解説・講評

writer / tester : Astral__, cho57020, Comavius, hotman78, kenken714, Koi51, noya2, Nzt3, otoshigo, ponjuice, potato167, Rice_tawara459, shobonvip, simasima, tassei903, tatyam

目次

- 統計情報
- 運営想定難易度
- 解説(共通問題 6 問のみ)

統計情報 (Div. 2)

番号	問題名	全体AC数	オンサイトFA
A	Various Roots	41	‘11
B	Heavy Rotation	15	てんぷらファンクラブ
C	Sum of Three Inversions	2	なし
D	Boomerang	63	高齢と人外
E	Max Twice Subsequence (Easy)	1	なし
F	Decimal Pyramid	45	しうまい*Meltdown*
G	Team division	0	なし
H	Akari Counting	0	なし
I	Subgrid Connected Componets	0	なし
J	KinGin’s Summit	2	QSy
K	Two-Way Communication	14	neko kick nyaaa-
L	Avoid Multiple	4	'11
M	Take K	58	(should/will/did)PlayManosaba
N	Fair Flags	6	しうまい*Meltdown*
O	Next STPC	101	‘11

番号	問題名	原案	writer	tester
A	Various Roots	noya2	ponjuice	Nzt3
B	Heavy Rotation	Rice_tawara459	Rice_tawara459	hotman78
C	Sum of Three Inversions	otoshigo	otoshigo	Rice_tawara459
D	Boomerang	noya2	Koi51	cho57020
E	Max Twice Subsequence (Easy)	noya2	noya2	tassei903
F	Decimal Pyramid	noya2	shobonvip	tassei903
G	Team division	hotman78	hotman78	tassei903
H	Akari Counting	potato167	hotman78	potato167
I	Subgrid Connected Componets	shobonvip	shobonvip	Astral__
J	KinGin’s Summit	potato167	tassei903	Astral__
K	Two-Way Communication	tatyam	tatyam	
L	Avoid Multiple	shobonvip	Nzt3	Rice_tawara459
M	Take K	shobonvip	cho57020	shobonvip
N	Fair Flags	shobonvip	shobonvip	cho57020
O	Next STPC	noya2	tassei903	noya2

運営想定難易度 (Div. 2)

運営想定難易度順 (Div. 2)

ODMBLNJAFKHEGCI(by noya2)

ODMBALFGJNCKHEI(by potato167)

実際の難易度順

ODMFKANBLCJEGHI

オンサイト順位発表 (Div. 2)

3位
きたむらけん

2位

y_1

1位
チーバくん

統計情報 (Div. 1)

番号	問題名	全体AC数	オンサイトFA
A	Depth of Interval	9	Rubikun
B	Increasing Swaps	4	SYTR-9
C	Sum of Three Inversions	11	Rubikun
D	Grid Path Tree	0	
E	Max Twice Subsequence	0	
F	Decimal Pyramid	22	Rinshan Solution
G	Don’t make zero	14	若造
H	Akari Counting	7	Rinshan Solution
I	Subgrid Connected Components	2	
J	Divide Polygon	1	
K	Two-Way Communication	15	KouchikuZenbuToku
L	Colorful Quadrilateral	0	
M	Many Approaches	2	SYTR-9

番号	問題名	原案	writer	tester
A	Depth of Interval	noya2	noya2	potato167
B	Increasing Swaps	otoshigo	hotman78	otoshigo
C	Sum of Three Inversions	otoshigo	otoshigo	Rice_tawara459
D	Grid Path Tree	potato167	potato167	noya2
E	Max Twice Subsequence	noya2	noya2	tassei903
F	Decimal Pyramid	noya2	shobonvip	tassei903
G	Don’t make zero	simasima	Rice_tawara459	simasima
H	Akari Counting	potato167	hotman78	potato167
I	Subgrid Connected Components	shobonvip	shobonvip	Astral__
J	Divide Polygon	potato167	potato167	noya2
K	Two-Way Communication	tatyam	tatyam	hotman78
L	Colorful Quadrilateral	simasima	noya2	tatyam
M	Many Approaches	noya2	noya2	shobonvip

運営想定難易度 (Div. 1)

運営想定難易度順 (Div. 1)

FKCGHAMJIBDEL(by noya2)

FCKJAHDGBIMEL(by potato167)

実際の難易度順

FKGCAHBIJMDEL

オンサイト順位発表 (Div. 1)

3位
若造

2位
Rinshan Solution

1位
SYTR-9

解説（共通6問のみ）

C - Sum of Three Inversions

提案 : otoshigo
準備 : otoshigo
tester : Rice_tawara459

問題概要

以下の条件を満たすような、長さ N の整数列の 3 つ組 (A, B, C) の個数を求めよ

- (A[i], B[i], C[i]) は (1, 2, 3) の順列
- A に含まれる 1, 2 の個数はそれぞれ X, Y
- (Aの転倒数) + (Bの転倒数) + (Cの転倒数) = K

制約

- 2 <= N <= 50
- 0 <= X + Y <= N
- 0 <= K <= 3N(N - 1) / 2

Z を A に含まれる 3 の個数とする (Z = N - X - Y)

例

N = 3, X = 1, Y = 1, Z = 1, K = 4 で条件を満たす (A, B, C) の一例

A = (1, 2, 3) : 転倒数 0

B = (3, 3, 2) : 転倒数 2

C = (2, 1, 1) : 転倒数 2

- (A[i], B[i], C[i]) はすべて (1, 2, 3) の順列
- A に 1 が 1 個、2 が 1 個、3 が 1 個含まれている
- 転倒数の合計は 4

DP で解く

dp[(1, 2, 3) の個数][(1, 3, 2) の個数][(2, 1, 3) の個数]

　　[(2, 3, 1) の個数][(3, 1, 2) の個数][(3, 2, 1) の個数][転倒数]

という DP で O(N ^ 8) 時間で解ける（もう少し工夫すると O(N ^ 7) になる）

しかし、これでは **TLE** してしまう

順列のペアで考える

A = (1 , 2 , 3)
B = (3 , 3 , 2)
C = (2 , 1 , 1)

+1

A = (1 , 2 , 3)
B = (3 , 3 , 2)
C = (2 , 1 , 1)

+2

A = (1 , 2 , 3)
B = (3 , 3 , 2)
C = (2 , 1 , 1)

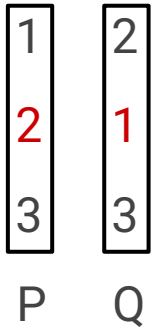
+1

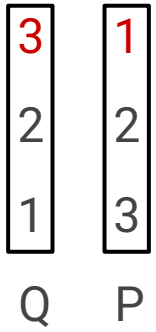
順列のペアをグループ分けする

グループ P : (1, 2, 3), (2, 3, 1), (3, 1, 2)

グループ Q : (1, 3, 2), (2, 1, 3), (3, 2, 1)

グループ P とグループ Q との転倒数の寄与は必ず +1 になる





順列のペアをグループ分けする

グループ P : (1, 2, 3), (2, 3, 1), (3, 1, 2)

グループ Q : (1, 3, 2), (2, 1, 3), (3, 2, 1)

(転倒数の合計)

= (グループ P 同士の転倒数の合計) + (グループ Q 同士の転倒数の合計)

+ (グループ P の順列の個数) × (グループ Q の順列の個数)

と表せる。

各グループでの転倒数の合計

グループ P 同士の転倒数の合計は

dp[(1, 2, 3) の個数][(2, 3, 1) の個数][(3, 1, 2) の個数][転倒数]

という DP で O(N ^ 5) 時間で求められる

グループ Q 同士の転倒数の合計も同じ DP で 求められる

答えの計算

(1, 2, 3), (2, 3, 1), (3, 1, 2) の個数をそれぞれ x, y, z とし、グループ P 同士の転倒数の合計を k とする

このとき (1, 3, 2), (2, 1, 3), (3, 2, 1) の個数はそれぞれ X - x, Y - y, Z - z であり、

グループ Q 同士の転倒数の合計は K - k - (x + y + z)(N - x - y - z) となる

よって、

comb(N, x + y + z) × dp[x][y][z][k] × dp[X - x][Y - y][Z - z][K - k - (x + y + z)(N - x - y - z)]

が答えに寄与する。

答えの計算

x, y, z, k を全探索することで O(N ^ 5) で答えを計算することができる

dp の構築と合わせて、この問題を O(N ^ 5) 時間で解くことができた

空間計算量も O(N ^ 5) となり、**MLE** しないように注意する必要がある

H - Akari Counting

提案 : potato167
準備 : hotman78
tester : potato167

問題概要

H × W の Akari(美術館) の盤面が与えられる。

ただし、マス (i,j) は A ≤ i≤ B, C ≤ j ≤ D のとき黒マス、

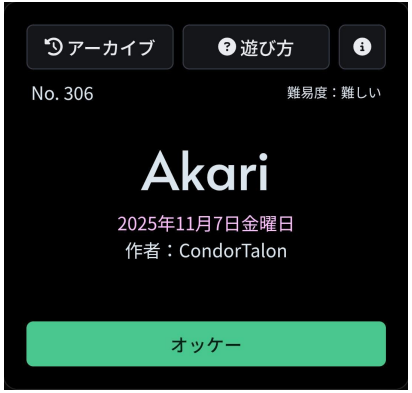
そうでないとき白マス

この問題の解としてあり得る照明の置き方は何個あるか

制約

- 1≤ A≤ B≤ H≤ 5e5

- 1≤ C≤ D≤ W≤ 5e5



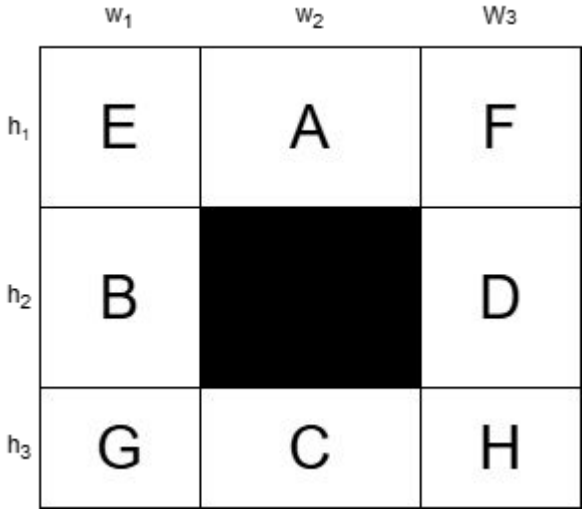
https://dailyakari.com/

考察の方針

盤面を A~H に分け、条件を整理していくと

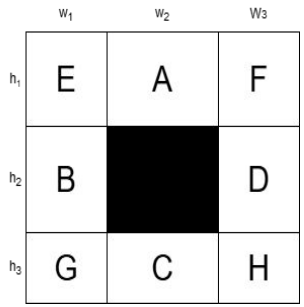
自然と数え上げ方が見えてくる

- 盤面 A, B, C, D が照らされる条件は？
- 盤面 E, F, G, H が照らされる条件は？

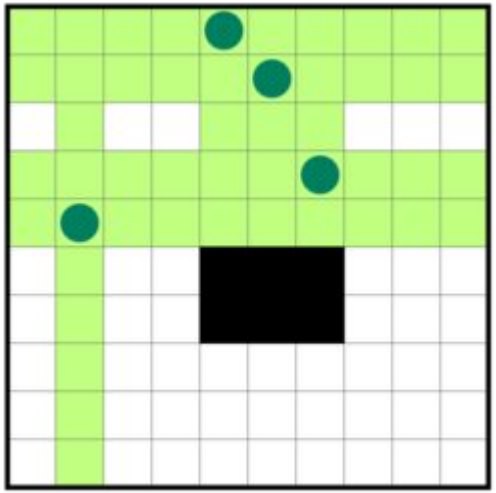


領域 A~H に置かれた照明の数を a~h とする

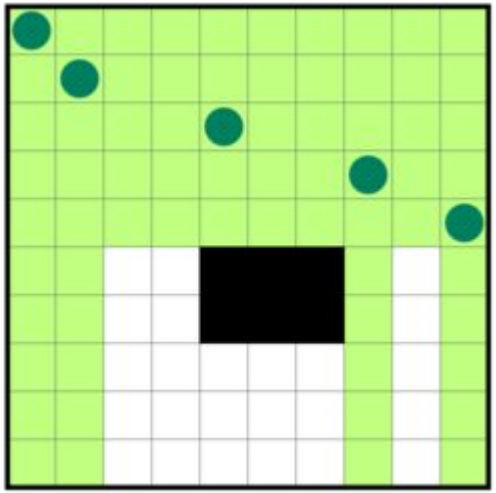
領域 A がすべて照らされる条件



つぎの二通りとなる



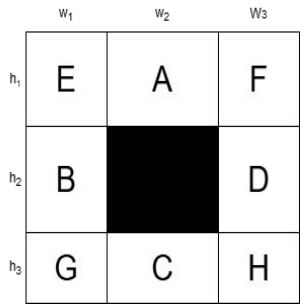
$a=w_2$ かつ $a+e+f \neq h_1$ のとき



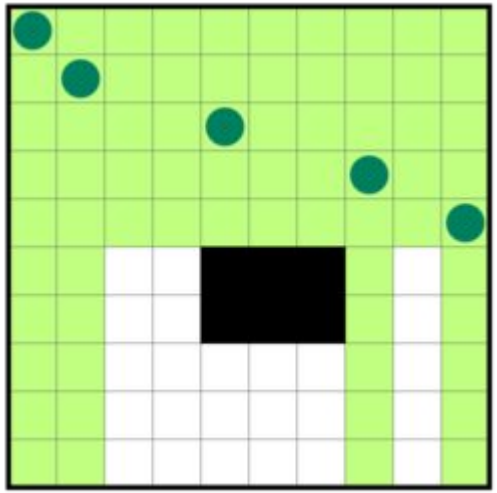
$a+e+f=h_1$ のとき

領域 B, C, D も同様

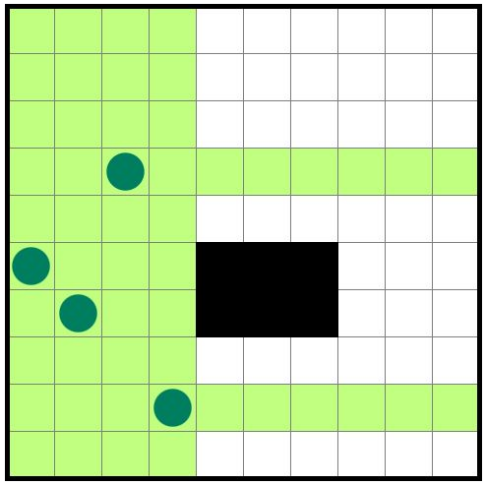
領域 E がすべて照らされる条件



つぎの二通りとなる

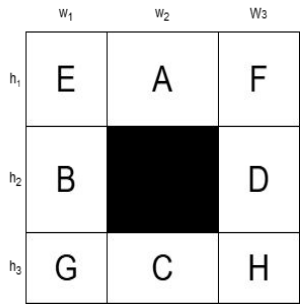


$a+e+f=h_1$ のとき



$b+e+g=w_1$のとき

領域 F, G, H も同様



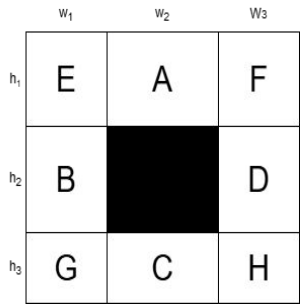
領域 E, F, G, H がすべて照らされる条件

- 領域 E がすべて照らされる条件は $a+e+f=h_1$ または $b+e+g=w_1$
- 領域 F がすべて照らされる条件は $a+e+f=h_1$ または $d+f+h=w_3$
- 領域 G がすべて照らされる条件は $c+g+h=h_3$ または $b+e+g=w_1$
- 領域 H がすべて照らされる条件は $c+g+h=h_3$ または $d+f+h=w_3$

よって、領域 E, F, G, H がすべて照らされる条件は

$(a+e+f= h_1$ かつ $c+g+h=h_3)$ または $(b+e+g=w_1$かつ $d+f+h=w_3)$

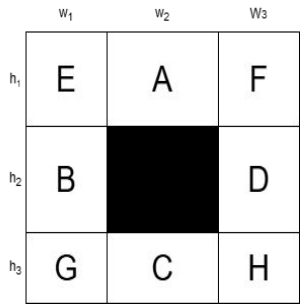
数え上げの問題点



- 領域 A が照らされる条件には 領域 A, E, F の照明が関与
- 領域 B が照らされる条件には 領域 B, E, G の照明が関与
- 領域 C が照らされる条件には 領域 C, G, H の照明が関与
- 領域 D が照らされる条件には 領域 D, F, H の照明が関与

このままでは、領域 E の照明が領域 A, B の両方に関与して数えづらい

数え上げの問題点



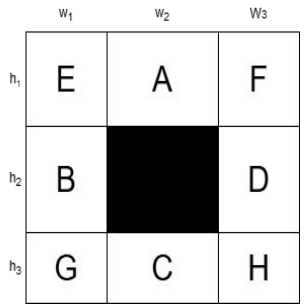
- 領域 A が照らされる条件には 領域 A, E, F の照明が関与
- 領域 B が照らされる条件には 領域 B, E, G の照明が関与
- 領域 C が照らされる条件には 領域 C, G, H の照明が関与
- 領域 D が照らされる条件には 領域 D, F, H の照明が関与

このままでは、領域 E の照明が領域 A, B の両方に関与して数えづらい

実は領域 A が照らされる条件には領域 E, F の照明の**行成分**のみが関与！！

実は領域 B が照らされる条件には領域 E, G の照明の**列成分**のみが関与！！

数え上げ

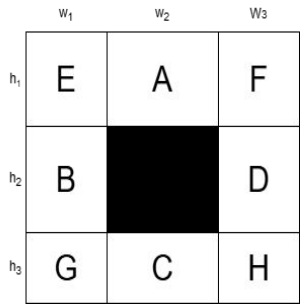


- $X_0[i]$, $X_1[i]$: e+f=i のときの「領域 A の照明の配置」と「領域 E,F の中で照明を置かれている行の集合」の組の場合の数

ただし、

- $X_0[i]$: $a+e+f= h_1$
- $X_1[i]$: $a+e+f\neq h_1$ かつ $a=w_2$

とし、同様に 領域 C, G, H について求めた物を $Y_0[i]$, $Y_1[i]$ とする



数え上げ

- $P_1[i] = \sum_{j+k=i} X_0[j]Y_1[k] + X_1[j]Y_0[k] + X_1[j]Y_1[k]$
- $P_0[i] = \sum_{j+k=i} X_0[j]Y_0[k]$

としたら、

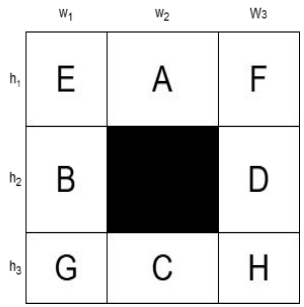
- $P_0[i]$, $P_1[i]$：e+f+g+h = i のときの「領域 A,C の照明の配置」と「領域 E,F,G,H の中で照明を置かれている行の集合」の組の場合の数

ただし、

- $P_0[i]$:a+e+f= h_1 かつ c+g+h=h_3
- $P_1[i]$：a+e+f≠ h_1 または c+g+h≠h_3

列について求めたものも同様に$Q_0[i]$, $Q_1[i]$ とする

数え上げ



- $T[i] = \sum_{j+k=i} P_0[j]Q_1[k] + P_1[j]Q_0[k] + P_1[j]Q_1[k]$

としたら、

- T[i] : e+f+g+h = i のときの「領域 A, B, C, D の照明の配置」と「領域 E, F, G, H の中で照明を置かれている行の集合」と「領域 E, F, G, H の中で照明を置かれている列の集合」の組の場合の数

ただし、

(a+e+f= h_1 かつ c+g+h=h_3) または (b+e+g=w_1かつ d+f+h=w_3)

よって、行と列の対応付けを考えて $\sum_i$ i!×T[i] が答え！！

E - Max Twice Subsequences

提案 : noya2
準備 : noya2
tester : tassei903

問題概要

整数列 B = (B_1,B_2,...,B_n) に非空な部分列として 2 回以上現れるものが存在するかどうか判定し、存在するなら辞書順で最大のものを求めよ。

例

B = (3, 2, 1, 2, 3) の答え : (3, 2, 3)

(3, 2, 1, 2, 3) と (3, 2, 1, 2, 3) で 2 つ取れる

B = (3, 2, 1) の答え : なし

問題概要

整数列 B = (B_1,B_2,...,B_n) に非空な部分列として 2 回以上現れるものが存在するかどうか判定し、存在するなら辞書順で最大のものを求めよ。

という部分問題を考える。

整数列 A = (A_1,A_2,...,A_N) が与えられる。部分問題で B = A[L_i,R_i] としたときの答えを求めよ、というクエリを Q 個答える。

- 1 <= N, Q <= 2e5

- 1 <= A_i <= N

- **クエリで与えられる区間の長さの総和は　1e7 以下 (Easy のみ)**

部分問題を線形時間で解く

Q. 部分列として 2 回以上現れるとは？（ARC195A - Twice Subsequence で既出）

A. 前から貪欲に取ったときと後ろから貪欲に取ったときで、位置が異なる

重要な考察 1 : 位置はちょうど 1 つだけ異なるとして良い

(1,2,1,2,2,1,1,1,2)　(1,2,1,2,2,1,1,1,2)

(1,2,1,2,2,1,1,1,2) → (1,2,1,2,2,1,1,1,2)

たくさん異なるかもしれない → **1 つを除いて揃えられる！**

部分問題を線形時間で解く

狭義単調増加な添字の列 (i_1, ..., i_a, l, r, j_1, ..., j_b) であって B_l = B_r を満たすようなものに対する (B[i_1], ..., B[i_a], B[l], B[j_1], ..., B[j_b]) の辞書順最大化

に言い換えられた。

[考えられそうなアルゴリズム]

1. 後ろに重複要素を残したまま貪欲に (i_1, ..., i_a) を取る

2. 適切なタイミングで打ち切って B_l = B_r なる重複要素を取る

3. それ以降は本当に貪欲に最後まで (j_1, ..., j_b) を取る

部分問題を線形時間で解く

狭義単調増加な添字の列 (i_1, ..., i_a, l, r, j_1, ..., j_b) であって B_l = B_r を満たすようなものに対する (B[i_1], ..., B[i_a], B[l], B[j_1], ..., B[j_b]) の辞書順最大化

に言い換えられた。

[考えられそうなアルゴリズム]

1. 後ろに重複要素を残したまま貪欲に (i_1, ..., i_a) を取る

2. 適切なタイミングで打ち切って B_l = B_r なる重複要素を取る

3. それ以降は本当に貪欲に最後まで (j_1, ..., j_b) を取る

部分問題を線形時間で解く

先頭の要素を最大化するには？

先頭の要素を B[i] としてみる

- j < i なる B[j] は取れない

- **{B[i], B[i+1], ..., B[n]} に重複要素がないとダメ**

例 : (1, 2, 3, 2, 1) で 3 を先頭には選べない

重複要素がない

部分問題を線形時間で解く

{B[s],B[s+1],...,B[n]} に重複要素がないような最小の s を計算しておく

- 先頭の要素を B[i] とするとき、i < s が必要

- 逆にそのような B[i] なら先頭にできる

- B[i] として {B[1],B[2],...,B[s-1]} の最大値を選ぶべき！

- 同じ B[i] なら i が最小のものを選ぶ（部分列を貪欲に取るので）

部分問題を線形時間で解く

狭義単調増加な添字の列 (i_1, ..., i_a, l, r, j_1, ..., j_b) であって B_l = B_r を満たすようなものに対する (B[i_1], ..., B[i_a], B[l], B[j_1], ..., B[j_b]) の辞書順最大化

に言い換えられた。

[考えられそうなアルゴリズム]

1. 後ろに重複要素を残したまま貪欲に (i_1, ..., i_a) を取る

2. 適切なタイミングで打ち切って B_l = B_r なる重複要素を取る

3. それ以降は本当に貪欲に最後まで (j_1, ..., j_b) を取る

部分問題を線形時間で解く

いつ打ち切るべきなのか？

最適な先頭の添字 i が求められたとして、次の 2 つの選択肢がある:

1. a >= 1 を確定させて i_1 := i とし、同様の貪欲を続ける

2. a = 0 を確定させて l := i とし、r として B[r] = B[l], r > l なる最小の r を取り、次の段階へ進む

部分問題を線形時間で解く

例 : (2, 3, 1, 3, 1, 2) とすると i = 2, B[i] = 3 で、

1. i_1 := 2 として、残りの (1, 3, 1, 2) で同様の貪欲をする

2. l := 1, r := 3 として、r より大きな添字の部分 (1, 2) について本当の貪欲をする

注意！片方しか選択肢がない場合もある

例 : (2, 3, 1, 3, 4, 2) とすると i = 2, B[i] = 3 だが、1. を選ぶと残りの部分が重複要素を持っていないので不適

例 : (2, 4, 1, 3, 1, 2) とすると i = 2, B[i] = 4 だが、2. を選ぶと r が存在しない（B[i] が重複していない）ので不適

部分問題を線形時間で解く

選択肢が 1 つの場合はそれを選ぶしかない。2 つの場合はどちらを選ぶべきか？

→ 先頭の要素は最大化されているから、次の要素を最大化するべき

各選択肢で次の要素の最大値はどうなるか？

1. を選んだ場合：残りの部分で位置 s より手前にある値の最大値
2. を選んだ場合：r より後ろにある値の最大値

部分問題を線形時間で解く

例 : (2, 3, 1, 3, 1, 2) とすると i = 2, B[i] = 3 で、

1. i_1 := 2 として、残りの (1, 3, 1, 2) で同様の貪欲をする

→ 次の要素の最大値は 1

2. l := 1, r := 3 として、r より大きな添字の部分 (1, 2) について本当の貪欲をする

→ 次の要素の最大値は 2

選択肢 2. の方が大きいので選択肢 2. を選ぶべき！

部分問題を線形時間で解く

値が同じ場合はどちらを選ぶべきか？

→ **実は 1. を選ぶべき**

選択肢 2. を選んだとして r より後ろを貪欲に取った部分列を、選択肢 1. を選んでも取ることができる！

例 : (2, 3, 2, 3, 1, 2) とすると i = 2, B[i] = 3 で、

2. を選んだ場合 : (1, 2) で本当の貪欲をして (2) を得る

1. を選んだ場合 : (2, 3, 1, 2) で同様の貪欲をするが、次の要素の最大値として 1. で選んだ 2 より手前の 2 を取ることができる

部分問題を線形時間で解く

前から決めていく最適な戦略が得られた！

- s
- tolim[j] := j より後ろで s より手前にある値の最大値とそれを与える最小の添字
- toend[j] := j より後ろにある値の最大値とそれを与える最小の添字
- nxt[j] := j より後ろにある B[j] と同じ値を与える最小の添字

を前計算しておくことで、貪欲に次の添字を取ったり選択肢の比較をしたりするのが定数時間でできる。

→ 線形時間で解けた！

高速化（Div. 1 のみ）

s や貪欲で取る添字が R に強く依存し、逆に L にはほとんど依存しない

→ 右端 R の昇順にクエリ (L, R) をオフラインで処理する

- R が固定されているとき、大量の L について高速に答えられるか？
- R を 1 増やしたときの更新が高速に行えるか？

高速化（Div. 1 のみ）

s や貪欲で取る添字が R に強く依存し、逆に L にはほとんど依存しない

→ 右端 R の昇順にクエリ (L, R) をオフラインで処理する

- R が固定されているとき、大量の L について高速に答えられるか？
- R を 1 増やしたときの更新が高速に行えるか？

高速化（Div. 1 のみ）

選択肢 2. を選んで本当の貪欲に切り替わるタイミングが分かったとする。このときの l, r を取っておく。

tolim, toend に現れる添字 j の位置に列 (B[j]) の連結に関するローリングハッシュをセグメント木に乗せておけば、

前者への [L, l] と後者への (r, R] への区間取得で答えを計算できる！

高速化（Div. 1 のみ）

いつ本当の貪欲に切り替わるのか？

tolim に現れる添字のうち、i として選ばれたときに切り替わるか切り替わらないかは L に依らずに決まっている

→ 切り替わる添字の集合の中で L 以上のもののうち最小のものの位置

(set の lower_bound)で本当の貪欲に切り替わる！

高速化（Div. 1 のみ）

持っておきたいもの：

- s
- tolim に現れる添字の集合・それに対応するセグメント木
- toend に現れる添字の集合・それに対応するセグメント木
- nxt
- 切り替わる添字の集合 I

高速化（Div. 1 のみ）

s や貪欲で取る添字が R に強く依存し、逆に L にはほとんど依存しない

→ 右端 R の昇順にクエリ (L, R) をオフラインで処理する

- R が固定されているとき、大量の L について高速に答えられるか？
- R を 1 増やしたときの更新が高速に行えるか？

高速化（Div. 1 のみ）

s の更新：R の増加に対して単調増加なので、全体で線形時間で

tolim の更新：s の増加に対応して stack で管理できる

toend の更新：R の増加に対応して stack で管理できる

nxt の更新：R の増加に対して高々 1 箇所しか変化しない

l の更新：tolim, toend, nxt の更新に対応して更新される ← これが大変

高速化（Div. 1 のみ）時間切れ

I の更新はかなり複雑で大変です。I の要素がそうでなくなる回数が全体で線形回なので、新たに I の要素になるものを愚直に調べても良いです。

このことを利用して R の増加に対して償却 O(log N) 時間で I の更新を行うことができます。詳細は AtCoder 上の解説をご覧ください。

これらを丁寧に実装すると、全体で O((N+Q)log N) 時間で動作する解法が得られます。

F - Decimal Pyramid

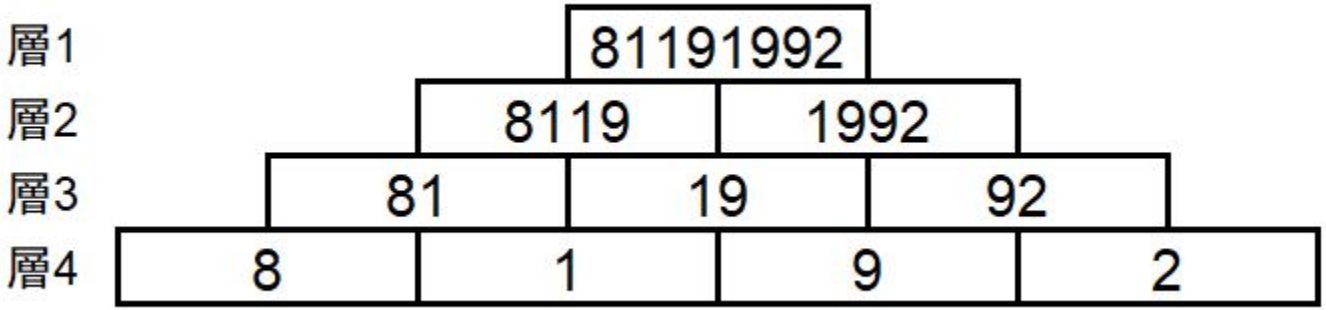
提案：noya2
準備：shobonvip
tester：tassei903

問題概要

高さNのピラミッドがあり、最下層のブロックに書かれている値が与えられる。

最下層以外の層のブロックには、その左下と右下のブロックに書いてある数を結合したものが書かれている

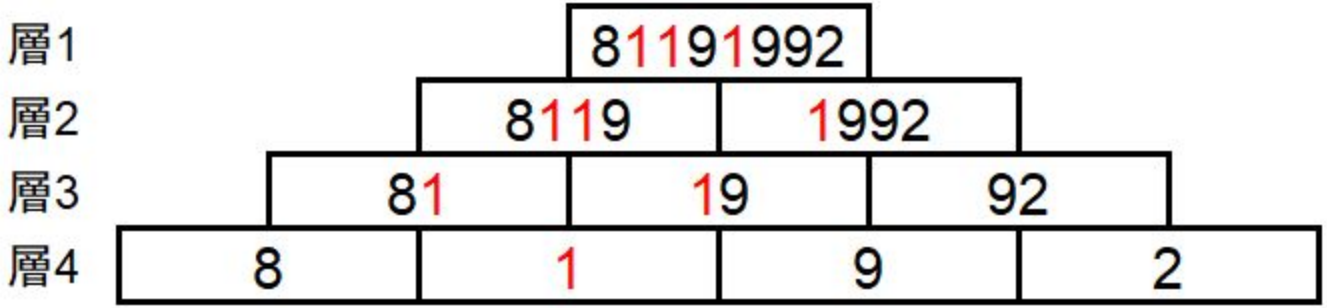
一番上の値 mod 998244353 を求めよ（この場合、81191992）



制約: **N≦200000, 実行時間 3 秒**

考察

最下層の各ブロックが答えにどれだけ寄与するか　を考えてみる（主客転倒）



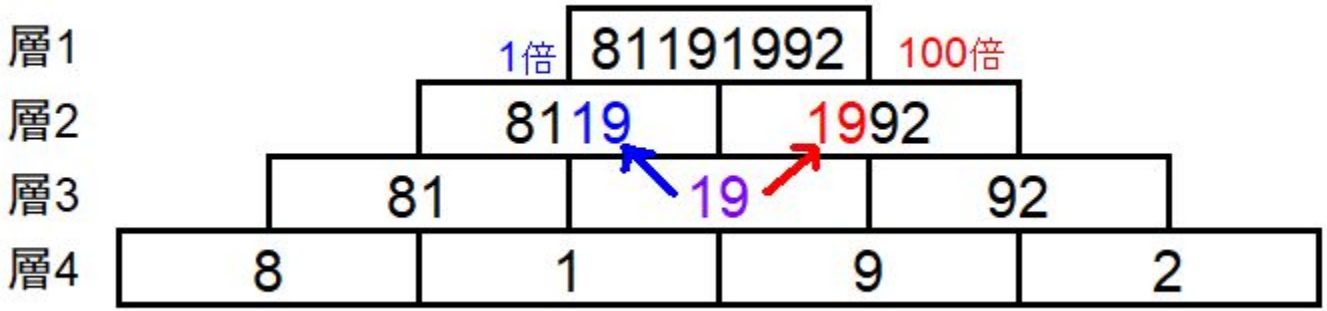
とりあえず、答えに直接行くのではなく、１個上の層に行ったときの挙動を見てみよう

１個上へ層に行くときの挙動を観察

この例において、層3の「19」がどのように層2に現れるかというと……

・「19」の**左上**のブロックに**１倍**で出現する

・「19」の**右上**のブロックに**100倍**で出現する

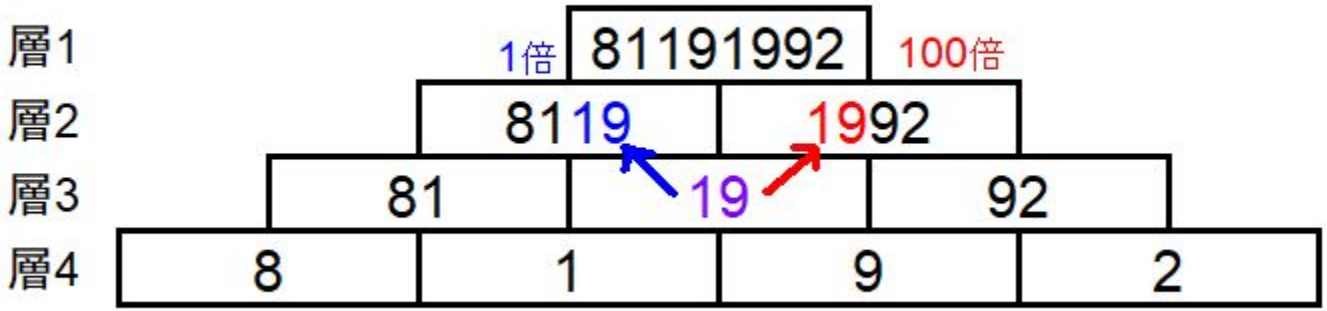


1個上へ層に行くときの挙動を観察

一般に、層 i のブロックは層 i-1 に対して

・**左上**のブロックに **1 倍**で出現する

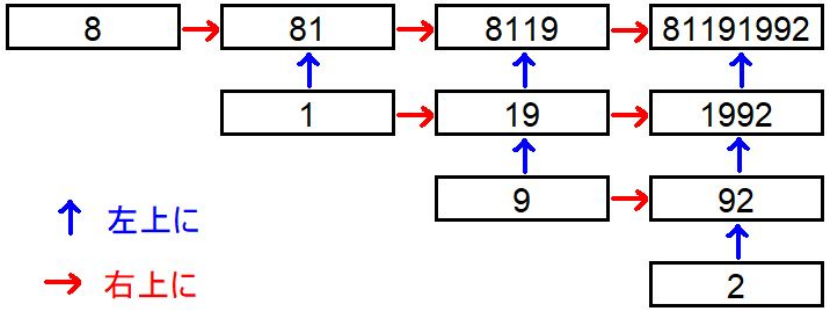
・**右上**のブロックに **10^(2^(N-i)) 倍**で出現する

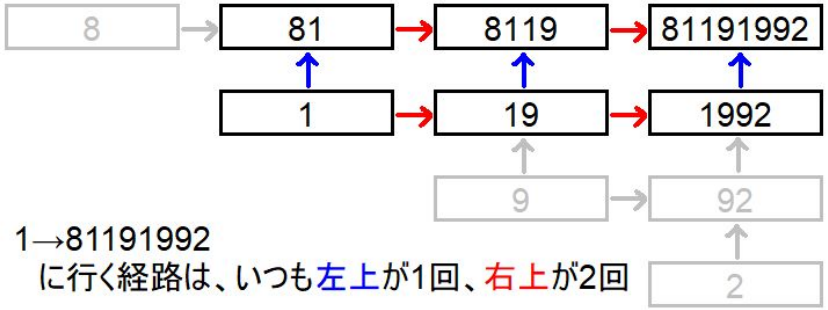


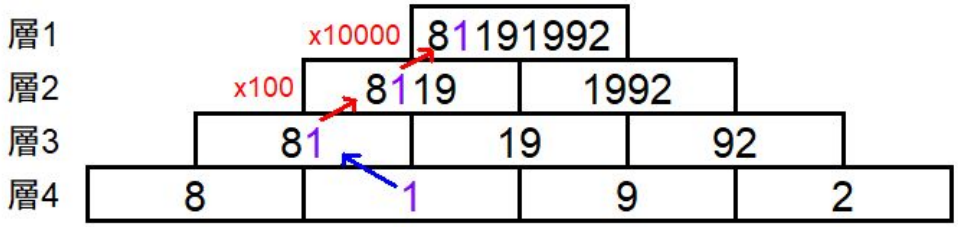
考察

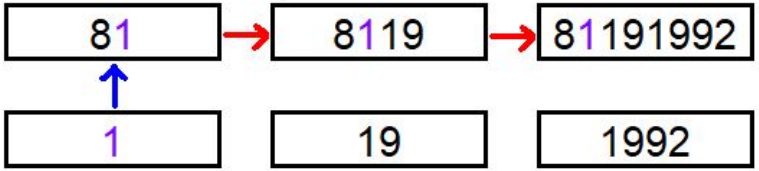
あるブロックの値が1個上のブロックにどう現れるのかは分かったが、もう特に何も言うことはない… と思いきや？

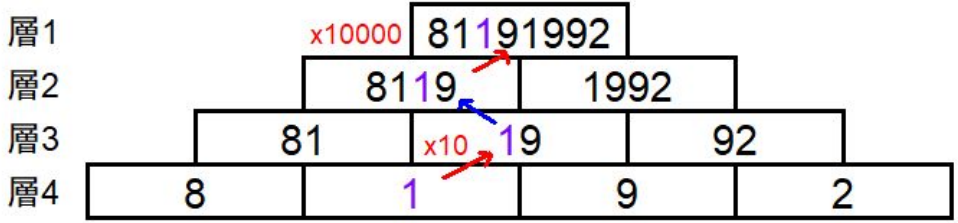
重要な考察：最下層のブロックが最上位にいくとき、**左上にいく回数と右上にいく回数**は決まっている！

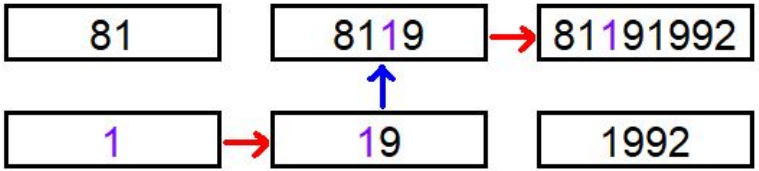


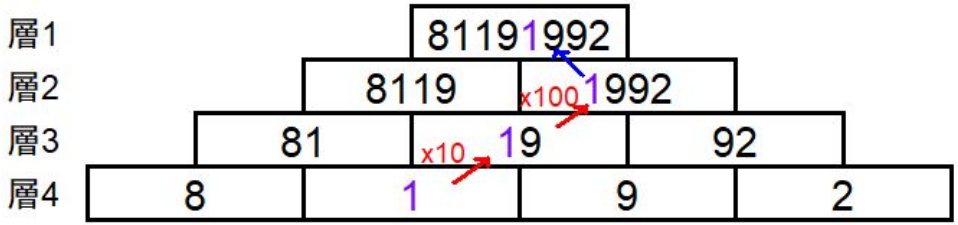


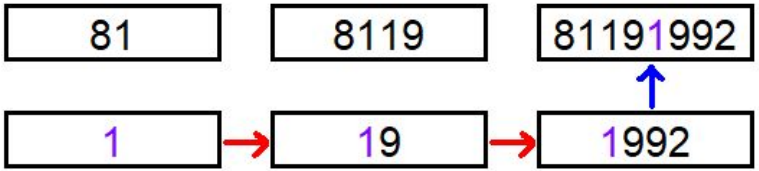

層1
層2
層3
層4
x10000
x100
81191992
8119
1992
81
19
92
8
1
9
2


81
8119
81191992
1
19
1992

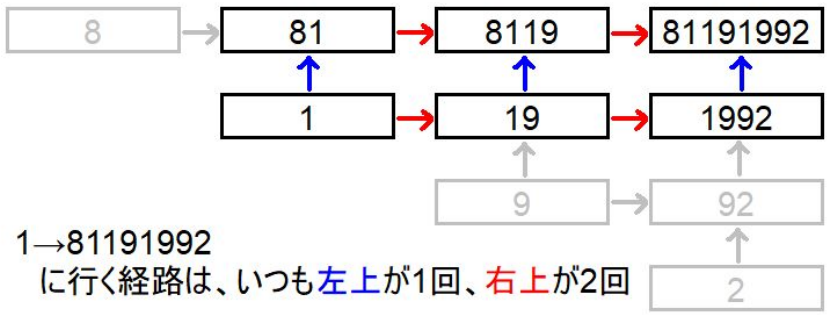

層1
層2
層3
層4
x10000
x10
81191992
8119
1992
81
19
92
8
1
9
2


81
8119
81191992
1
19
1992


層1
層2
層3
層4
x100
x10
81191992
8119
1992
81
19
92
8
1
9
2


81
8119
81191992
1
19
1992

考察にとどめをさそう



1→81191992
に行く経路は、いつも左上が1回、右上が2回

・最下層の左から i 番目のブロックは
左上に i-1 回、右上に N-i 回進む

・t 回目の移動のとき、左上に行くと1倍、右上に行くと 10^(2^(t-1)) 倍される

「右上に進む」を x として多項式で表すと、最下層の左から i 番目のブロック（値は　　）の答えへの寄与は、

$C_{N,i}$

$$C_{N,i}[x^{N-i}]\prod_{t=1}^{N-1}(1+10^{2^{t-1}}x)$$

完成！

$$C_{N,i}[x^{N-i}]\prod_{t=1}^{N-1}(1+10^{2^{t-1}}x)$$

よく見ると、求める多項式は i の値に依存していない
→ この多項式の展開を求めれば最下層の全部のブロックの寄与が分かり、問題の答えも分かる
→ 多項式 mod 998244353 の展開は、**多項式マージテク** と呼ばれる分割統治法（FFTを使う）で時間計算量 O(N log^2 N) で求めることができ、間に合う
　多項式マージテクは FPS24題の I - スコアの他、ABC352-GやACL Beginner ContestのF、などたくさん出題例がある

I - Subgrid Connected Components

提案 : shobonvip
準備 : shobonvip
tester : Astral__

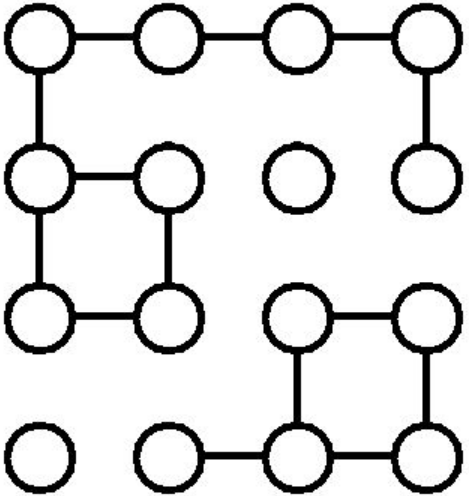
問題概要

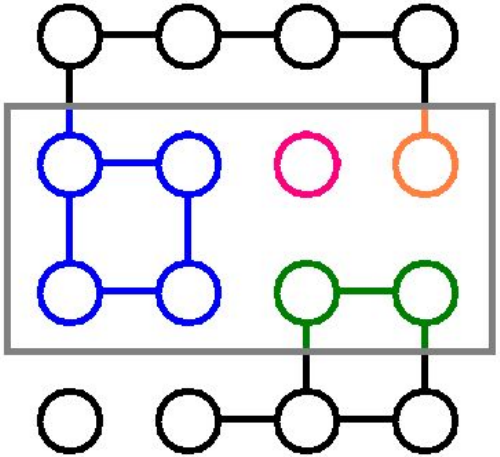
(N+1)×(N+1) グリッドグラフの辺があったりなかったりするものが与えられる

次のクエリにQ回答えよ：部分グリッド [U,D]×[L,R] の連結成分の個数は何個か？

制約：N≦2000, Q≦7000, 実行時間 2.5 秒, メモリ制限 384 MiB

```
o-o-o-o
|.....|
o-o.o.o
|.|...=
o-o.o-o
....|.|
o.o-o-o
```



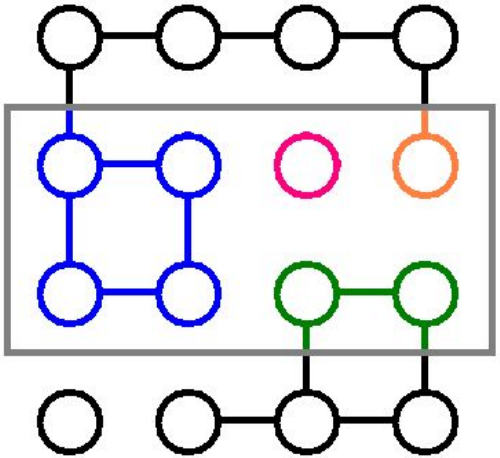


考察

(N+1)×(N+1) グリッドグラフの辺があったりなかったりするものが与えられる

次のクエリにQ個答えよ：部分グリッド [U,D]×[L,R] の連結成分の個数は何個か？

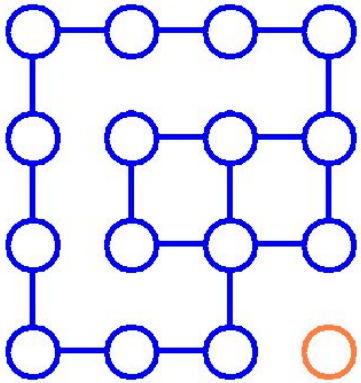
制約：N≦2000, Q≦7000, 実行時間 2.5 秒, メモリ制限 384 MiB

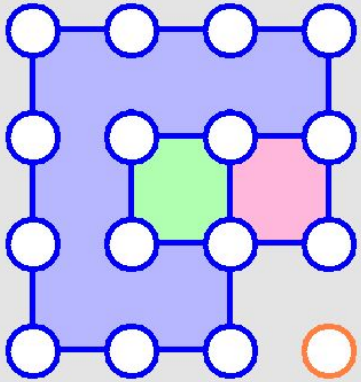


グリッドグラフは**平面グラフ**

うまく使えないか……？

考察





双対グラフなので、辺によって区切られる**領域**が考えられる

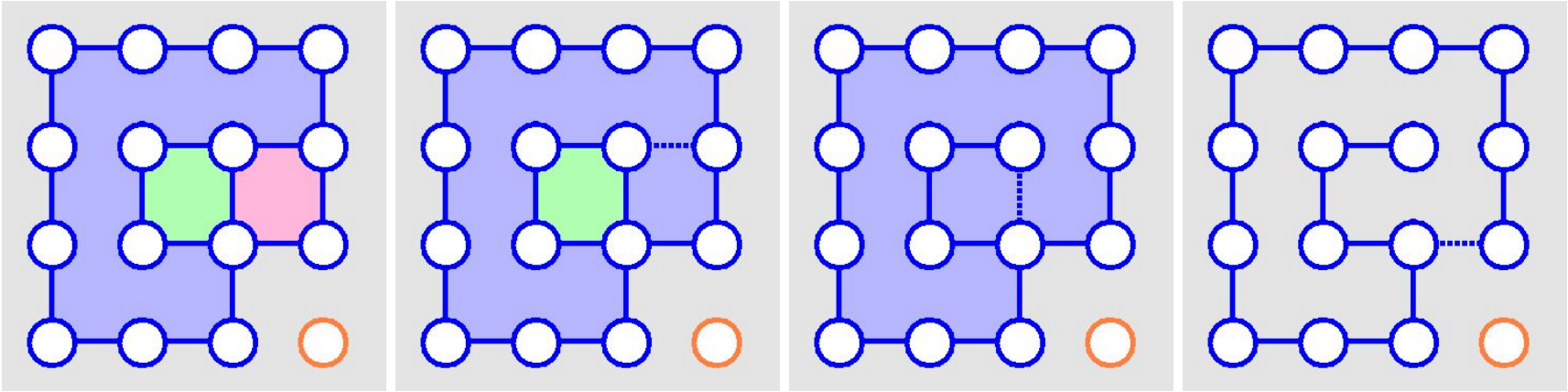
一番外にも無限に広がる領域があると考える

外の領域以外は、必ず辺によって囲まれている

←この場合、領域は 4 個

考察

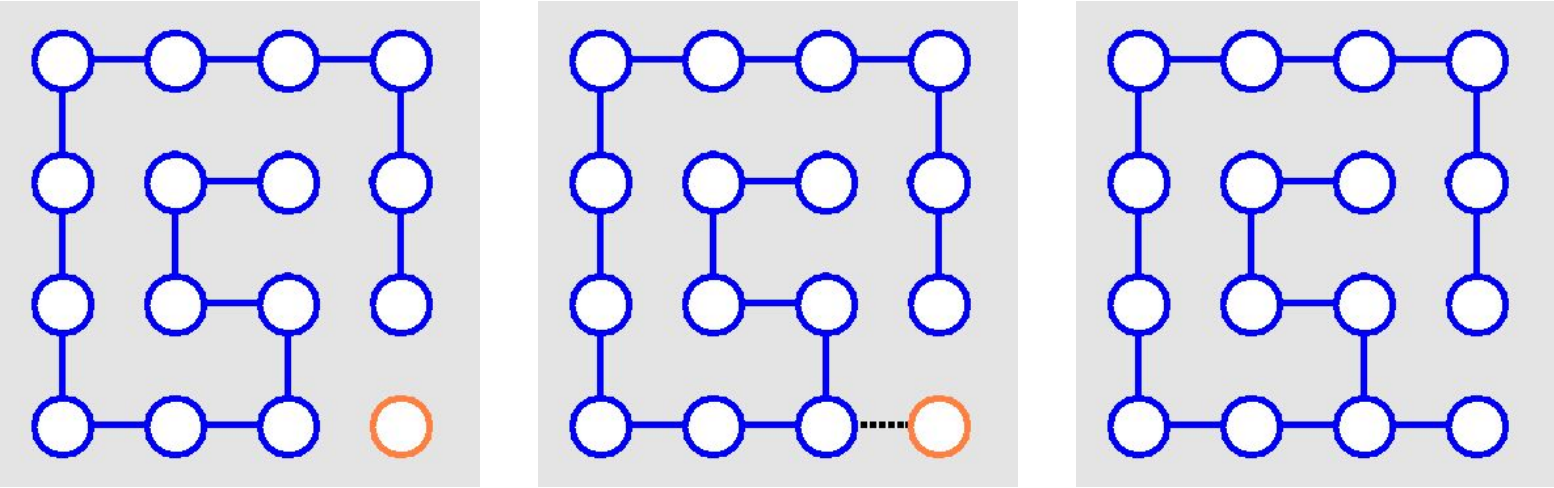
領域を囲っている辺を 1 本削除すると、
連結成分数を変えずに領域を 1 個減らすことができる



領域の個数を r として、r-1 回辺を削除することで**全域森**ができる

考察

その後、異なる連結成分間に辺を追加すると、
領域の数を変えずに連結成分数を 1 減らすことができる



連結成分数を a として、a-1 回辺を追加することで**全域木**ができる

考察

領域の個数を r として、r-1 回辺を削除することで全域森にできる

その後、連結成分数を a として、a-1 回辺を追加することで全域木にできる

全域木ということは……？？

考察

領域の個数を r として、r-1 回辺を削除することで全域森にできる

その後、連結成分数を a として、a-1 回辺を追加することで全域木にできる

全域木ということは……？？

最終的な辺の本数 は **(頂点の個数)-1** になる

→ もともとの辺の本数を e として、頂点の個数を v として

$$e - (r-1) + (a-1) = v-1$$

整理して

$$a = v - e + r - 1$$

考察

$$a = v - e + r - 1$$

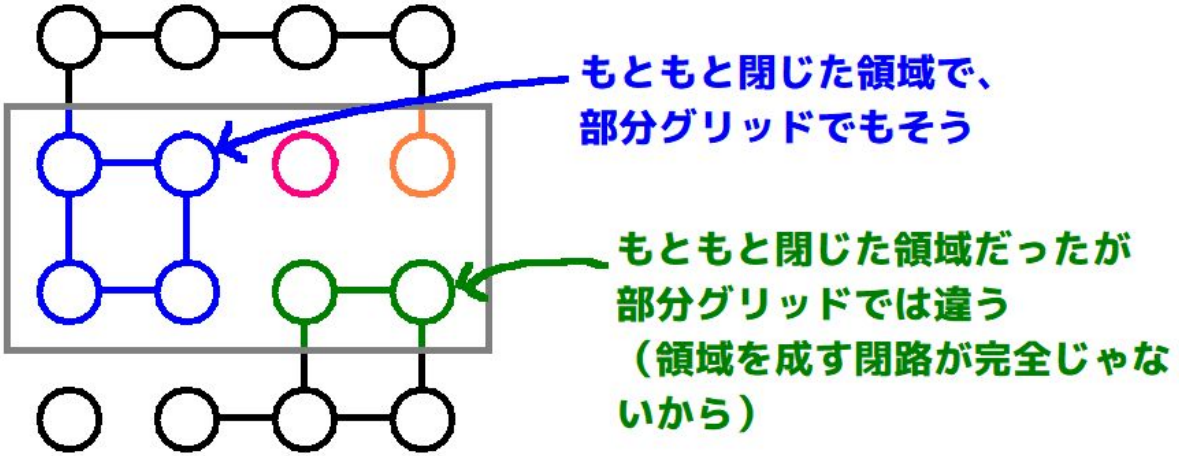
部分グリッドの頂点の個数 v、辺の本数 e、領域の個数 r が分かれば、連結成分の個数 a が分かる

頂点の個数 v は問題文中にも書かれている

辺の個数 e は辺がある場所を累積和にのせてカウントできる

領域の個数 r が一番難しい！！

→ もとのグリッドの領域と、部分グリッドの領域はどのような関係があるか
考える必要がある



もともと閉じた領域であるものについて、それが部分グリッドにおいて

・完全に含まれている
　→ 部分グリッドでもそれが閉じた領域になる

・一部分しかない or 全く含まれてない
　→ 部分グリッドでは領域が開かれる（もっというと、外の領域とつながる）

やりたいこと

部分グリッドに**完全に**含まれる、
「もともと閉じてあった領域」の個数が知りたい

→ これが難しい
「長方形に含まれる長方形の個数」のクエリにはなるけど……

やりたいこと

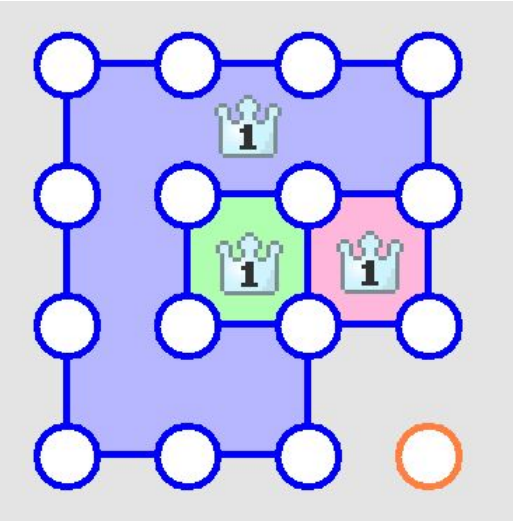
部分グリッドに**完全に**含まれる、
「もともと閉じてあった領域」の個数が知りたい

→ これが難しい
「長方形に含まれる長方形の個数」のクエリにはなるけど……

→「長方形に含まれる**点**の個数」のクエリなら累積和などで簡単に解ける
どうにかして、これに出来ないのか？

「長方形に含まれる**点**の個数」のクエリなら累積和などで簡単に解ける
どうにかして、これに出来ないのか？

→「もともと閉じてあった領域」の**ある 1 点を代表点として決めてみる**

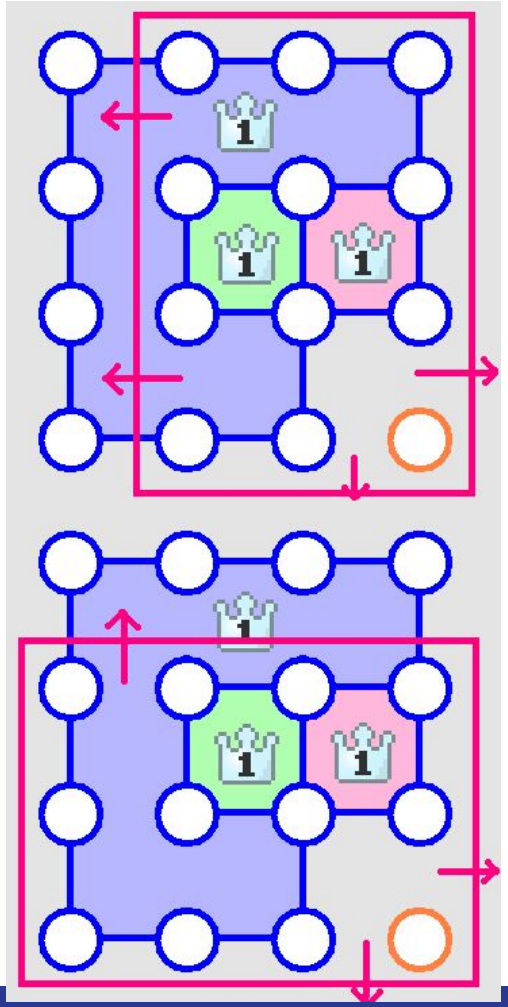


部分グリッドがその領域を

・完全に含んでいる
→ 代表点は必ず部分グリッドに含まれる

・完全に含んでない
→ 代表点は必ず部分グリッドに含まれない

・一部分だけ含んでいる
→ 代表点は部分グリッドに含まれたり含まれなかったりする
含まれていたら代表点を除きたい



・一部分だけ含んでいる
→ 代表点は部分グリッドに含まれたり含まれなかったりする
　含まれていたら除きたい

「一部分だけ含んでいる」＝「その領域は外に繋がっている」ということで、**部分グリッドから外に繋がっているところだけ調べればよい**

　もともとの閉じた領域の代表点がその部分グリッド入っていたら、除く（左上）。そうでないなら除かない（左下）

部分グリッドから外に繋がっている場所は**高々 4N 個しかない！**
　うまく処理すれば O(N) でそういう代表点たちを除ける

まとめ

1. もともと閉じている領域を求め、その代表点を自由に決める O(N^2)
2. 代表点の個数を2次元累積和で前計算 O(N^2)
3. クエリでは、まず「頂点の個数」「辺の個数」「代表点の個数」をそれぞれ求める O(1)
4. 部分グリッドから外に繋がっている領域を見て、もし代表点が部分グリッドに入っていたら「代表点の個数」から 1 を引く O(N)
5. 最終的に「代表点の個数」は「領域の個数」-1（外の領域に注意！）となる
6. **a = v - e + r - 1** の式から答えを求める

時間厳しくてすみません
Pythonは解けないと思います

前計算	時間: O(N^2)	空間: O(N^2)
クエリ	時間: O(N)	空間: O(N)
合計	時間: O(N^2 + NQ)	空間: O(N^2)

K - Two-Way Communication

提案 : tatyam
準備 : tatyam
tester : hotman78 (CTF)

ふたつの交通機関 (Two Transportations)

- JOI 春合宿 2019 Day 2 問題 3
- N 頂点の重み付き無向グラフがあるが、辺の情報は Alice と Bob に分けて与えられる
- 頂点 1 から各頂点までの最短距離を求めよ
- 何ビットの通信があれば最短距離を求められるか？
  - Communication Complexity という概念 (最近研究で見つけた)
- 本番満点を取った思い出の問題だが、久々に見たらさらに改善できる
- JOI の Communication 問題をやろう！

部分問題

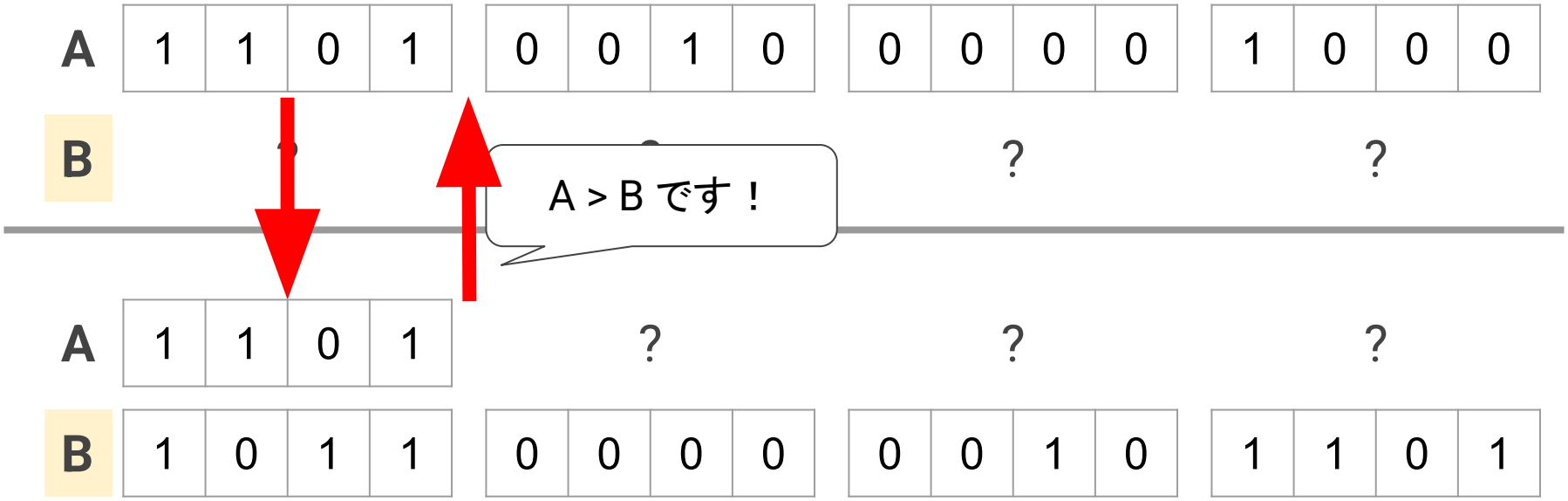
- Alice が非負整数 A, Bob が非負整数 B を持っている
- 何ビットの通信で、双方が、A < B かどうかと C := min(A, B) を特定できるか？

平方分割

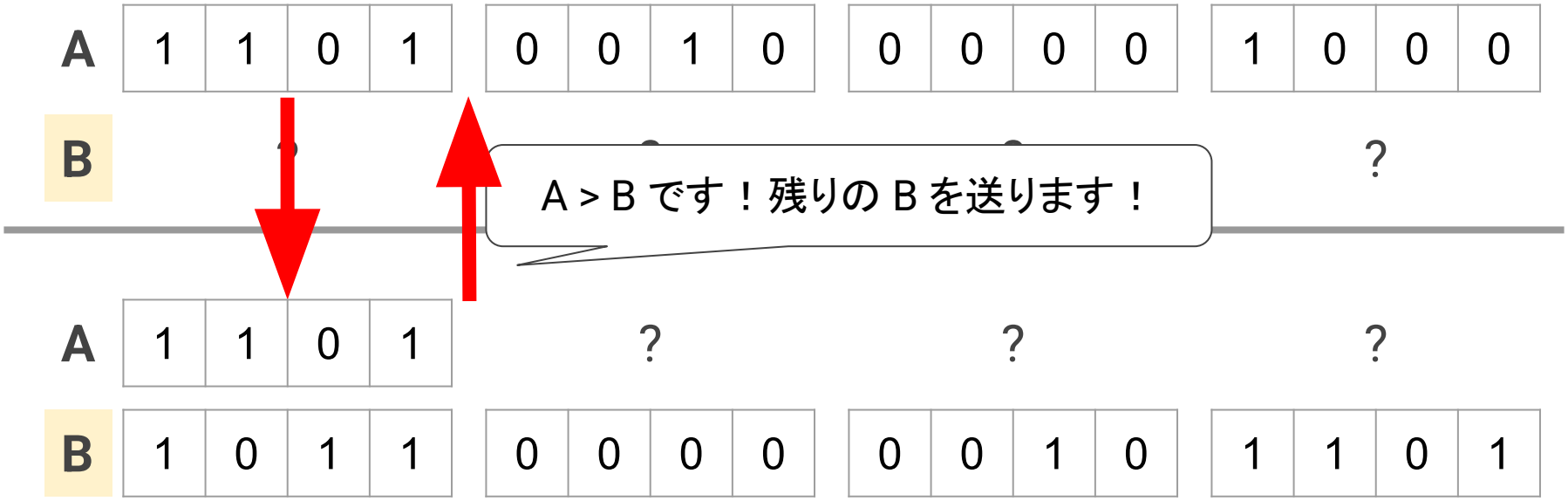
A	1	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0
B	?				?				?				?			

A	?				?				?				?			
B	1	0	1	1	0	0	0	0	0	0	1	0	1	1	0	1

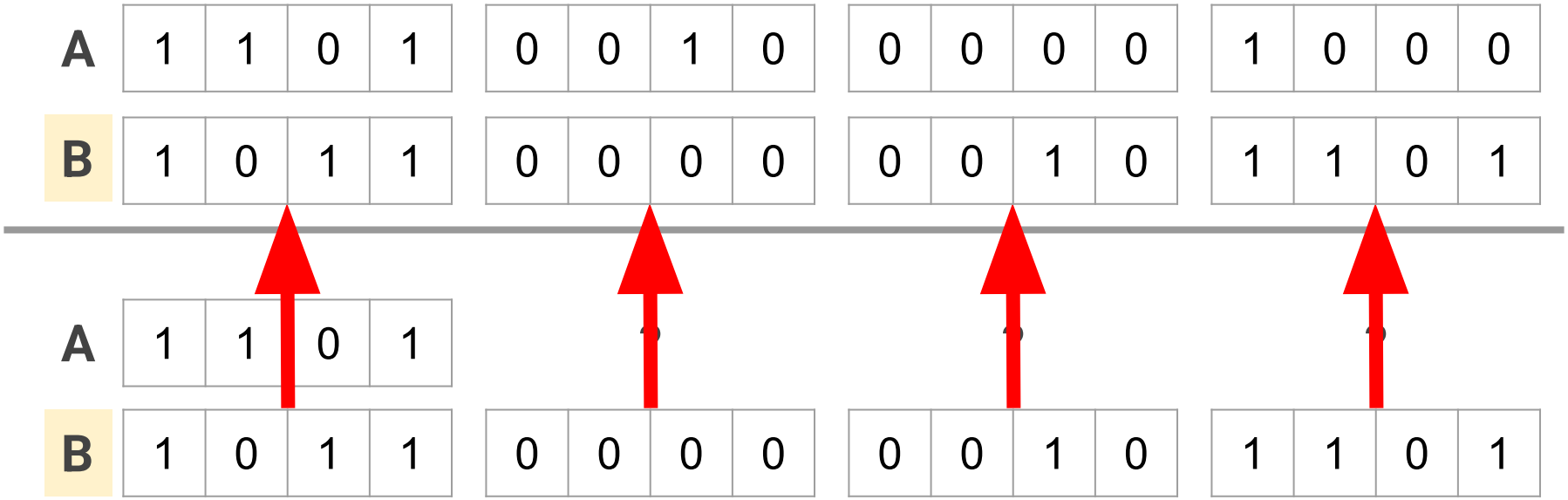
平方分割



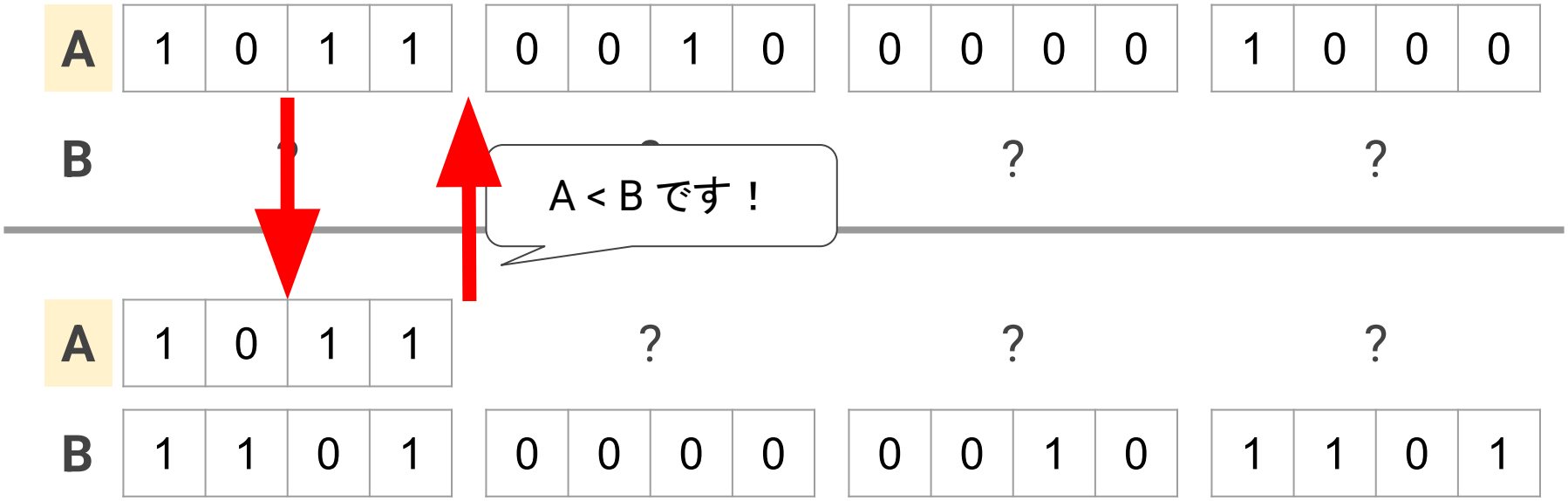
平方分割


A
1 1 0 1
0 0 1 0
0 0 0 0
1 0 0 0
B
?
A > B です！残りの B を送ります！
A
1 1 0 1
? ? ?
B
1 0 1 1
0 0 0 0
0 0 1 0
1 1 0 1

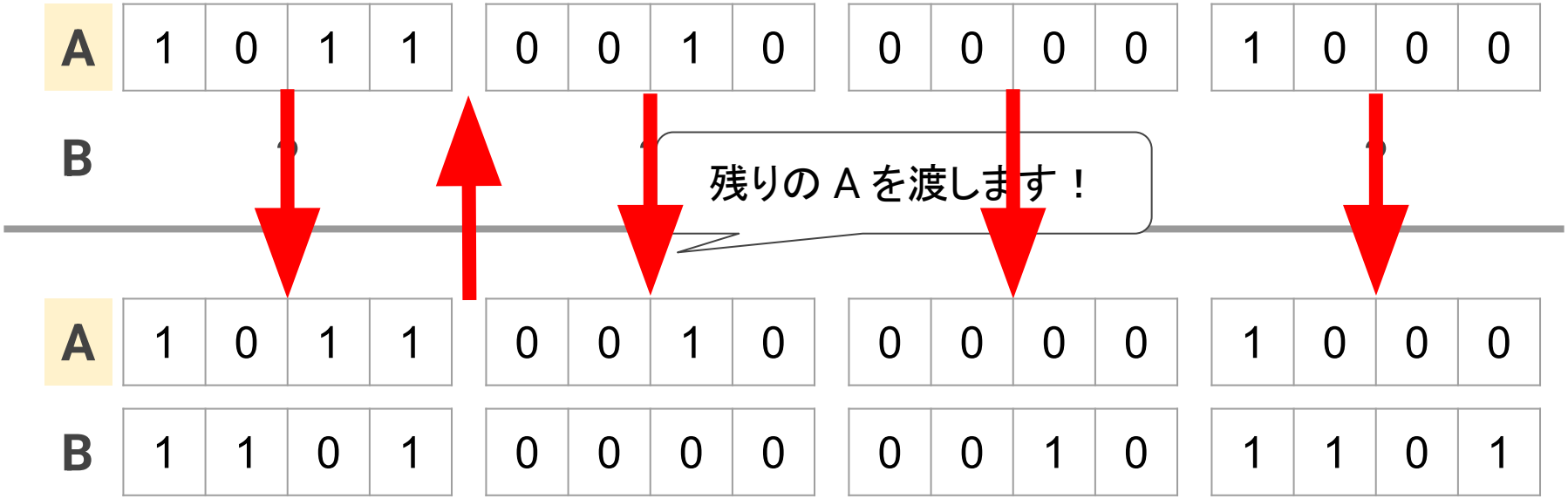
平方分割


A
1 1 0 1
0 0 1 0
0 0 0 0
1 0 0 0
B
1 0 1 1
0 0 0 0
0 0 1 0
1 1 0 1
A
1 1 0 1
B
1 0 1 1
0 0 0 0
0 0 1 0
1 1 0 1

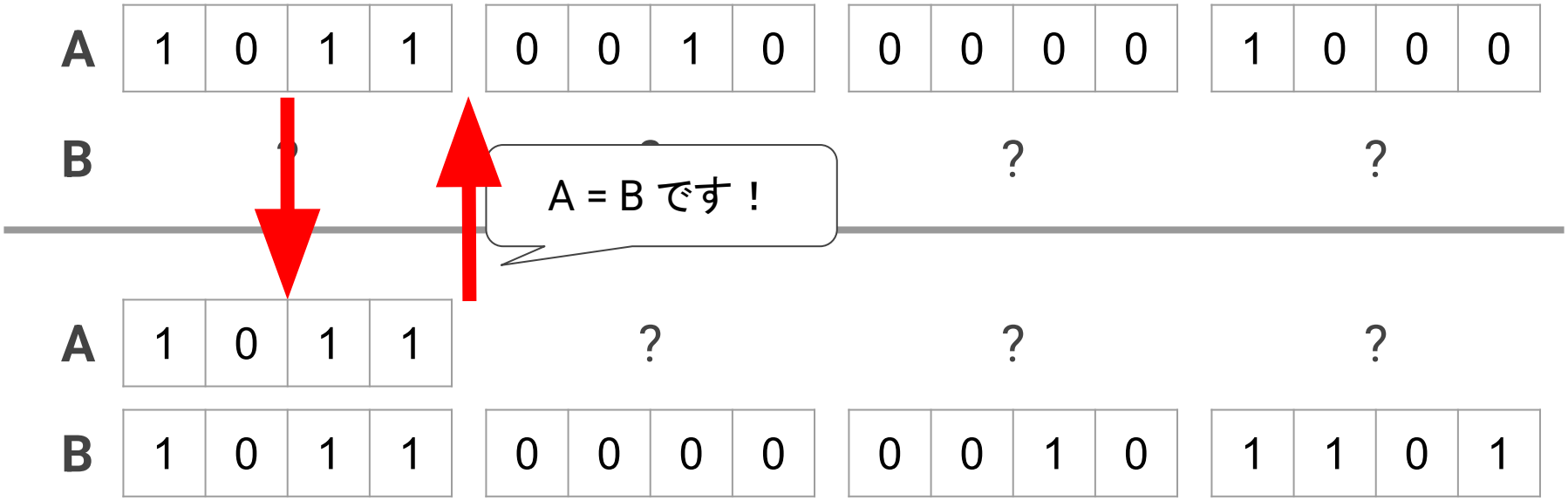
平方分割



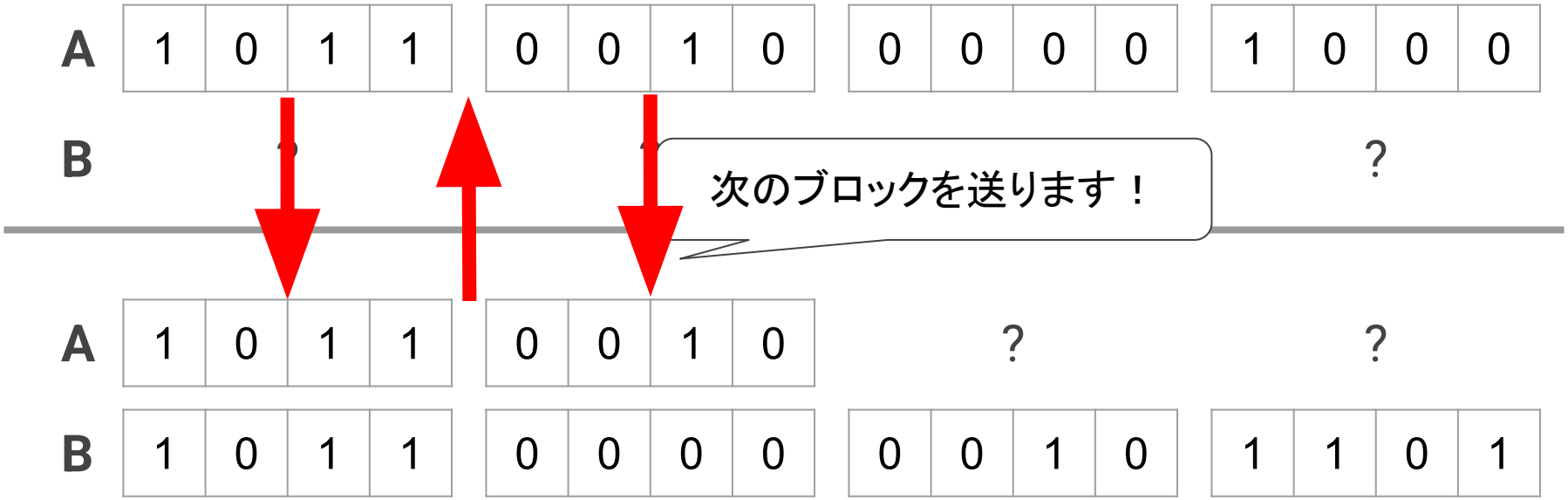
平方分割


A
1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0
B
残りの A を渡します！
A
1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0
B
1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 1

平方分割



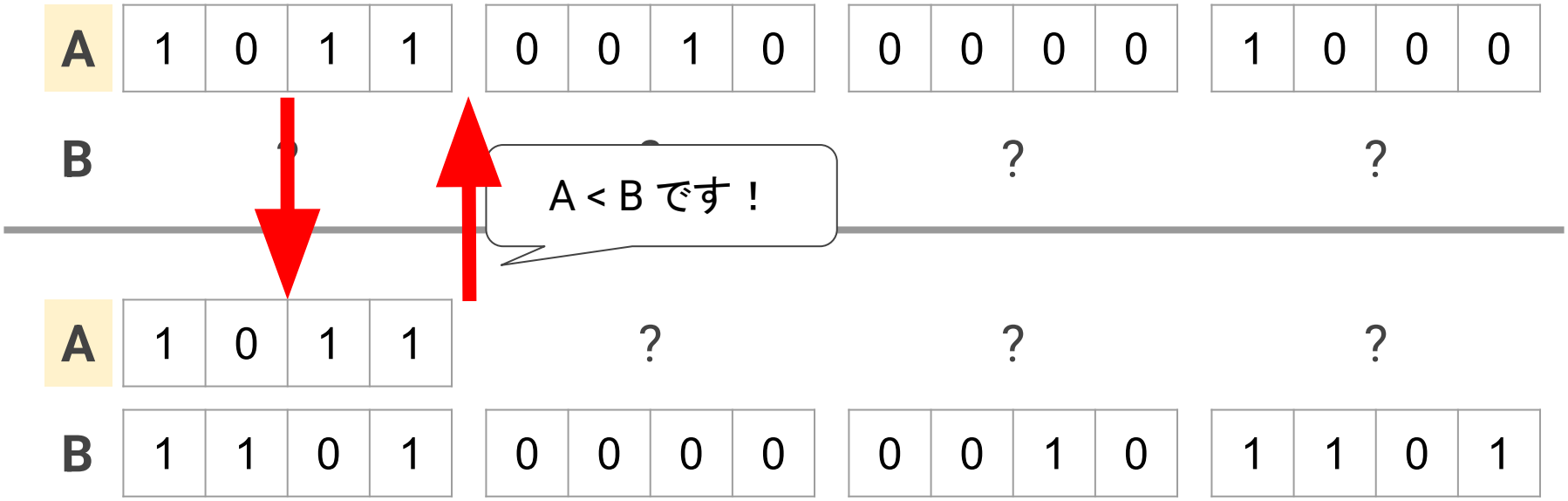
平方分割


A
1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0
B
?
?
?
次のブロックを送ります！
A
1 0 1 1 0 0 1 0 ? ?
B
1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 1

クエリ回数

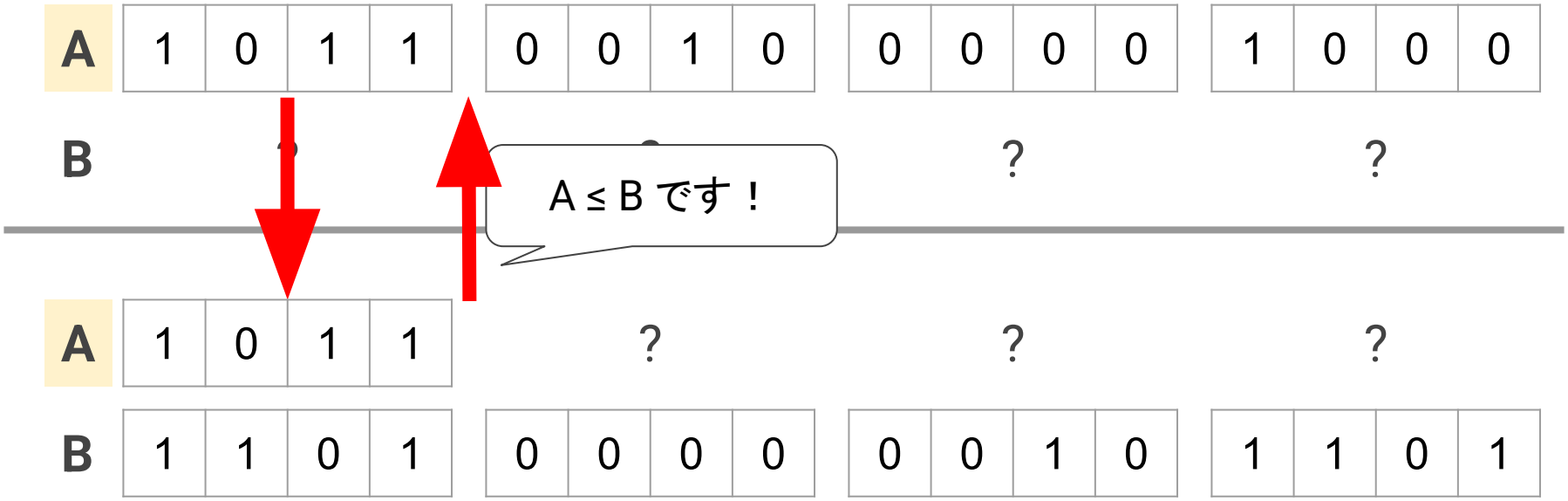
- 64 bit の共有 : 64 回
- ブロックの値が同じかどうか返答 : 8 回
- ブロックの値が異なる場合、どちらが小さいか : 1 回
- A > B だった場合、ブロックの再送信 : 8 回
- 64 + 8 + 1 + 8 = 81 回 (66 点)

無駄なところ？

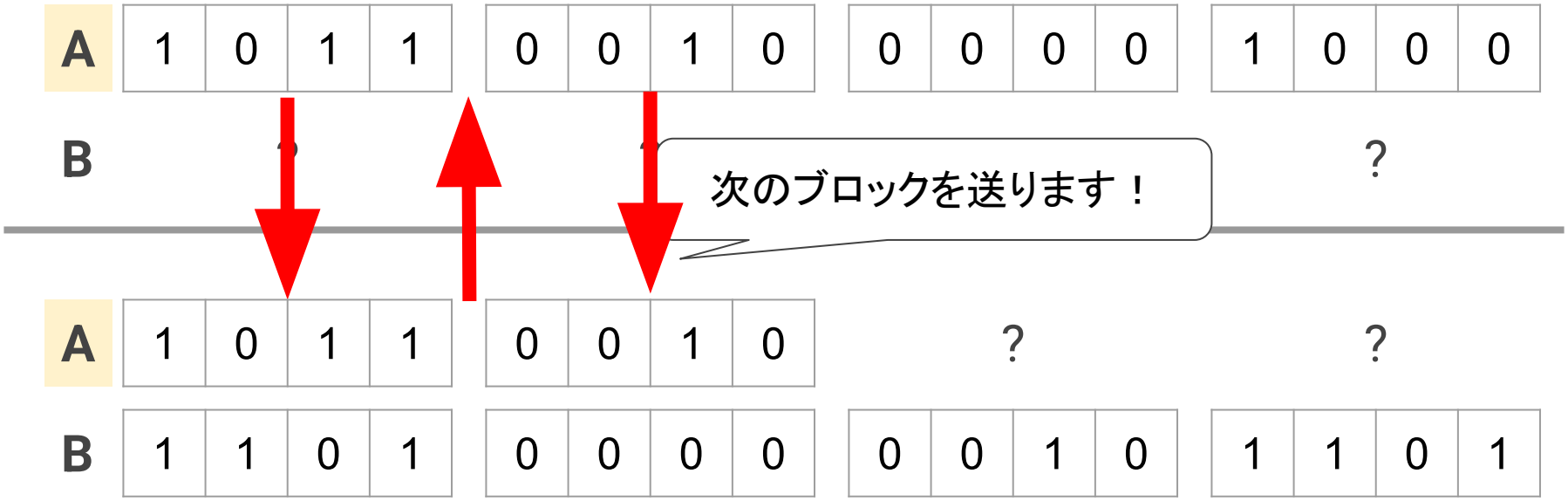


- A = B と返答しても A の値を送ってもらえる！

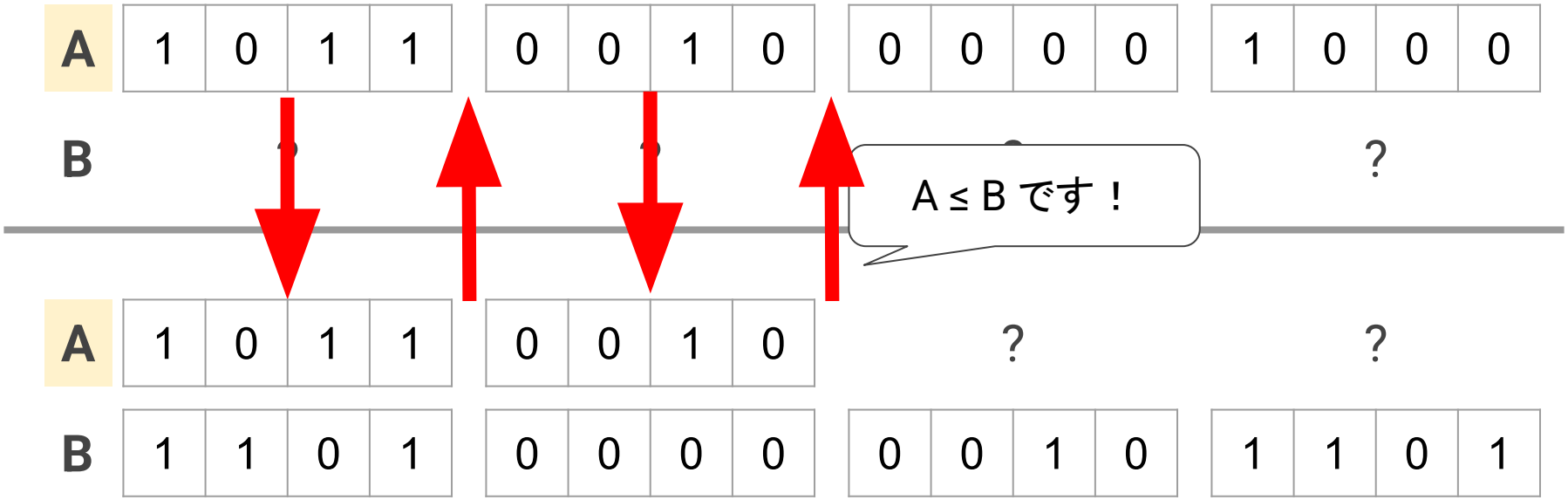
改善



改善


A
1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0
B
? ? ?
次のブロックを送ります！
A
1 0 1 1 0 0 1 0 ? ?
B
1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 1

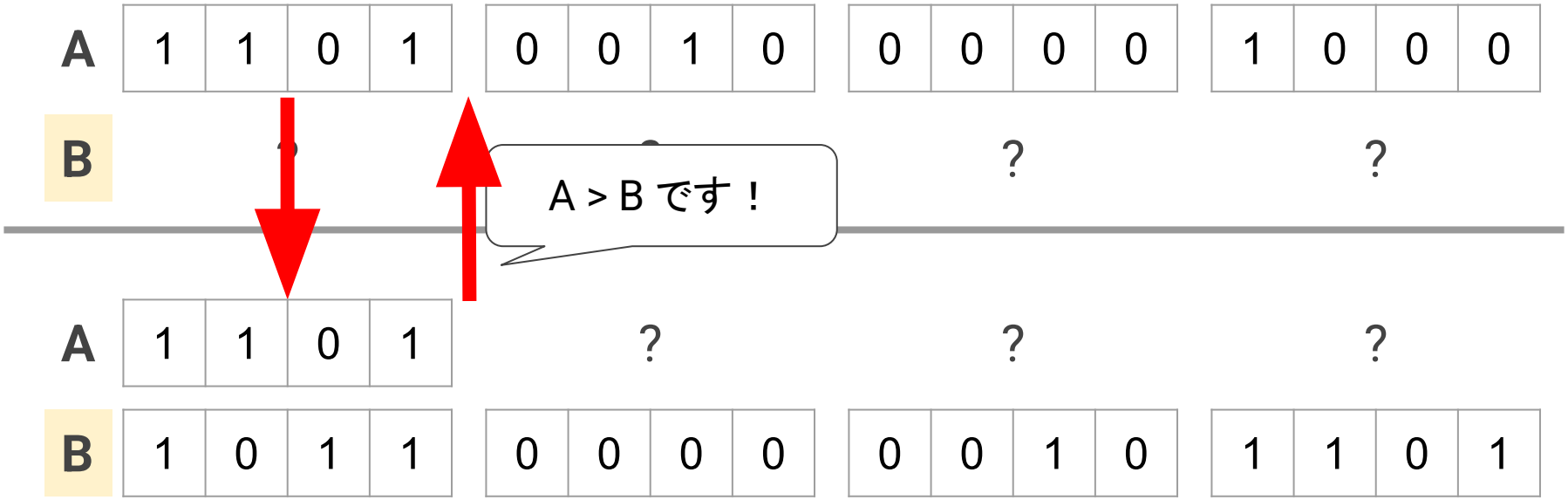
改善



クエリ回数

- 64 bit の共有 : 64 回
- ここまでで A ≤ B かどうか返答 : 8 回
- A > B だった場合、ブロックの再送信 : 8 回
- 64 + 8 + 8 = 80 回 (70 点)

本当に平方分割が最適解？



- 早く A > B が判明すると、その後の B からの応答がなくなり、80 回も使っていない！

クエリ回数

- 64 bit の共有 : 64 回
- ここまでで A ≤ B かどうか返答 : (ここまでのブロックの個数) 回
- A > B だった場合、ブロックの再送信 : (このブロックの長さ) 回
- 最後のブロックまで A ≤ B であった場合、64 + 8 + 8 = 80 回 (70 点)

クエリ回数

- 64 bit の共有 : 64 回
- ここまでで A ≤ B かどうか返答 : (ここまでのブロックの個数) 回
- A > B だった場合、ブロックの再送信 : (このブロックの長さ) 回
- 最後のブロックまで A ≤ B であった場合、64 + 8 + 8 = 80 回 (70 点)

- **後ろのブロックほど短くすべき！**

1ビットずつ減らしていく

A	1	1	0	1	0	0	1	0	0	0	0	0	1	0	0
B	?					?					?				

A	?					?					?				
B	1	0	1	1	0	0	0	0	0	0	1	0	1	1	0

満点解法

- 1 個目のブロックで A > B となった場合、64 + 1 + 11 = 76 回
- 2 個目のブロックで A > B となった場合、64 + 2 + 10 = 76 回
- 3 個目のブロックで A > B となった場合、64 + 3 + 9 = 76 回
- 9 個目のブロックで A > B となった場合、64 + 9 + 3 = 76 回
- 10 個目のブロックで A > B となった場合、64 + 10 + 1 = 75 回
- 76 回で 100 点