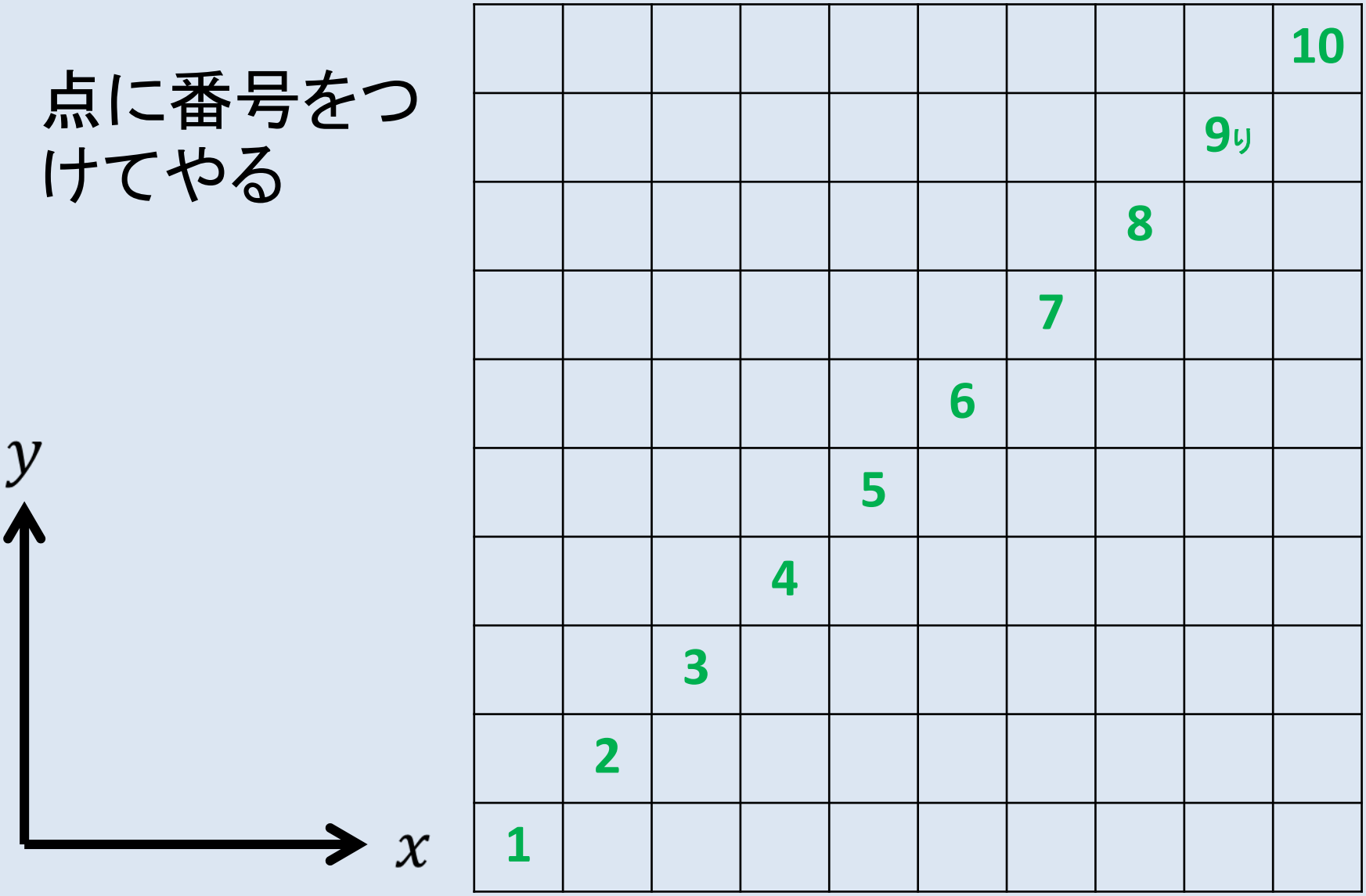
さらに考察

つまり、クエリを逆順に見て行った時のこれらの点の行き先



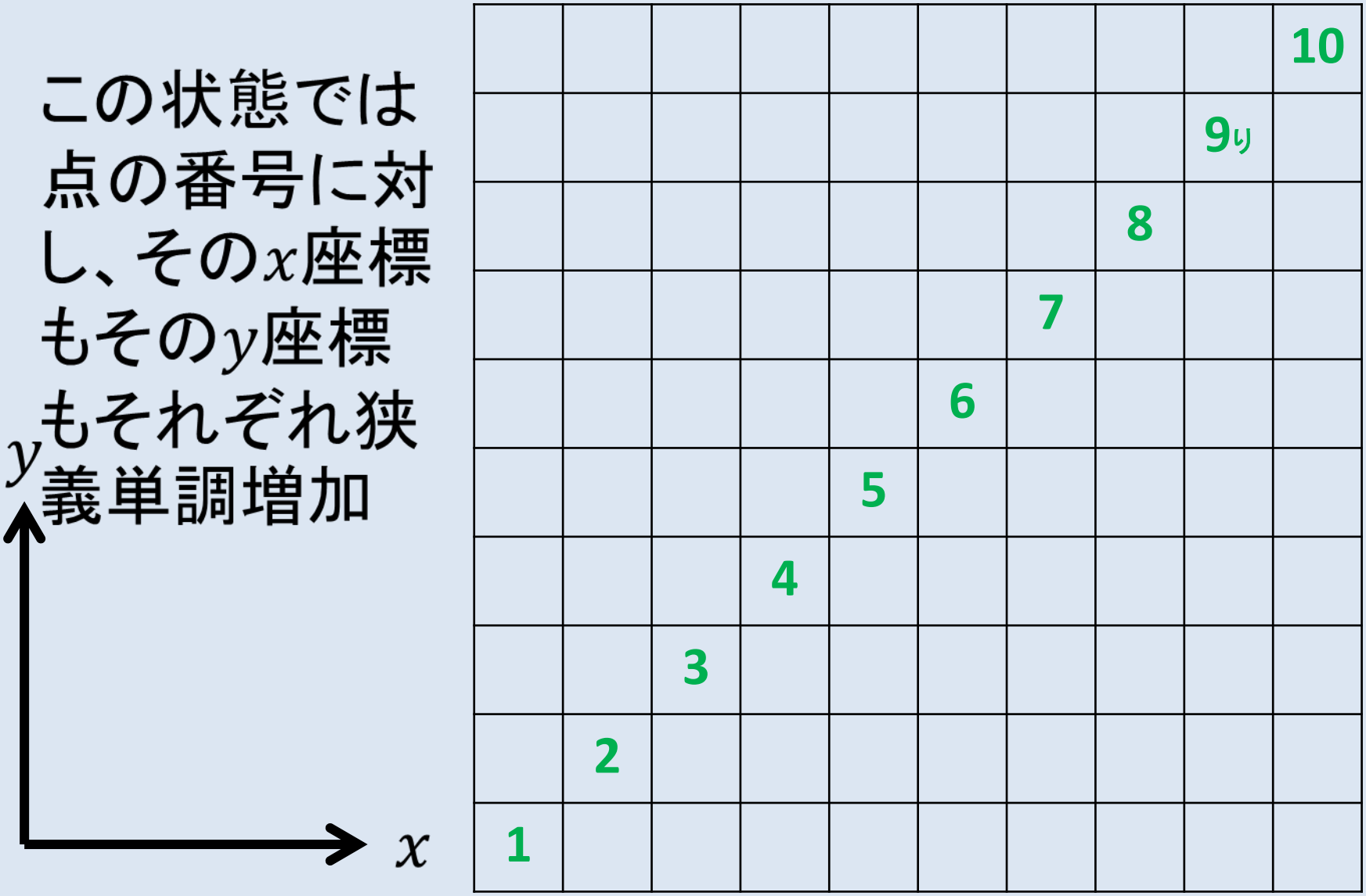
さらに考察

点に番号をつけてやる



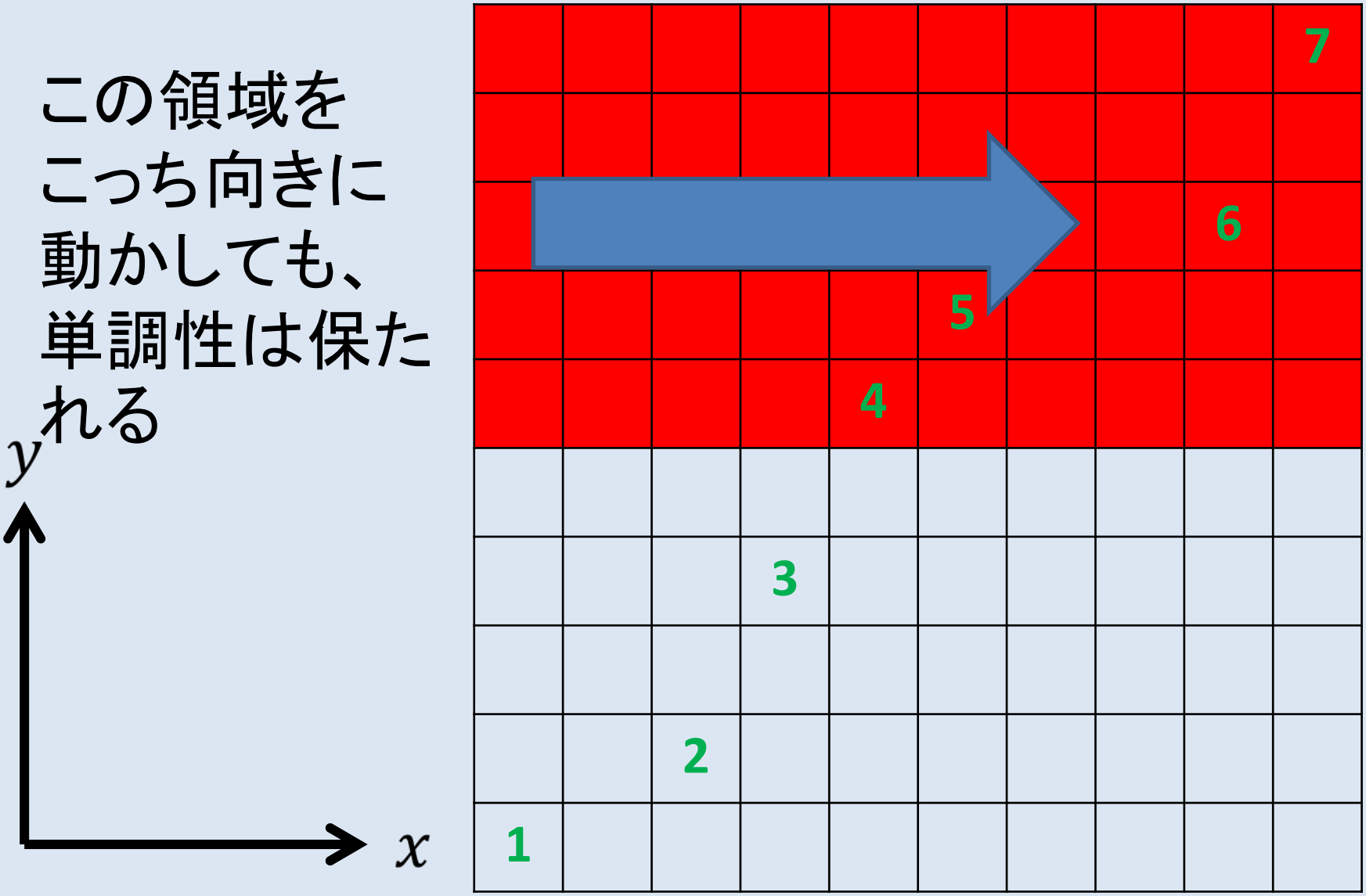
さらに考察

この状態では
点の番号に対
し、そのx座標
もそのy座標
もそれぞれ狭
義単調増加



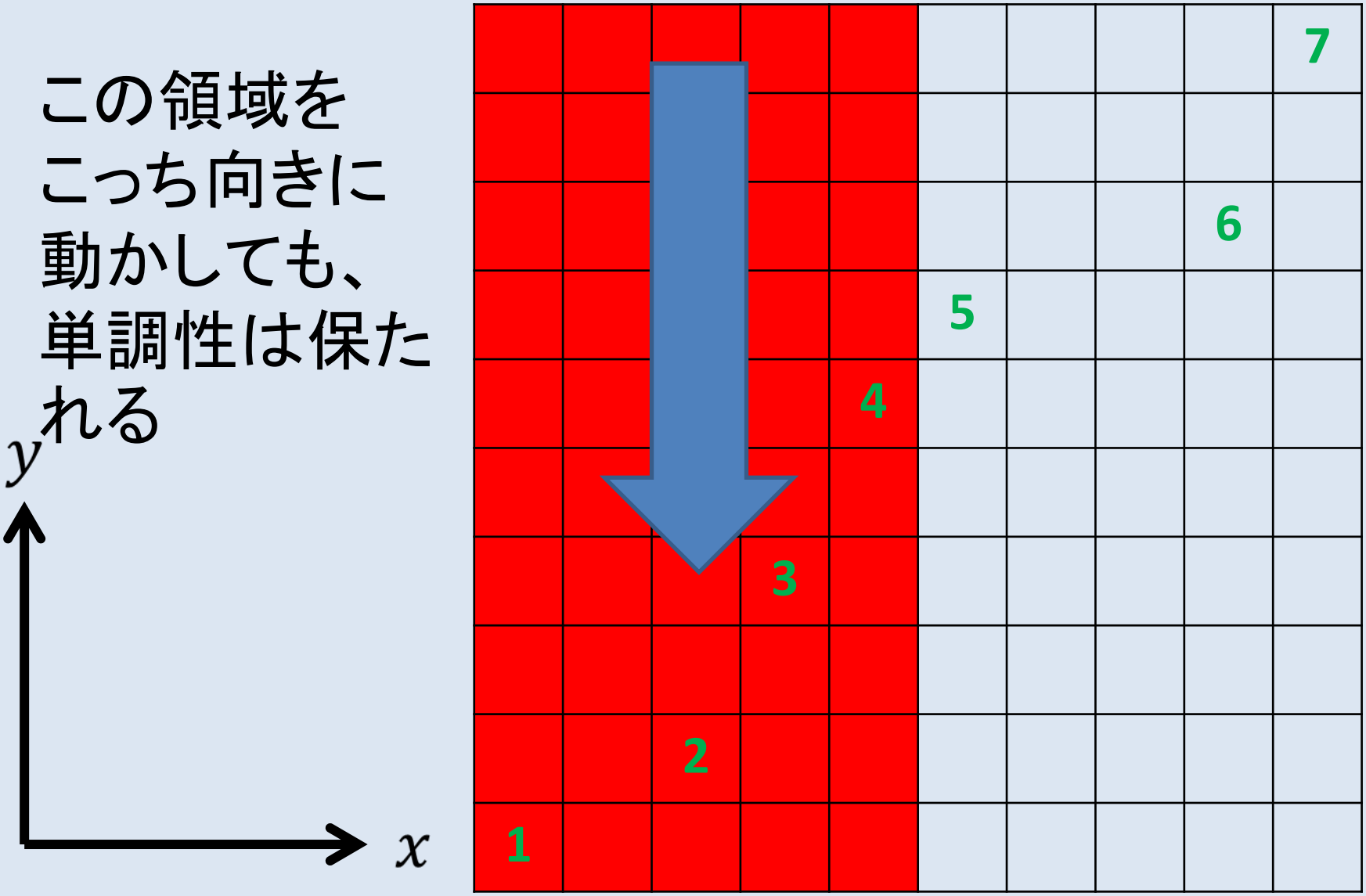
さらに考察

この領域を
こっち向きに
動かしても、
単調性は保たれる



さらに考察

この領域を
こっち向きに
動かしても、
単調性は保たれる

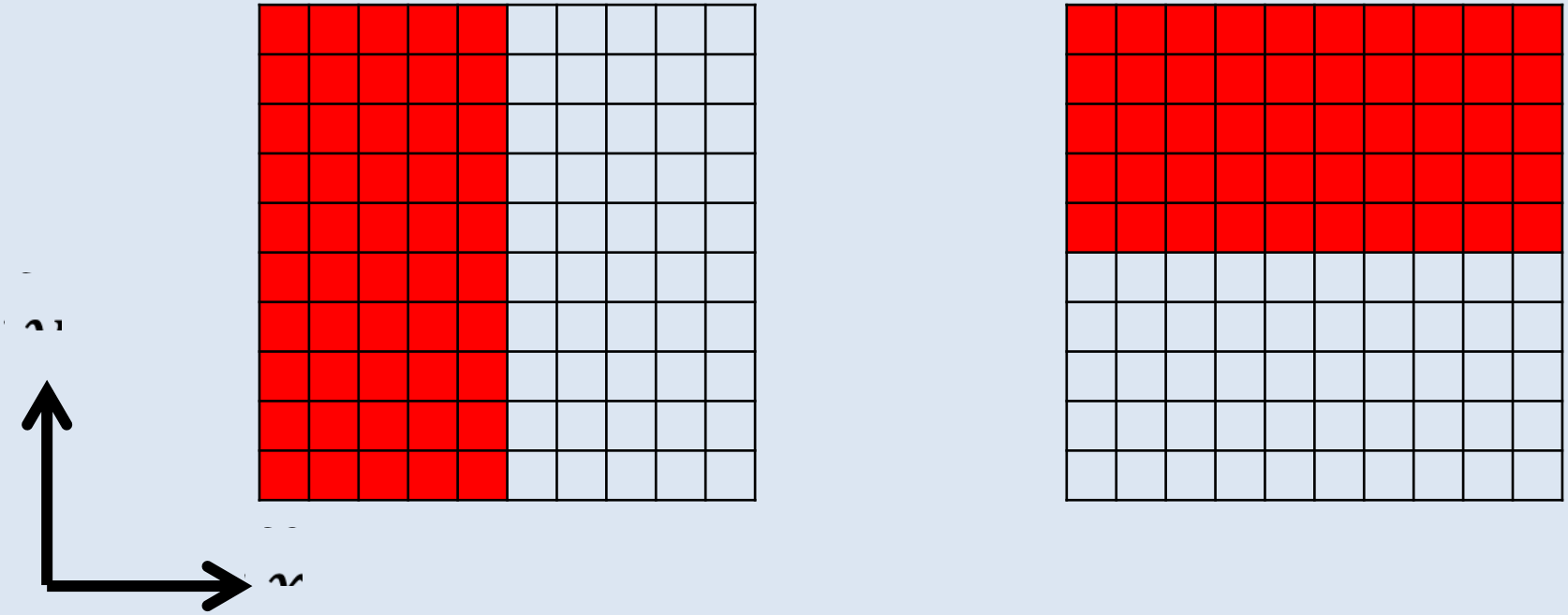


さらに考察

- 結局、クエリを逆順に処理するときに、点の番号に対するそのx座標の単調性も、そのy座標の単調性も常に保たれている

さらに考察

- 一方、隆起をした領域は、x座標がクエリで与えられた値以下の領域または、y座標がクエリで与えられた値以上の領域



さらに考察

- 単調性より各クエリごとに、隆起をした領域にある点は、番号がある値以下の点すべてまたは、番号がある値以上の点すべてとなる
  - 特に、番号は連続した区間となる
- さらに、クエリの処理は、これらの点のx座標あるいはy座標に、$2L_i$を足すもしくは引くことであった

さらに考察

- ということは、各点のx座標を並べた列とy座標を並べた列を用意しておけば、このクエリは単に区間への一定値の加算で表現できる

さらに考察

- 残った問題は、「どの区間に値を足せばいいか」を求めること
- これもやはり単調性より、「x座標がクエリで与えられた値以下の最後の点」または、「y座標がクエリで与えられた値以上の最初の点」を求めれば求まる

小課題3(18+16+66点)

- $N \leq 200000, Q \leq 200000$
- ここまでの考察より、
- 「区間への一定値の加算」
- 「狭義単調増加な数列に対して、ある値以下の最後の値または、ある値以上の最初の値を求める」
- というクエリが高速に処理できればいい

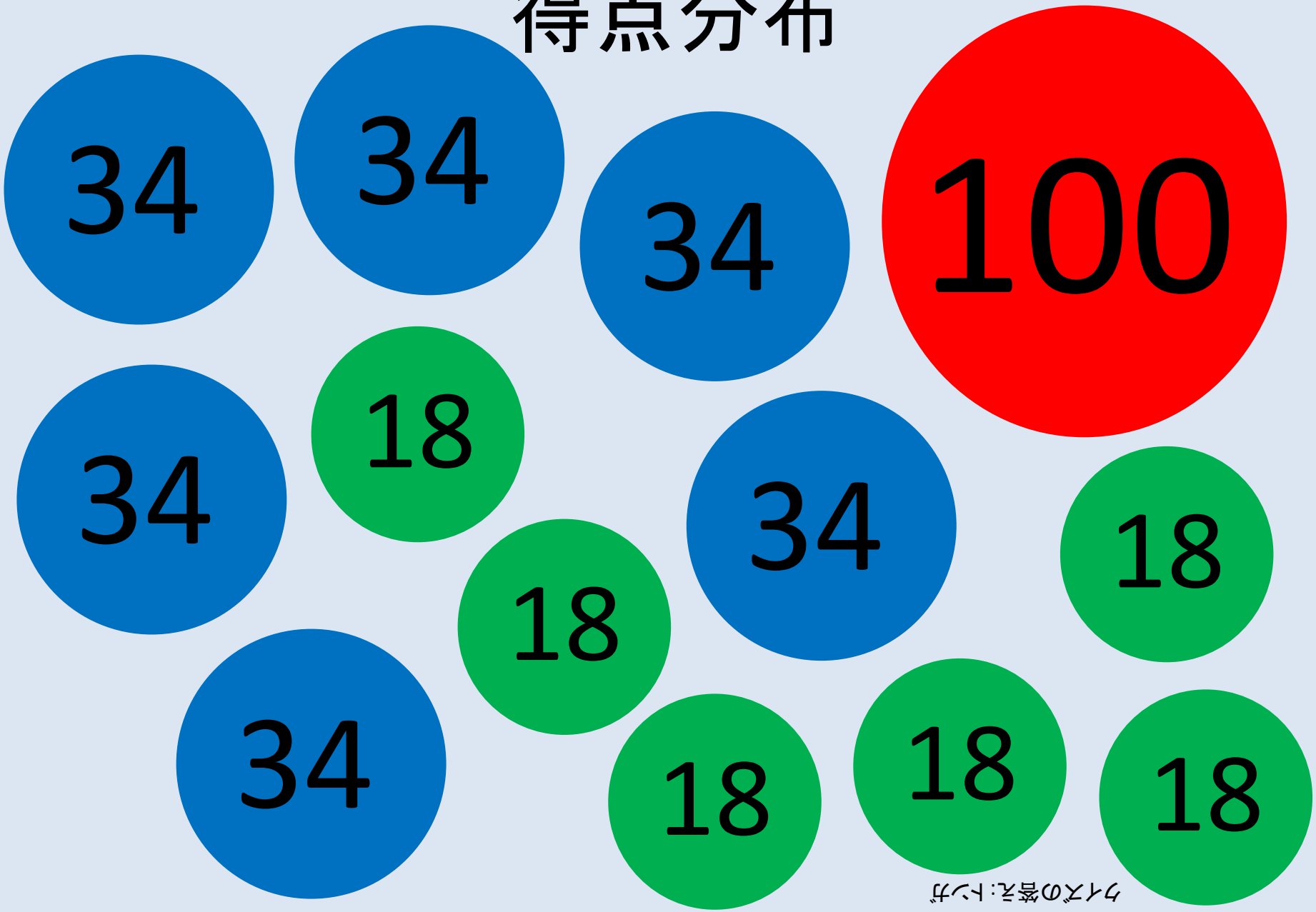
小課題3(18+16+66点)

- BITもしくはsegment treeを使う
- 区間への一定値の加算は、BITの累積和として持たせるか、segment treeの遅延更新でできる
- 境界値を求めるのは、単調性より、これらのデータ構造から値を取り出して二分探索をすると求まる

小課題3(18+16+66点)

- 最初のクエリは$O(\log N)$で処理できる
- 二つ目のクエリは$O(\log^2 N)$で処理できる
  - 二分探索をsegment treeの中に入れてやれば$O(\log N)$でもできる
- 以上より、この問題が$O(Q\log N)$または$O(Q\log^2 N)$で解けた
- 小課題1,2と併せ、満点が得られる

得点分布



クイズの答え：トンガ