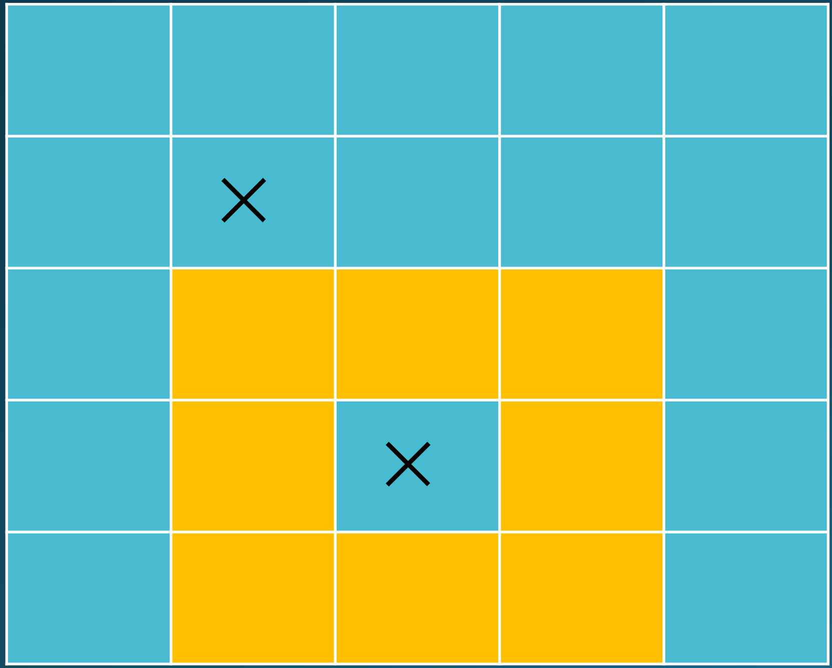
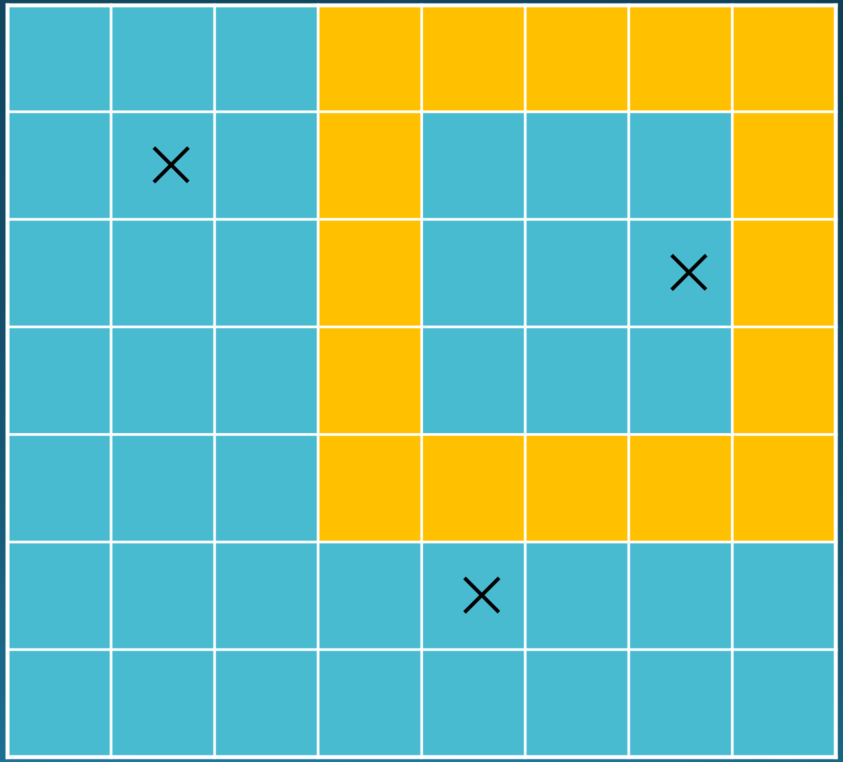
本選問 5 Rampart

問題概要

- グリッドが与えられます
- 幅1の（十分大きい）正方形の枠の置き方は何通り？





小課題 1

- 置けない位置を無視すると，調べる位置は O(HW min(H, W)) 個
- それぞれの位置について，O(1) でチェックできればよい

- 累積和を使う

累積和

- 詳しいやり方は参考書（蟻本など）を見てください
- 累積和を適切に使うと，前処理 O(HW) で， 2 次元配列のある長方形領域の値の和が O(1) で求められる
- 2 次元配列として，「その位置に城壁がなかったら 1, あったかもしれないなら 0」とした配列をとる
- 枠を決めたとき，枠の上の「そこには城壁がなかった」マスの数は O(1) で求められる

入出力例１

	×			
		×		

↑置けないマス

0	0	0	0	0
0	1	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0

↑配列

入出力例1

0	0	0	0	0
0	1	1	1	1
0	1	1	1	1
0	1	2	2	2
0	1	2	2	2

↑累積和

0	0	0	0	0
0	1	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0

↑配列

入出力例１

- →の橙色の部分に１が何個あるか知りたいとき

0	0	0	0	0
0	1	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0

入出力例1

- ↓の橙の部分の和から

0	0	0	0	0
0	1	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0

↓の赤の部分の和を引く

0	0	0	0	0
0	1	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0

小課題 1

- こうやると，枠を決めた時のチェックが O(1) でできる
- 前処理 O(HW)
- 調べる枠候補の数は O(HW min(H, W))
- 合計 O(HW min(H, W))

- 小課題 1 が通って 4 点が得られる

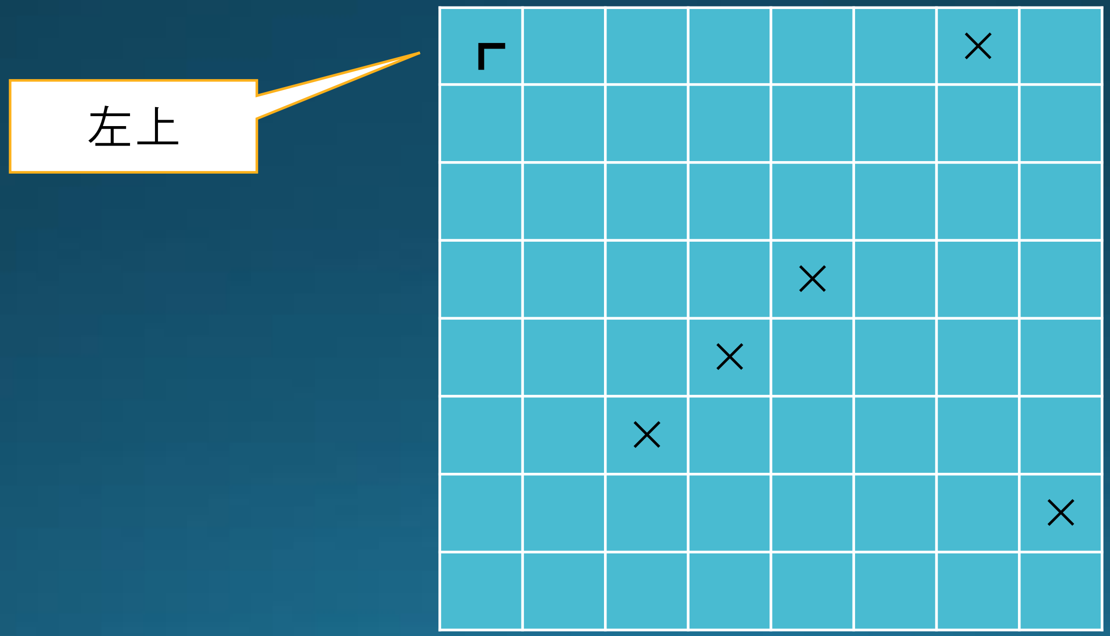
小課題 2

- P <= 10
- 「置けないマス」の数が異様に少ない

- ところで，置けないマスがまったく存在しなければ答えは簡単に求められる
- この小課題では，「だめな枠の数」を数えるとよさそう

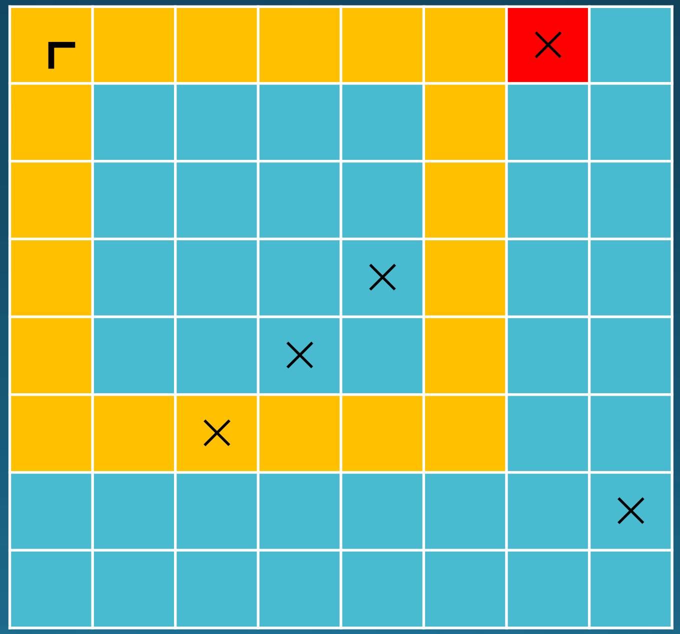
小課題 2

- 枠の左上を (a, b) に固定したときに，だめな枠が何個あるか数える



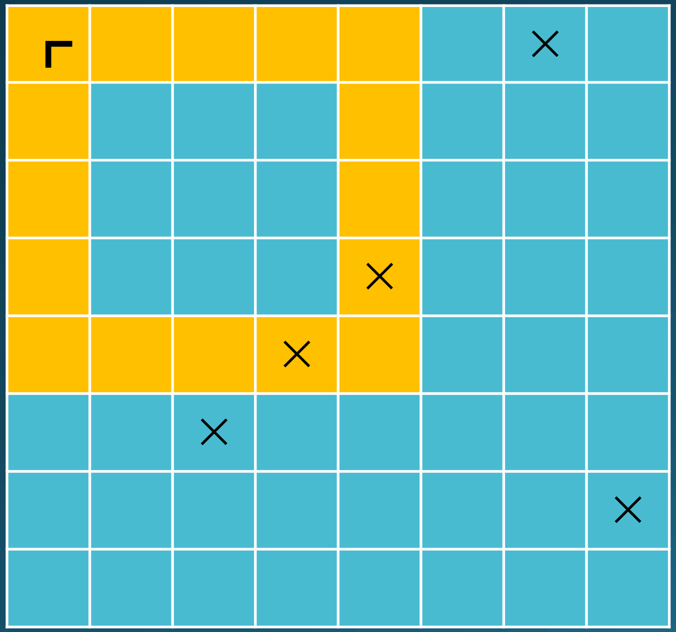
小課題 2

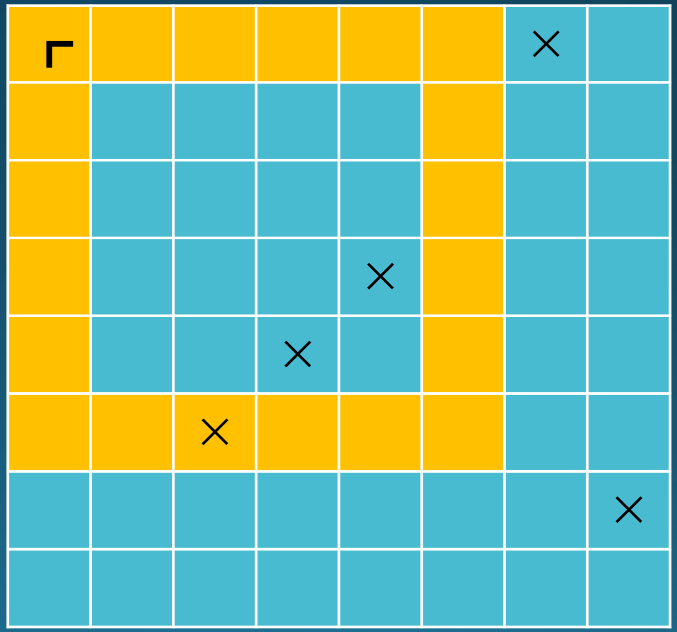
- 「左上」から，下や右にたどっていって，だめなマスにぶつかるところがあったらそこが枠の大きさの限界



小課題 2

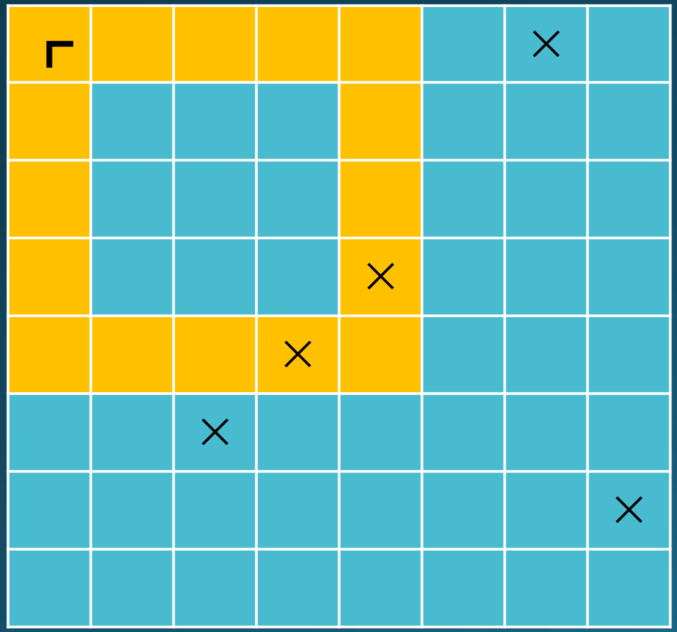
- それ以外で，右下にある「だめなマス」は，枠の大きさを動かすとちょうど 1 回だけぶつかる

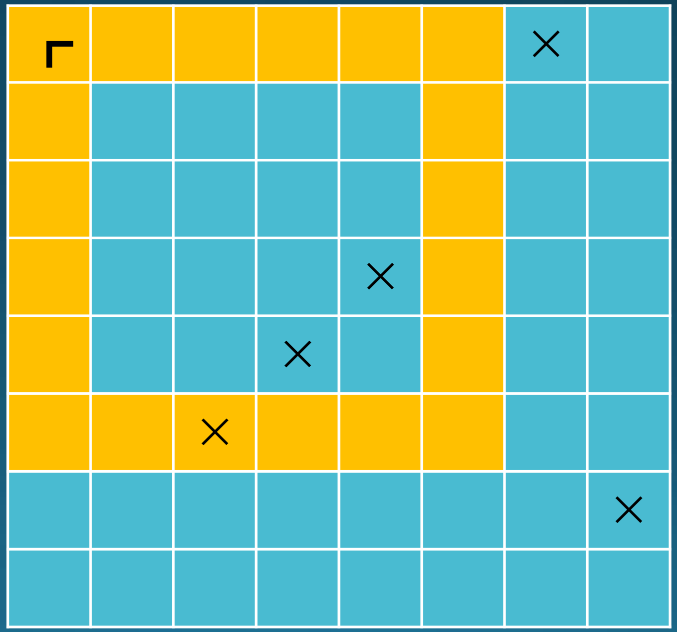




小課題 2

- ぶつかる大きさを列挙して，下限 (L) と上限 (左上から伸ばした辺が，壁かだめなマスにぶつかるまで) の間にあるような大きさが何種類あるか数える





小課題 2

- 同じ大きさでぶつかるようなマスが複数あるかもしれないので注意
- ソートなどによって重複を除く

- 左上の選び方 O(HW)
- だめな大きさを調べるのに O(P log P) (ソートの場合)
- O(HW P log P) で少しこわい (>_<) が余裕で通る
- 16 点が得られる
- P の大きさで場合分けして小課題 1 も解くと 20 点

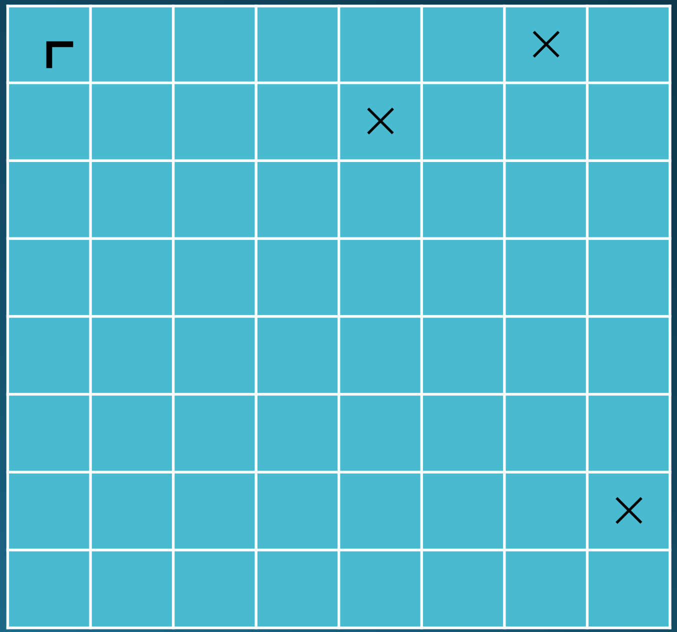
満点解法の前に

- 4000 * 4000 だと，答えは最大で 200 億以上になる
- TLE 10 秒とはいえ，答えをいちいち数え上げる余裕はない

- 複数の答えをまとめて数え上げる必要がある

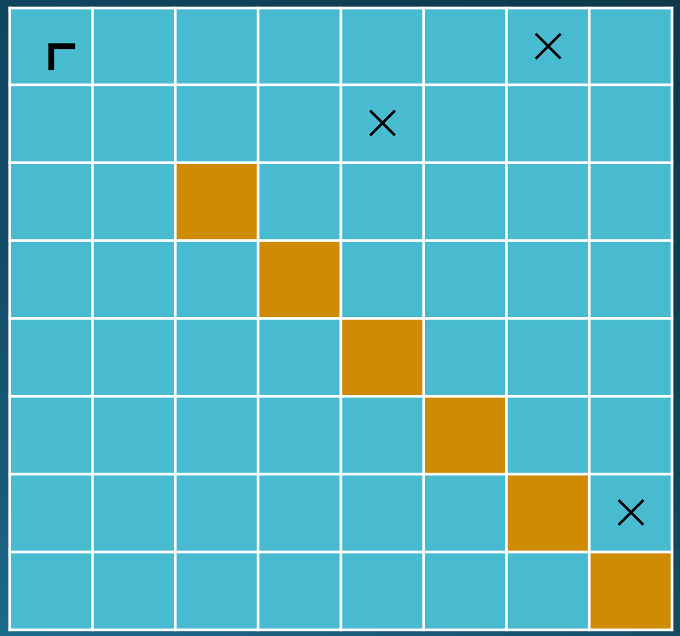
左上を固定

- 変な関係のものたちをいっぺんに数えるのは無理そう
- 左上を固定して，右下を動かして考える



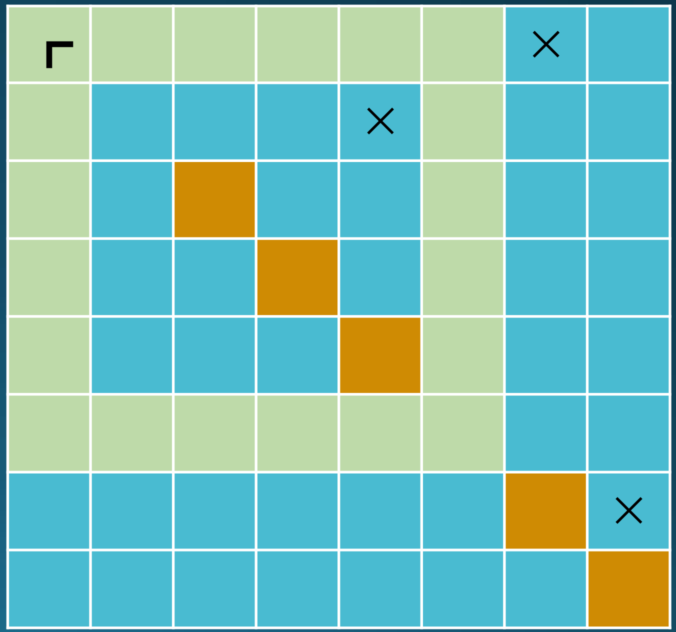
右下の動ける範囲

- 右下は，左上からななめ 45 度の線の上にある
  - 正方形なので当然
- L の大きさによって，あまり近くてはいけない



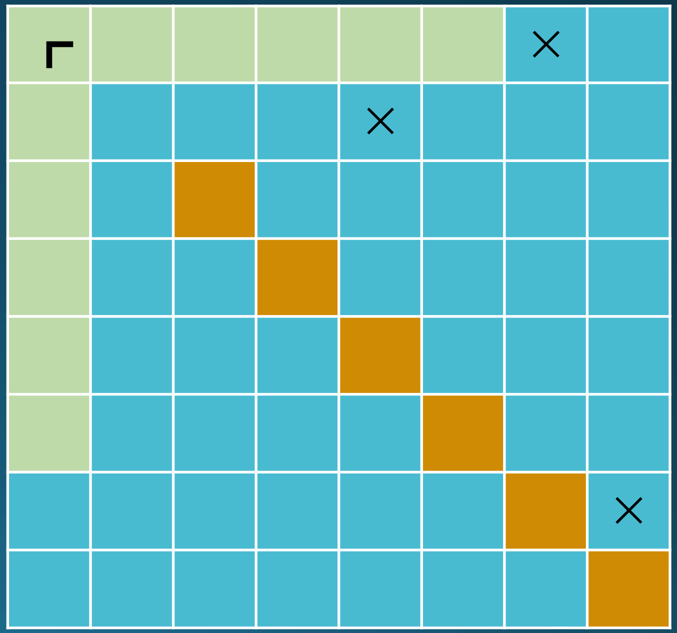
城壁を分けて考える？

- 城壁は枠型
- 左上から伸びる ┏型の部分と,
- 右下から伸びる┛ 型の部分とに分けて考えられる



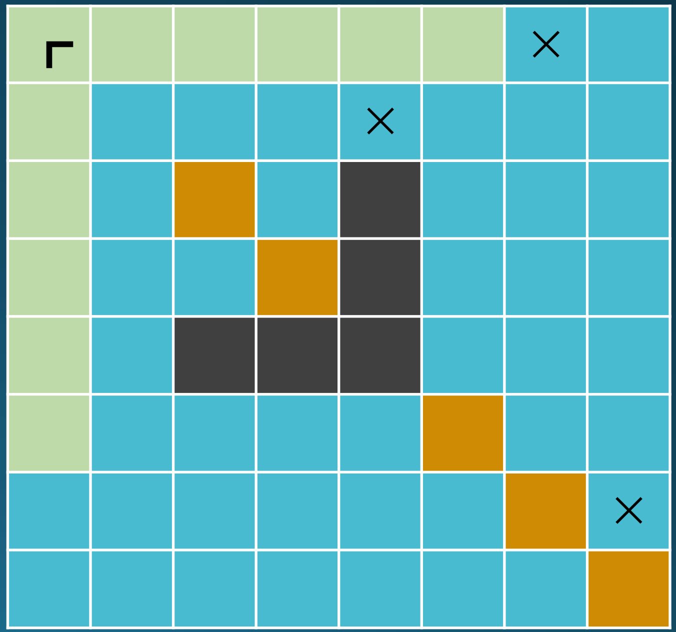
左上部分についての条件

- あまり大きい城壁を作ると，左上から伸びる城壁部分が「外周」「だめなマス」にぶつかってしまう
- 下，右に伸ばしていって，「外周」「だめなマス」にぶつかるまでだけ考えればよい



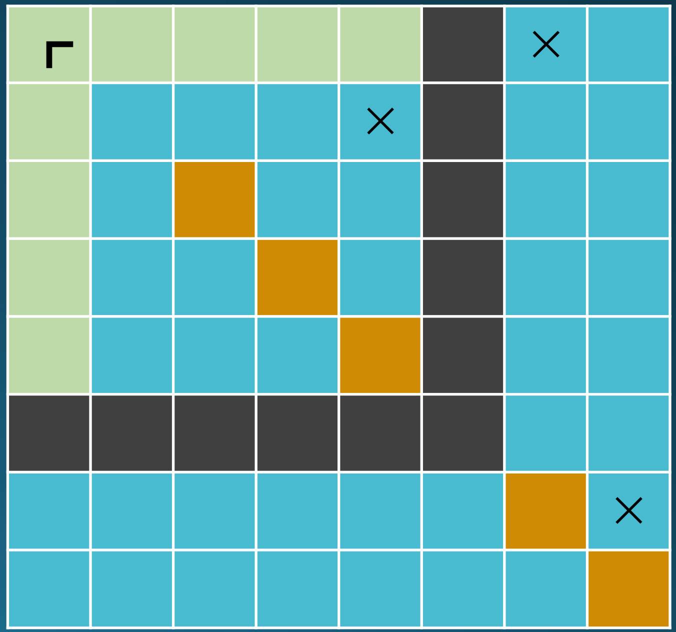
右下部分についての条件

- 右下についても同様のことが言える
- あまり早くに「だめなマス」にぶつかってしまうと，（今固定している）左上から伸びる部分と出会って枠を作ることができない



右下部分についての条件

- こういうのはOK
- 無事に左上から伸びる部分とぶつかって枠ができてる

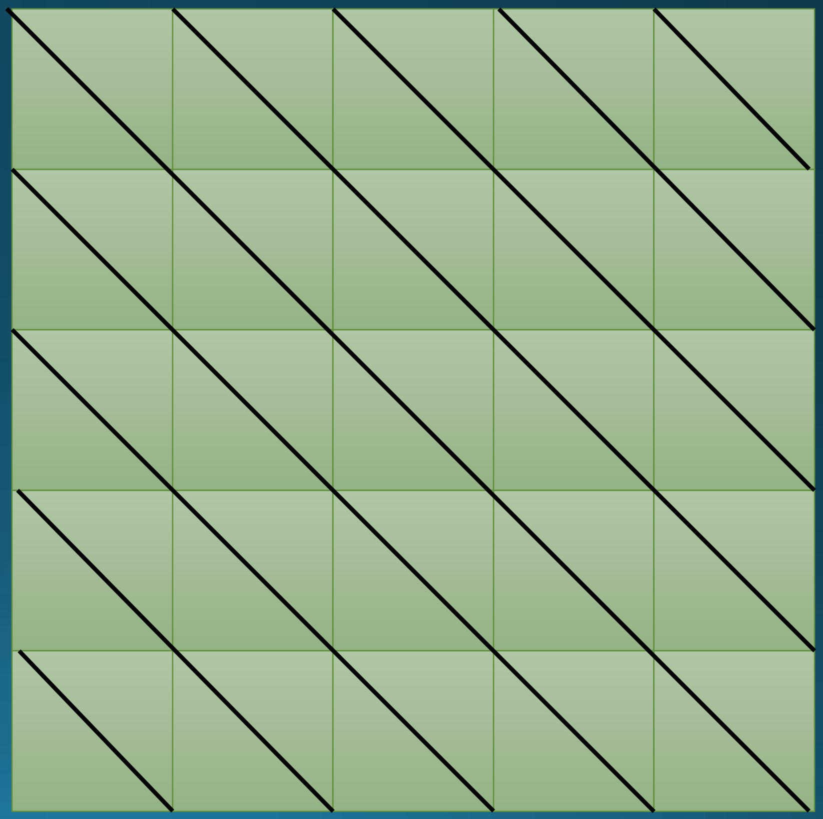


条件まとめ

- 左上/右下から伸ばせる長さの最大値を「限界長さ」と呼ぶことにすると…
- 左上，右下の間が(左上，右下含めて) D マス離れているなら
  - L <= D
  - D <= 左上の限界長さ
  - D <= 右下の限界長さ
  - 当然，左上と右下が同じ斜め線の上にある
- が必要十分

左上，右下の関係

- 同じ斜め線の上にないといけない
- 斜め線ごとに考えてよさそう



限界長さの計算

- 左上の場合について
- 右に何マス伸ばせるかは DP でわかる

	×			
		×		

				1
				1
				1
				1
				1

限界長さの計算

- 左上の場合について
- 右に何マス伸ばせるかはDPでわかる

	×			
		×		

			2	1
			2	1
			2	1
			2	1
			2	1

限界長さの計算

- 左上の場合について
- 右に何マス伸ばせるかはDPでわかる

	×			
		×		

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

限界長さの計算

- 左上の場合について
- 右に何マス伸ばせるかは DP でわかる

	×			
		×		

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

限界長さの計算

- 左上の場合について
- 右に何マス伸ばせるかは DP でわかる

	×			
		×		

5	4	3	2	1
1	0	3	2	1
5	4	3	2	1
2	1	0	2	1
5	4	3	2	1

限界長さの計算

- 左上の場合について
- 下に何マス伸ばせるかも DP でわかる
  - 右に伸ばすのとまったく同様！
  - min をとれば，限界長さが計算できる
- 右下の場合も，伸ばす方向を上と左にしてまったく同様にやればよい

- これで O(HW) で限界長さがすべて計算できる

条件（再掲）

- 左上，右下の間が(左上，右下含めて) D マス離れているなら
  - L <= D
  - D <= 左上の限界長さ
  - D <= 右下の限界長さ

- これを満たすような，左上，右下のペアを数えたい

条件（再掲）

- 左上，右下の間が(左上，右下含めて) D マス離れているなら
  - L <= D <= 左上の限界長さ
  - D <= 右下の限界長さ

- 上の条件だけだったらなんとかなりそう？
  - 左上を決めた時，L 以上 (限界長さ) 以下の範囲に右下が何個あるか数えるとよい
  - データ構造でなんとかできそう
- 下の条件がうっとうしい

条件（書き直し）

- 左上，右下の間が(左上，右下含めて) D マス離れているなら
  - L <= D <= min(左上の限界長さ, 右下の限界長さ)
- つまり，限界長さが短い方により D は制限される

条件（もっと書き直し）

- 左上 / 右下を固定したとき,
- 「L <= D <= (その点の限界長さ)」をみたす相手で,

  !!! 自分より限界長さが長い点 !!!
- を求めたい

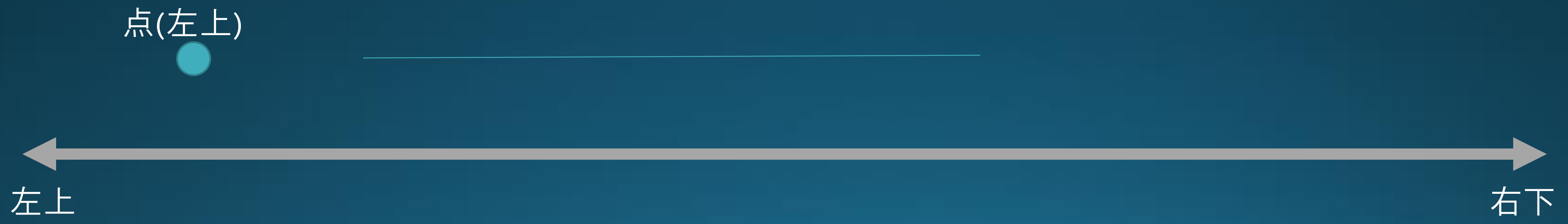
- 限界長さが同じときは，適当に順序を定める

満点解法

- 斜め線を固定して考える
- 線の上の各点を左上/右下としたときの限界長さが長い方から順番に見ていく

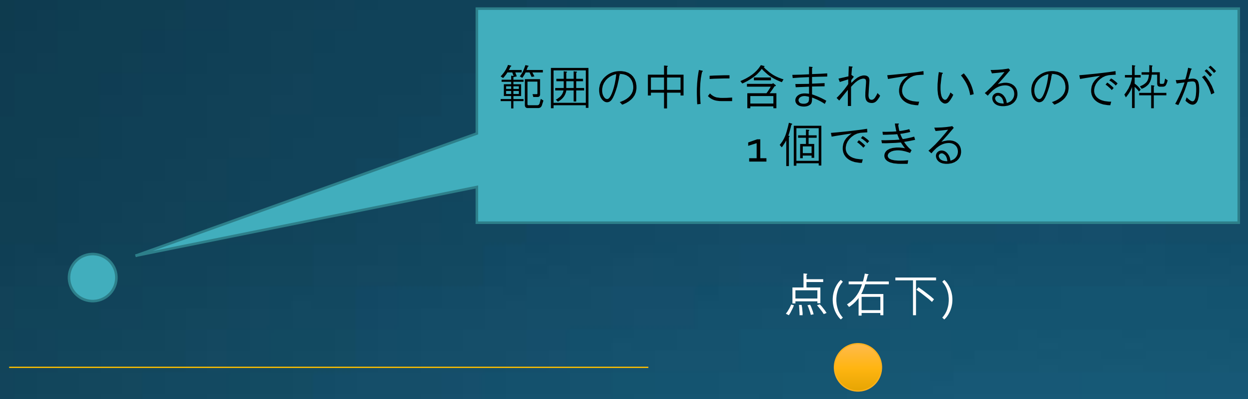
満点解法

- 斜め線を固定して考える
- 線の上の各点を左上/右下としたときの限界長さが長い方から順番に見ていく
- 妥当な範囲の中に，ペアとなりうる点が何個あるか数える



満点解法

- 斜め線を固定して考える
- 線の上の各点を左上/右下としたときの限界長さが長い方から順番に見ていく

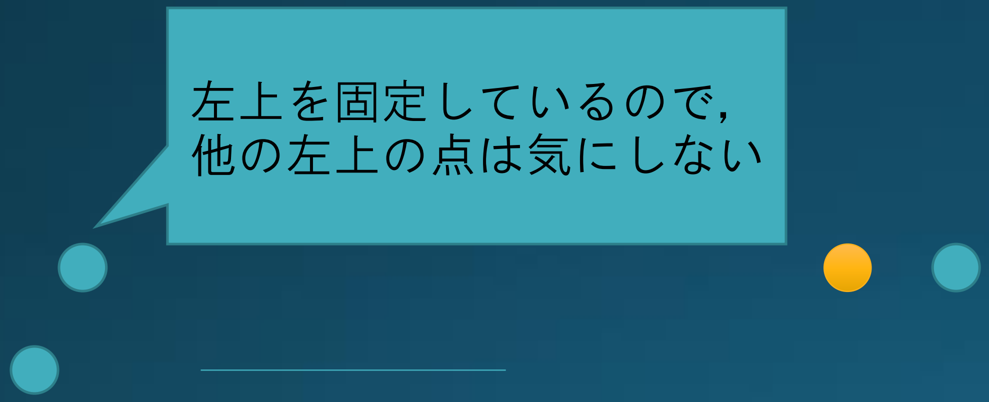


満点解法

- 斜め線を固定して考える
- 線の上の各点を左上/右下としたときの限界長さが長い方から順番に見ていく

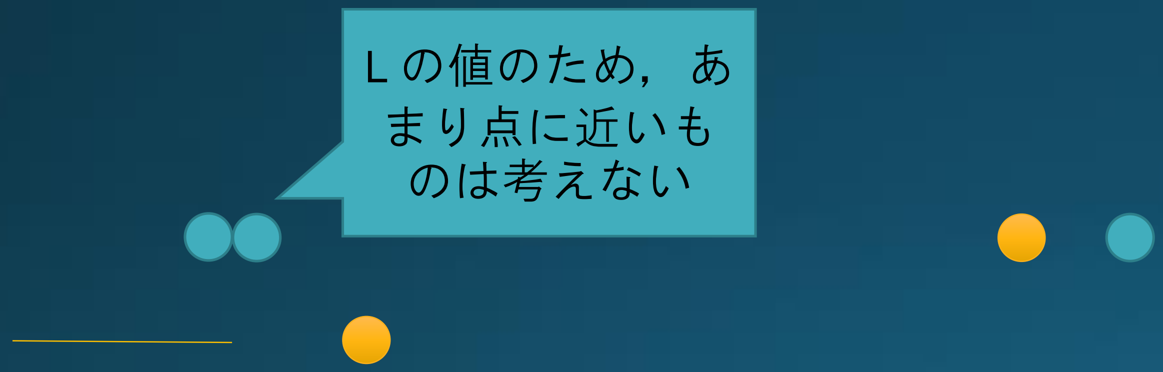
満点解法

- 斜め線を固定して考える
- 線の上の各点を左上/右下としたときの限界長さが長い方から順番に見ていく



満点解法

- 斜め線を固定して考える
- 線の上の各点を左上/右下としたときの限界長さが長い方から順番に見ていく



満点解法

- 今まで見た (左上 / 右下) の点たちの中に，ある範囲に含まれるものが何個あるか？を数えたい
- BIT (Binary Indexed Tree) が使える
  - 詳細は省略します
  - 蟻本などを見てください
- 点が現れたときには，BIT のその位置のところに 1 を加える
- 点を数えるときは，BIT の範囲クエリを使う

計算量

- H, W が N くらいであるとする
- 限界長さの計算は，O(N^2)
- 斜め線は O(N) 個
  - 各斜め線の上に，O(N) 個のマス
  - 限界長さでソートするのに O(N log N)
  - 各マスを見るときに O(log N) (BIT の操作)
  - 結局，斜め線ごとに O(N log N)
- あわせて O(N^2 log N)
- 満点が得られる

注意

- 4000 * 4000 をいっぺんに処理しようとすると多分時間がかかります
  - 4000 * 4000 * sizeof(int) = 64MB くらい
  - キャッシュに載らない
- 限界長さの計算をした後は，斜め線ごとに処理を完全に別々に行うのがよい

得点分布