

CPSCO 2019 session3

A - ASOKO

原案: drken

解答: drken, tempura

問題概要

- ・ 5 文字の文字列 S が与えられる
- ・ S 中の ‘A’ を ‘O’ に、 ’O’ を ‘A’ にせよ
- ・ 例: “OSAKA” -> “ASOKO”

解法 (C++, Python)

- ・ S の各文字について
 - ・ それが ‘A’ だったら ‘O’ に変換し、
 - ・ それが ‘O’ だったら ‘A’ に変換する
- ・ C++, Python では S の 0~4 文字目は S[0], S[1], S[2], S[3], S[4] で表せる
- ・ 例えば S[2] に対する処理は

```
if (S[2] == 'A') S[2] = '0';
else if (S[2] == '0') S[2] = 'A';
```

コード例 1

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string S;
7     cin >> S;
8
9     if (S[0] == 'A') S[0] = '0';
10    else if (S[0] == '0') S[0] = 'A';
11
12    if (S[1] == 'A') S[1] = '0';
13    else if (S[1] == '0') S[1] = 'A';
14
15    if (S[2] == 'A') S[2] = '0';
16    else if (S[2] == '0') S[2] = 'A';
17
18    if (S[3] == 'A') S[3] = '0';
19    else if (S[3] == '0') S[3] = 'A';
20
21    if (S[4] == 'A') S[4] = '0';
22    else if (S[4] == '0') S[4] = 'A';
23
24    cout << S << endl;
25 }
```

for 文ループを使うと楽

- S[0], S[1], S[2], S[3], S[4] に対する処理を統一的に行う (S の文字数が不明でもこれで OK)

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 int main() {
6     string S;
7     cin >> S;
8
9     for (int i = 0; i < S.size(); ++i) {
10         if (S[i] == 'A') S[i] = '0';
11         else if (S[i] == '0') S[i] = 'A';
12     }
13     cout << S << endl;
14 }
```

データ

- ・ ジャッジ解
 - ・ drken (C++): 12 行
 - ・ tempura (C++): 13 行
 - ・ tempura (Bash): 1 行
- ・ First AC (全体): kotatsugame (00:00:14)
- ・ First AC (オンライン): cpsco_I_am_yojo (00:01:06)
- ・ Accepted: 205
- ・ Submitted: 205

Bash 解 (番外編)

```
1 #!/bin/sh
2 tr A0 OA
```

CPSCO 2019 session3

B - Balloons

原案: drken

解答: drken, tempura, gojira

問題概要

- ・ 各色の風船が $a[0], a[1], \dots$ 個ある。
- ・ この中から M 個選ぶ
- ・ M 個の風船の中に含まれる色の種類数を最小にせよ。

解法

- ・ 例えば、 $M = 20$ 、 $a = (6, 3, 4, 8, 1)$ とかだったら
 - ・ a の最大値 8 をまず使って、 M は残り $20 - 8 = 12$ となり、
 - ・ a の二番目に大きい 6 を使って、残り $12 - 6 = 6$ となり、
 - ・ a の三番目に大きい 4 を使って、残り $6 - 4 = 2$ となり、
 - ・ a の四番目に大きい 3 の中から 2 個使えば終了
- ・ 一般には、 a を大きい順にソートして、順に M から引いていけば良い

4 個

実装 (C++)

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 int main() {
7     // 入力
8     long long N, M;
9     cin >> N >> M;
10    vector<long long> a(N);
11    for (int i = 0; i < N; ++i) cin >> a[i];
12
13    // 大きい順にソート
14    sort(a.begin(), a.end(), greater<long long>());
15
16    // M が 0 になるまで a を大きい順に引いて行く
17    int res = 0;
18    for (int i = 0; i < a.size() && M > 0; ++i) {
19        M -= a[i];
20        ++res;
21    }
22    cout << res << endl;
23 }
```

greater<T>() を指定すると
大きい順にソートできる

この手の処理はよくあるので
自分なりの書き方をテンプレ化しよう！

データ

- ・ ジャッジ解
 - ・ drken (C++): 21 行
 - ・ tempura (C++): 19 行
- ・ First AC (全体): th90tk297 (00:01:07)
- ・ First AC (オンライン): cpsco_socha (00:04:48)
- ・ Accepted: 200
- ・ Submitted: 203

CPSCO 2019 session3

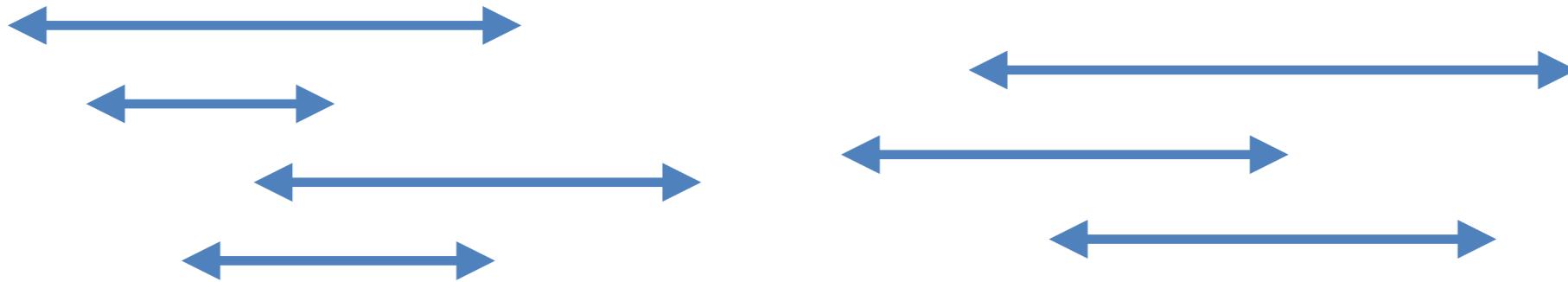
C - Camp Reception

原案: drken

解答: drken, tempura, gojira

問題概要

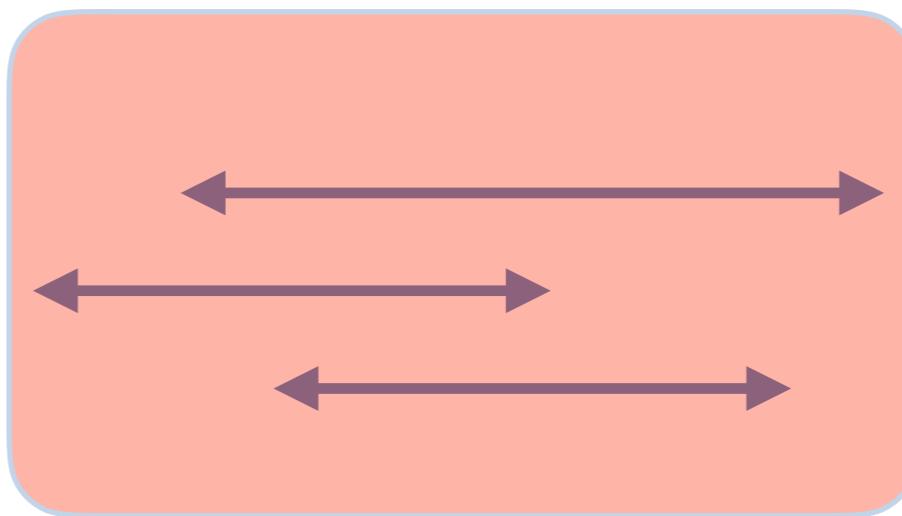
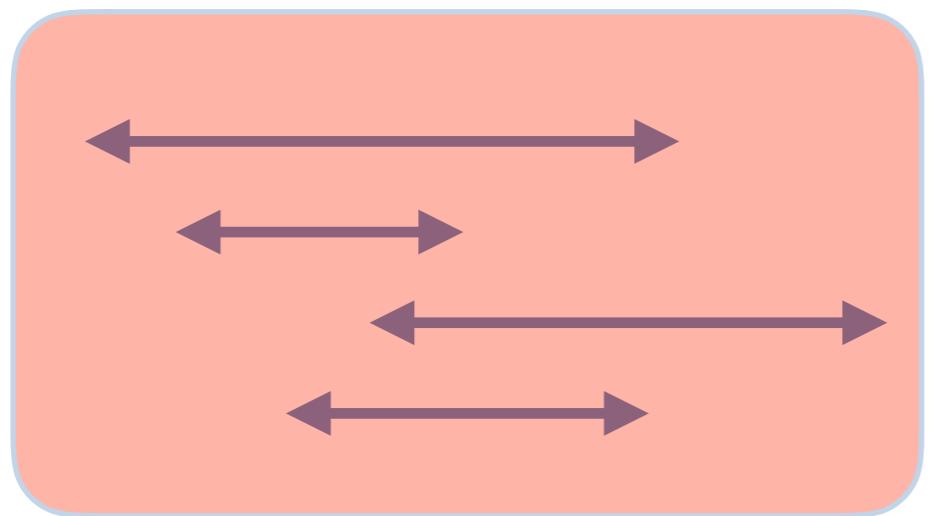
- N 個の区間がある。



- 区間が重なり合っているところをグループとみなしたとき、区間全体は何グループに分かれるか？

問題概要

- N 個の区間がある。



- 区間が重なり合っているところをグループとみなしたとき、区間全体は何グループに分かれるか？

二通りの解法

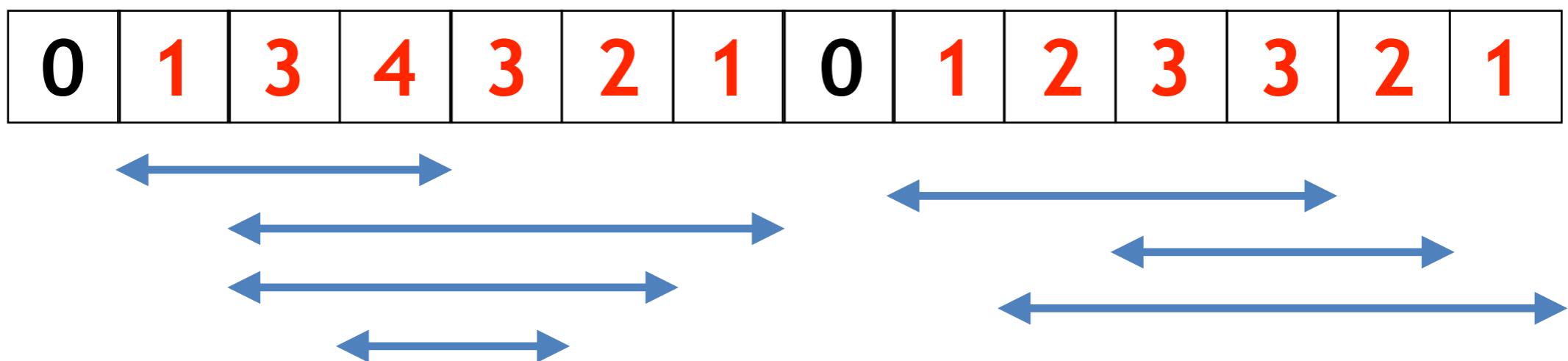
- ・ いもす法 + 区間分割
- ・ 区間を適切にソートして処理

二通りの解法

- ・ いもす法 + 区間分割
- ・ 区間を適切にソートして処理

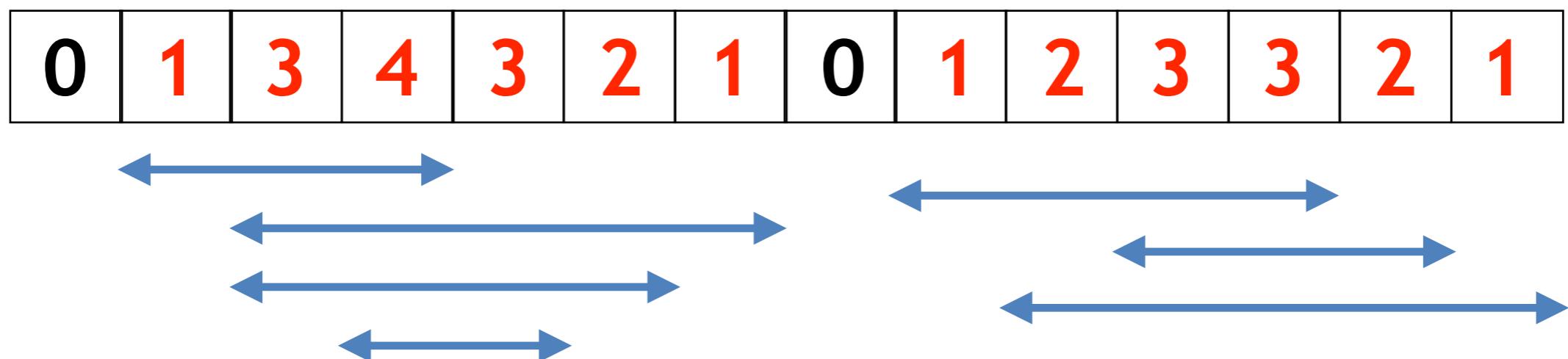
今回やりたいこと

- ・ 大きな配列を用意して、与えられた各区間に属する部分全体に 1 を足していく
- ・ 最後に 0 で区切り、何箇所切り取られるか数える

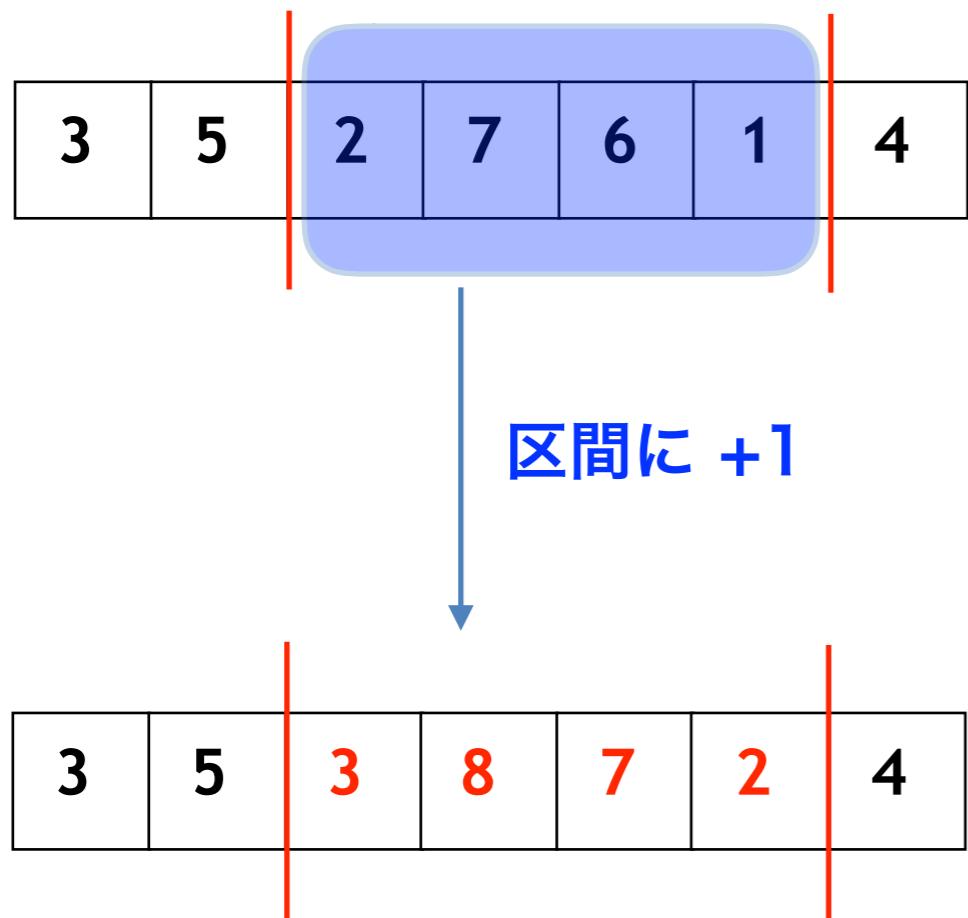


しかし

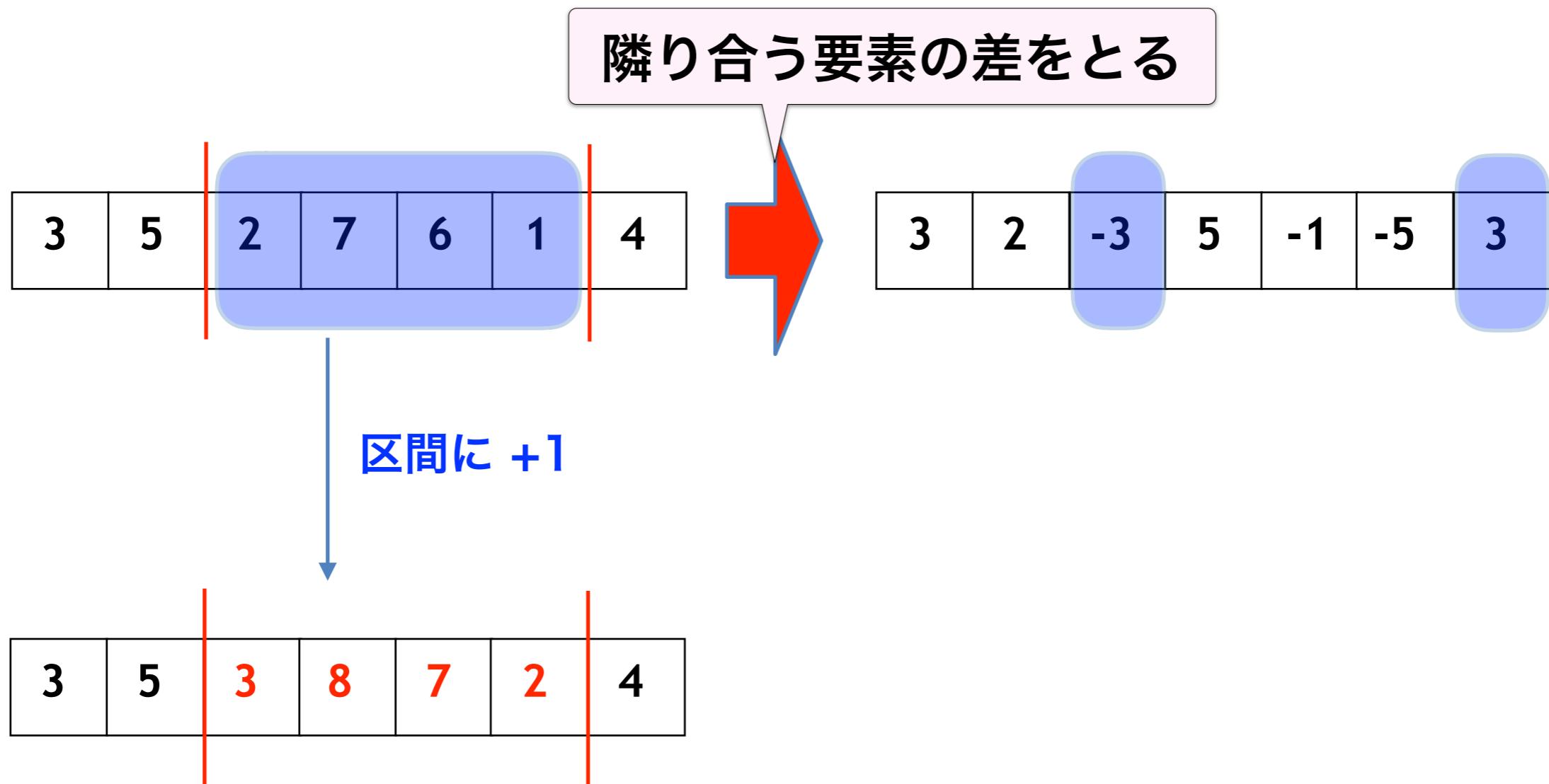
- しかし、 N 個の区間すべてについて、愚直に区間全体に 1 を足すのでは、計算時間がかかりすぎる
- 配列サイズを M として、 $O(NM)$



いもす法: 世界をchange

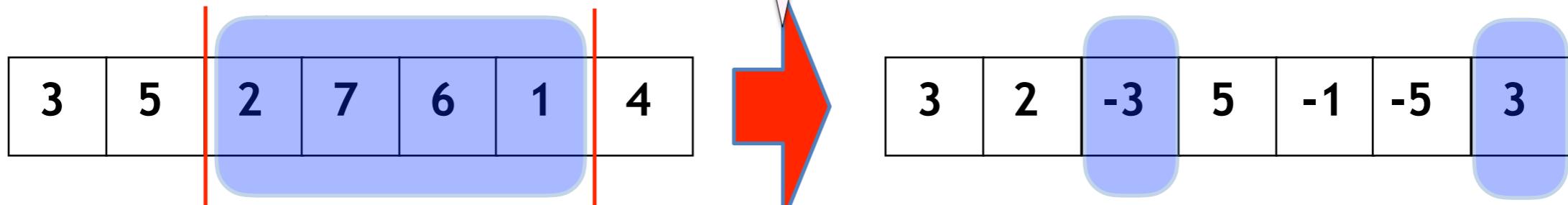


いもす法: 世界をchange

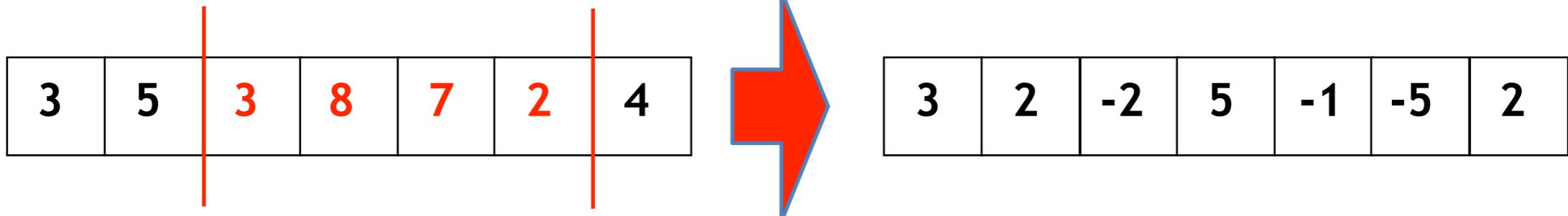


いもす法: 世界をchange

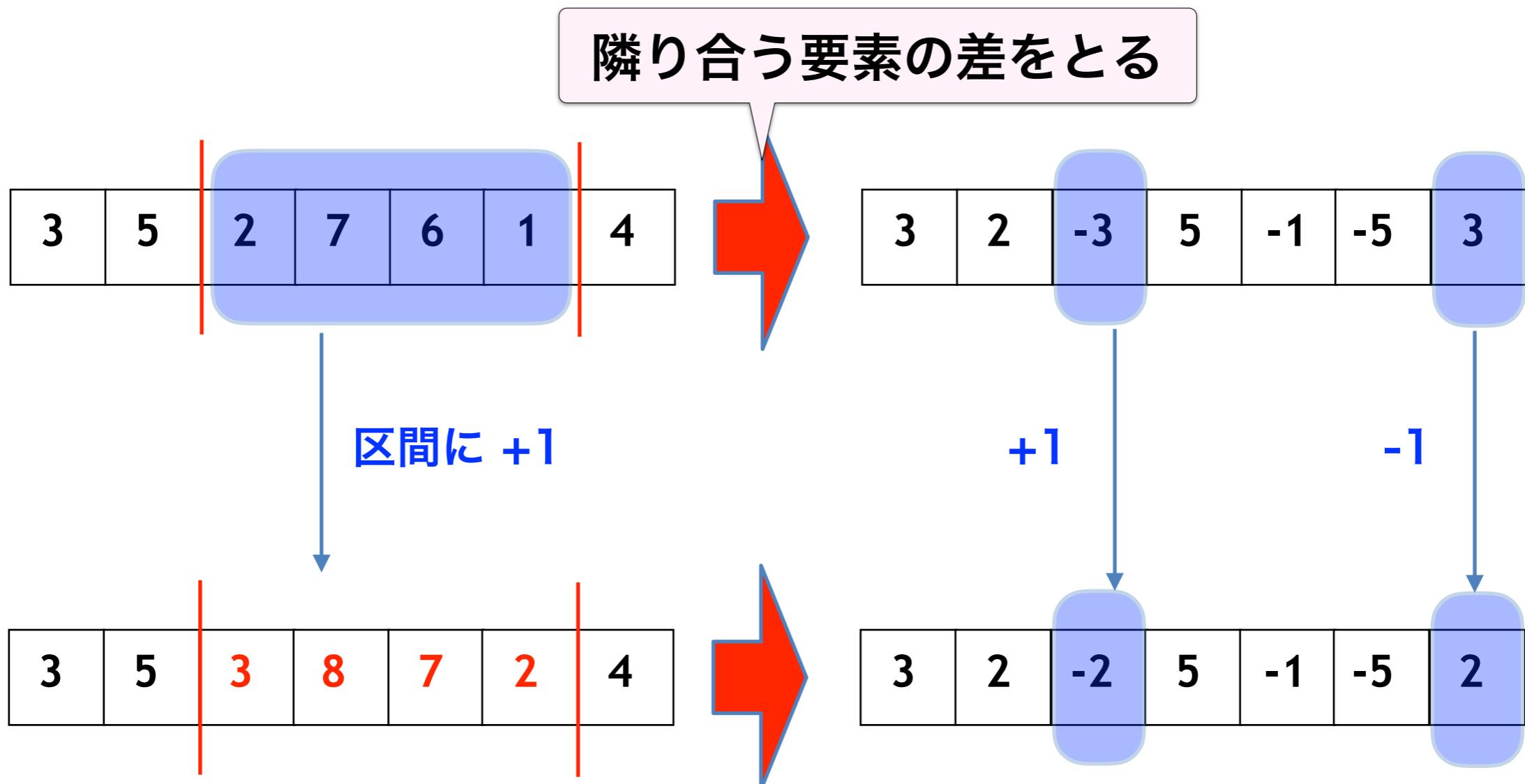
隣り合う要素の差をとる



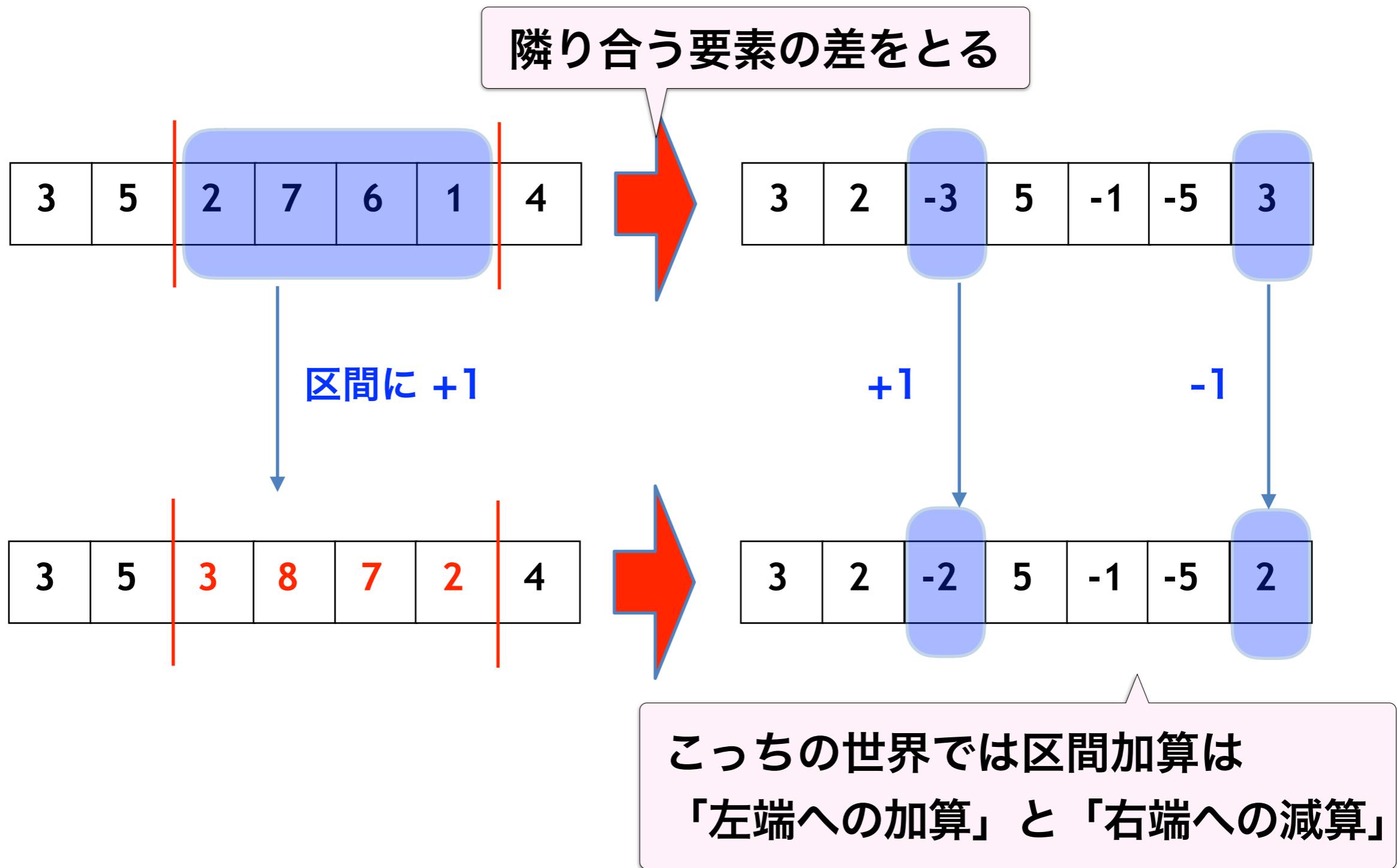
区間に +1



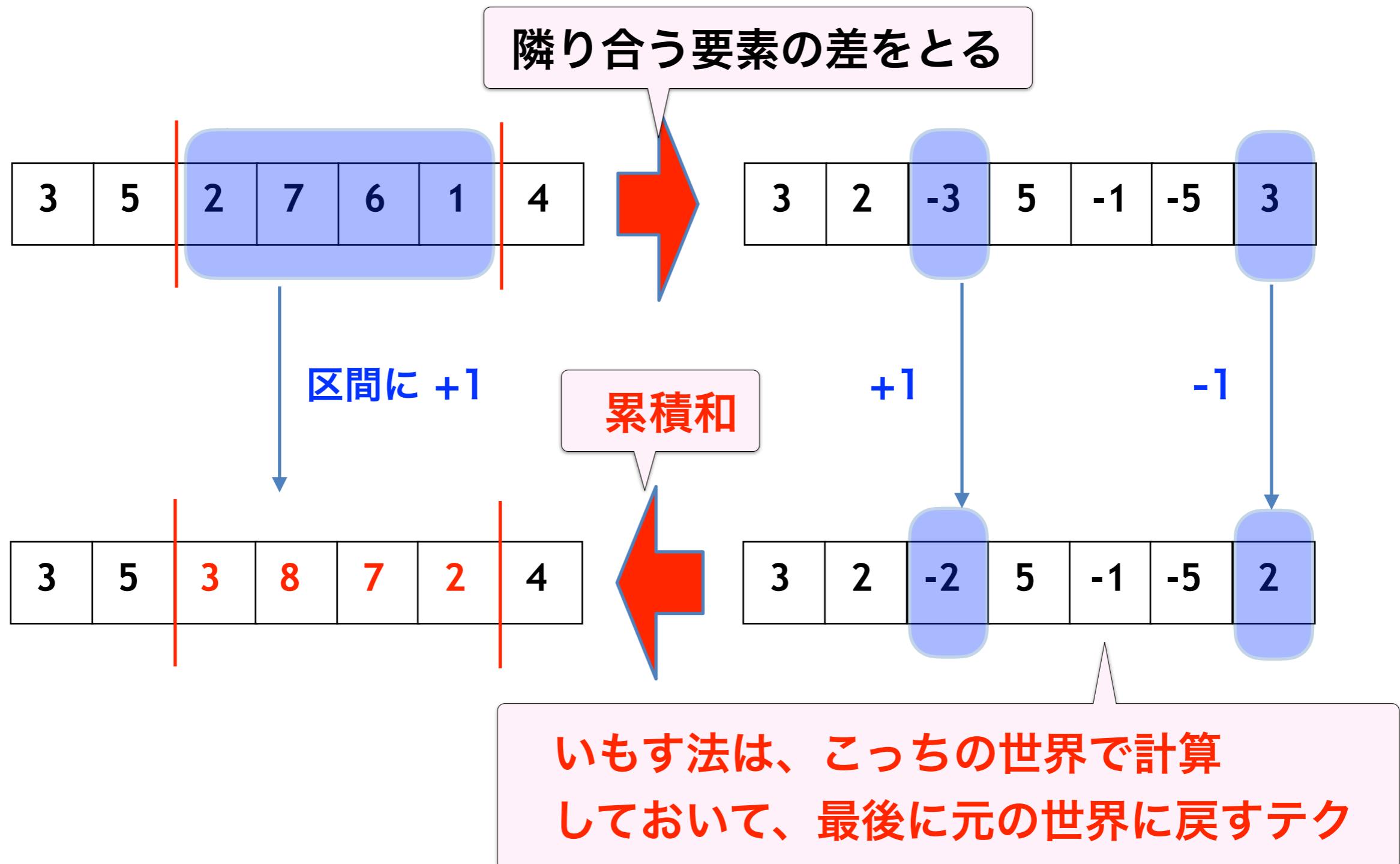
いもす法: 世界をchange



いもす法: 世界をchange

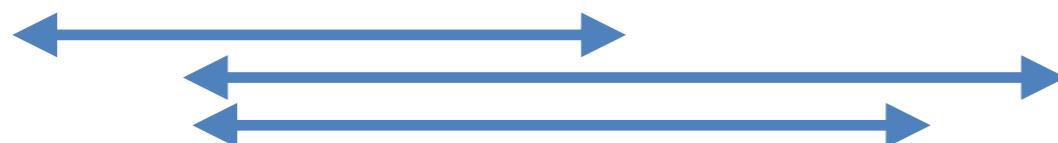


いもす法: 世界をchange



いもす法の例

0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---



0	1	0	0	0	-1	0	0	0	0
---	---	---	---	---	----	---	---	---	---



0	1	1	0	0	-1	0	0	-1	0
---	---	---	---	---	----	---	---	----	---



0	1	2	0	0	-1	0	-1	-1	0
---	---	---	---	---	----	---	----	----	---



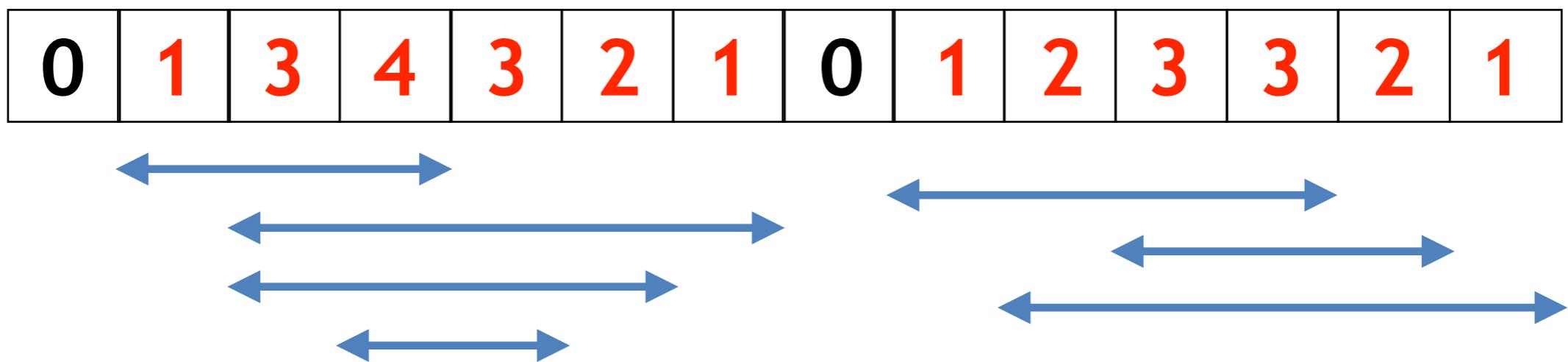
差をとった世界

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

最後に累積和

いもす法での解法まとめ

- ・ いもす法して、下図のように求めて
- ・ 0で区切る
- ・ 計算量はいもす法に $O(N + M)$ 、0区切りに $O(N)$



いもす法コード

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4 const int MAX = 1001010;
5
6 int main() {
7     int N; cin >> N;
8     vector<int> nums(MAX+1, 0);
9     for (int i = 0; i < N; ++i) {
10         int s, t; cin >> s >> t;
11         nums[s]++; nums[t]--;
12     }
13     for (int i = 0; i < MAX; ++i) nums[i+1] += nums[i];
14     int res = 0;
15     for (int i = 0; i < MAX;) {
16         if (nums[i] == 0) {
17             ++i;
18             continue;
19         }
20         while (i < MAX && nums[i] > 0) ++i;
21         ++res;
22     }
23     cout << res << endl;
24 }
```

いもす法

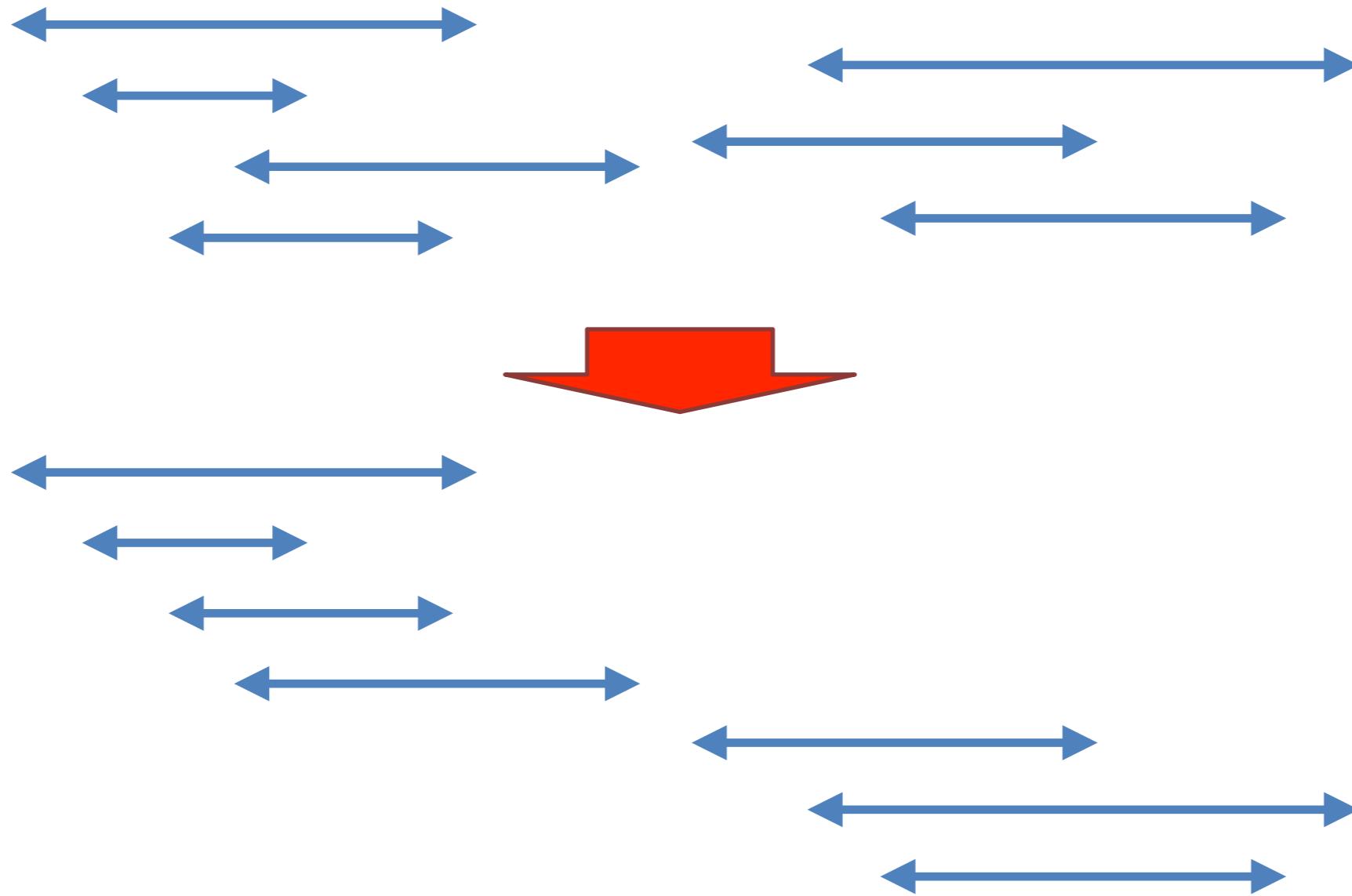
この手の区間を区切る処理もよくある
のでテンプレ化しよう！

二通りの解法

- ・ いもす法 + 区間分割
- ・ 区間を適切にソートして処理

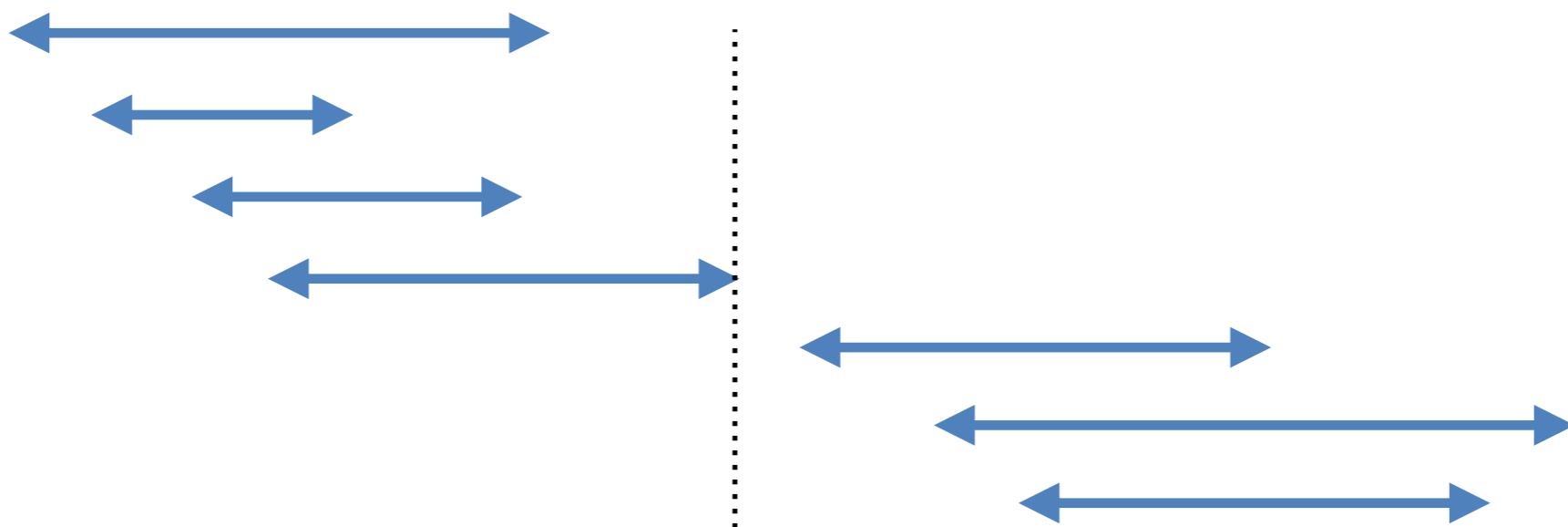
区間を並び替える

- 区間をの始点順にソート



区間を並び替える

- 区間を順番に見て行って、新たな区間の始点が「今までの区間の終端の最大値」よりも先にあったら、そこでグループを切る



区間ソートコード

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5 using pint = pair<int,int>;
6
7 int N;
8 vector<pint> a;
9
10 long long solve() {
11     sort(a.begin(), a.end(), [&](pint i, pint j) {
12         return i.first < j.first;});
13     int res = 0;
14     int cur = 0;
15     for (int i = 0; i < N; ++i) {
16         if (cur < a[i].first) ++res;
17         cur = max(cur, a[i].second);
18     }
19     return res;
20 }
21
22 int main() {
23     cin >> N;
24     a.resize(N);
25     for (int i = 0; i < N; ++i) cin >> a[i].first >> a[i].second;
26     cout << solve() << endl;
27 }
```

区間の始点でソート

cur = 今までの区間の終点の最大値

データ

- ジャッジ解
 - drken (C++): 27 行
 - tempura (C++): 21 行
 - gojira (C++): 37 行
- First AC (全体): latte0119 (00:02:43)
- First AC (オンライン): cpsco_I_am_yojo (00:08:58)
- Accepted: 175
- Submitted: 184

CPSCO 2019 session3

D - Decode RGB Sequence

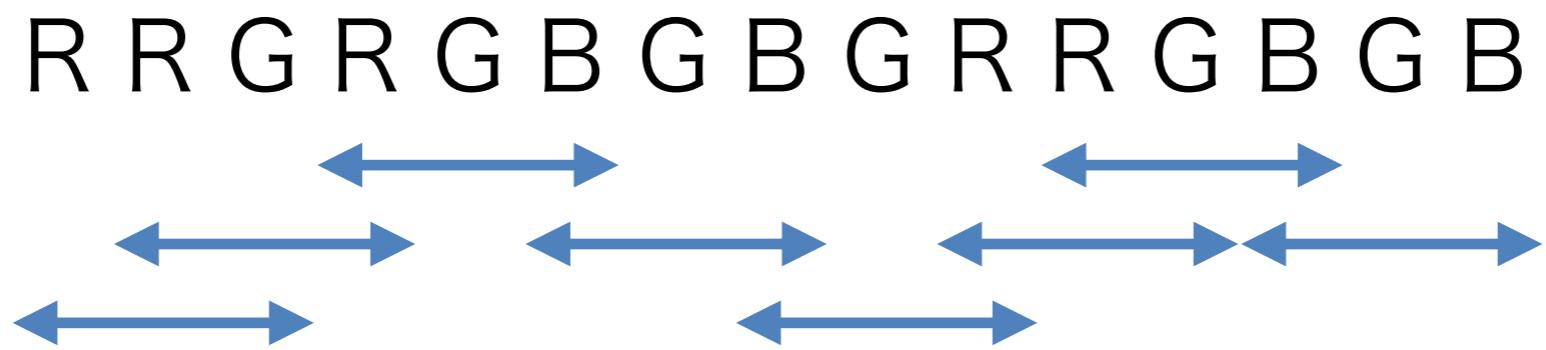
原案: drken

解答: drken, tempura, gojira

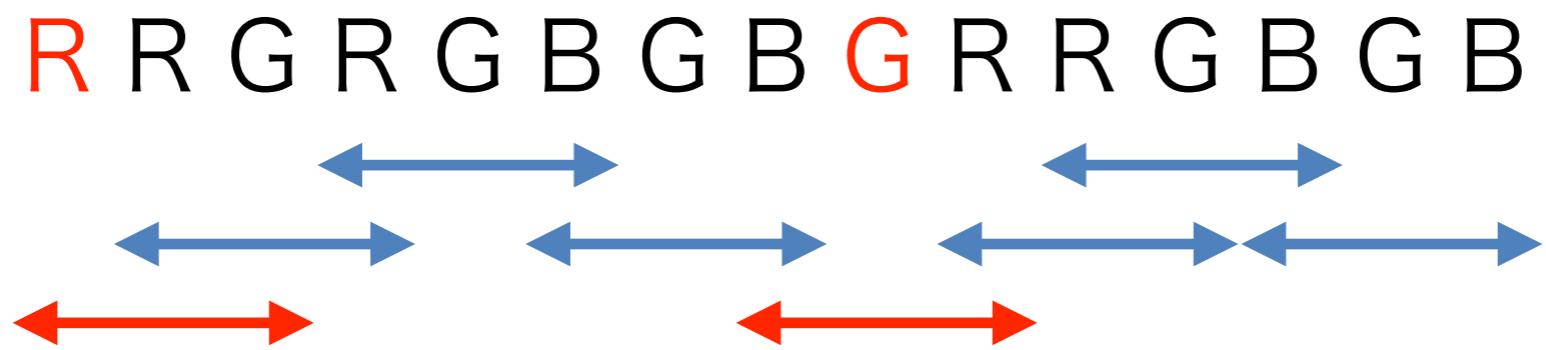
問題概要

- ・ 長さ N の配列の連続する 3 マスを “RGB” に上書きしていく
- ・ RGBBBGRRGB のような文字列が与えられるので、これを作れるかを判定せよ

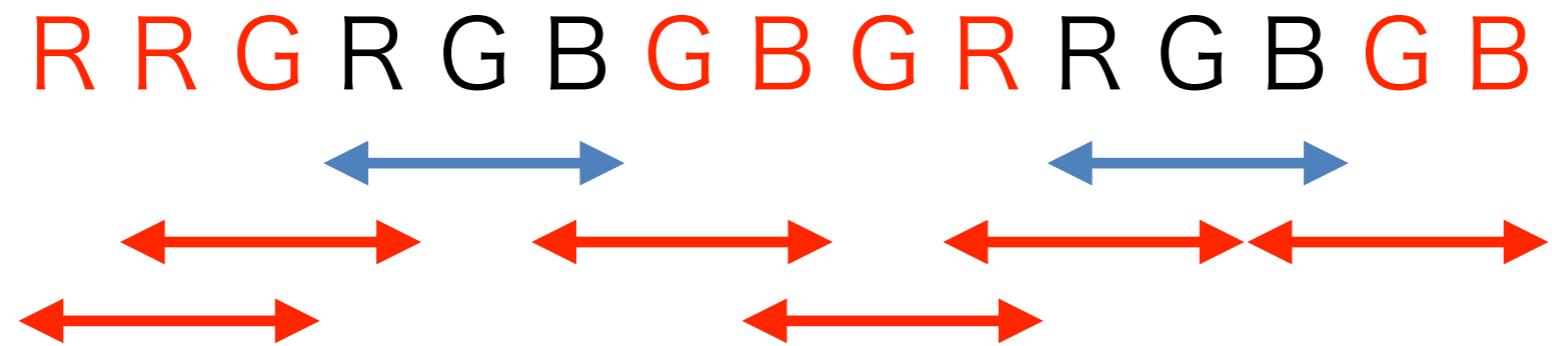
操作の意味を考える



操作の意味を考える



操作の意味を考える

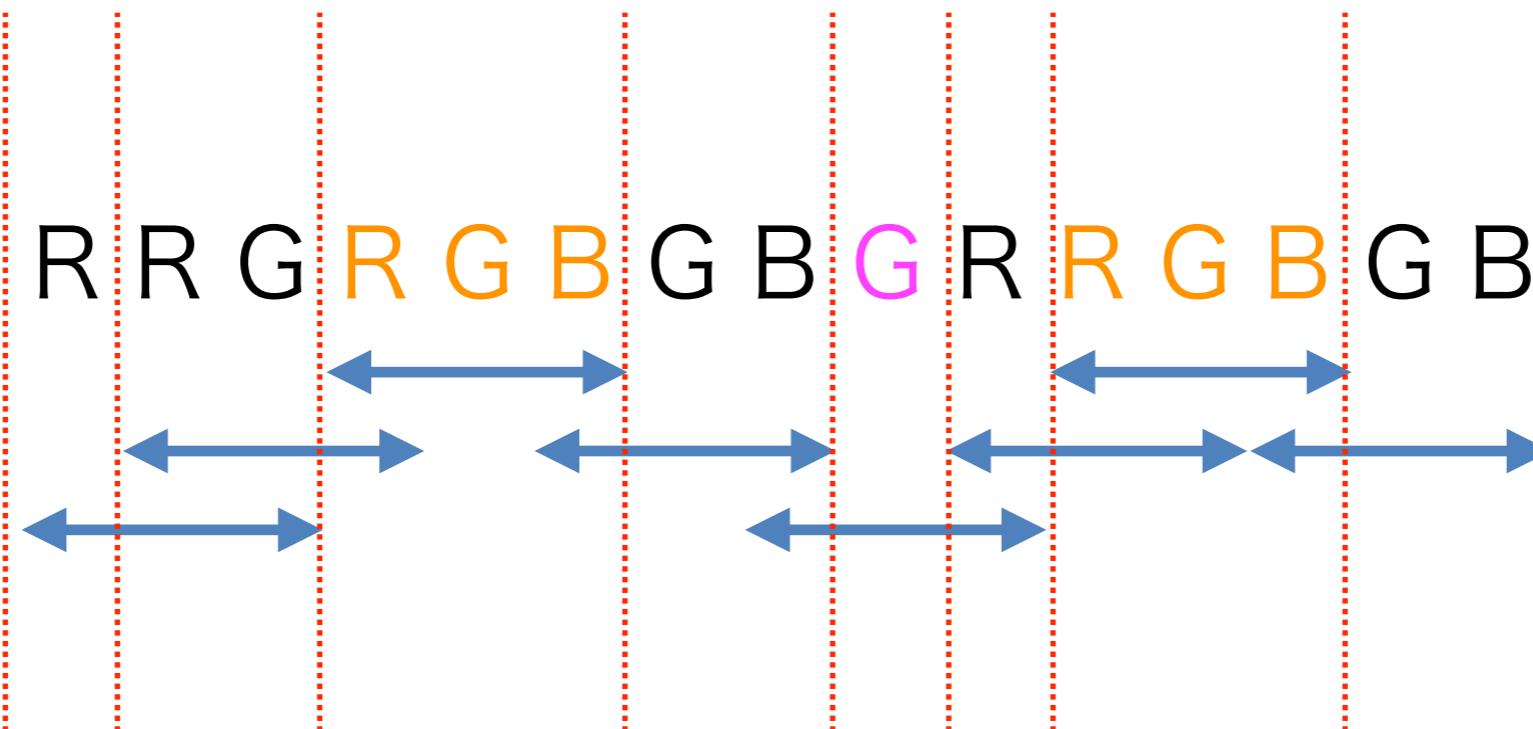


操作の意味を考える

RRGRGBGBGRRGBGB

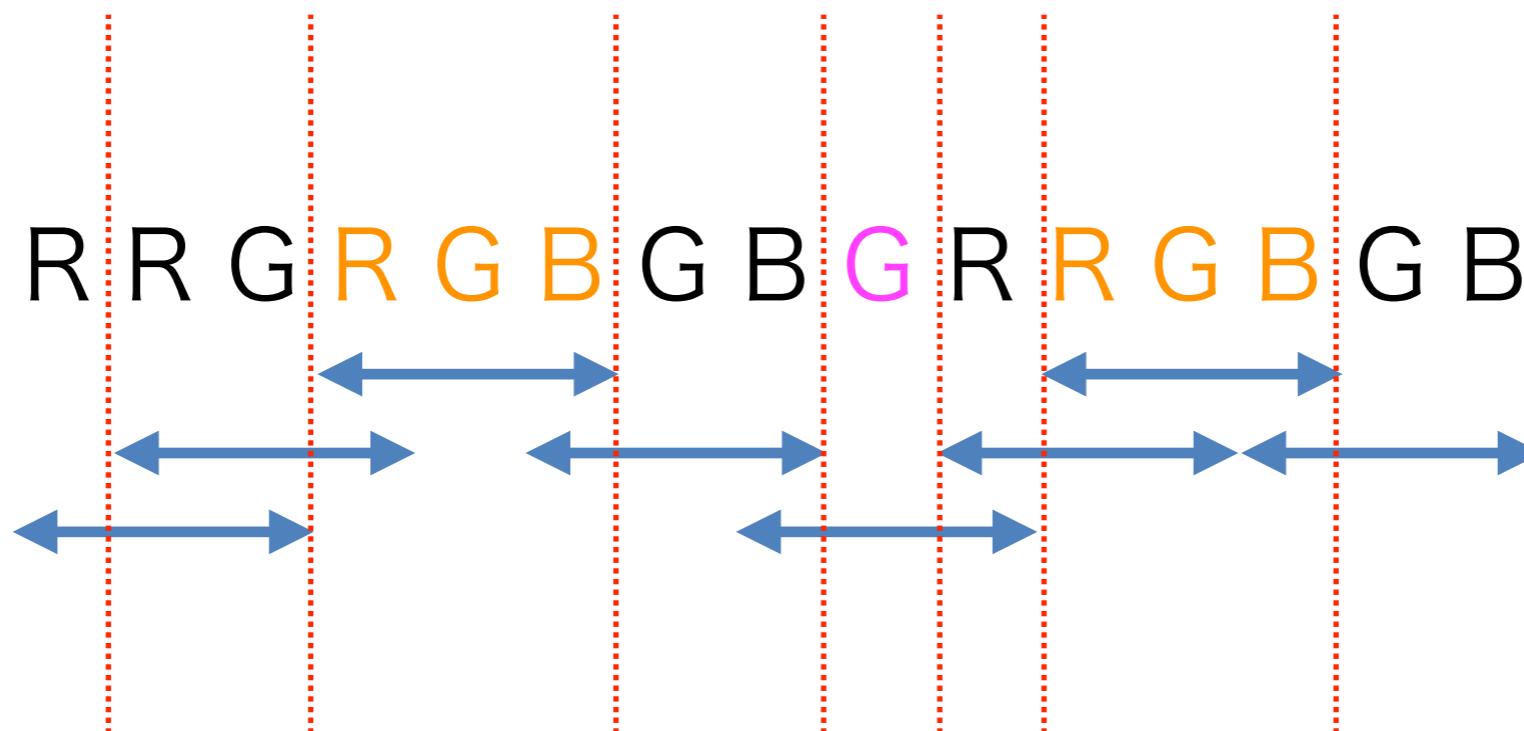
```
graph TD; R1[RR] --> R2[RG]; G1[RB] --> R3[RB]; G2[RB] --> R4[RB]; G3[RB] --> R5[RB]; G4[RB]
```

操作の意味を考える



- RGB の左側は “R” や “RG”
- RGB の右側は “B” や “GB”
- RGB と RGB に挟まれたところは单発 “G” も OK

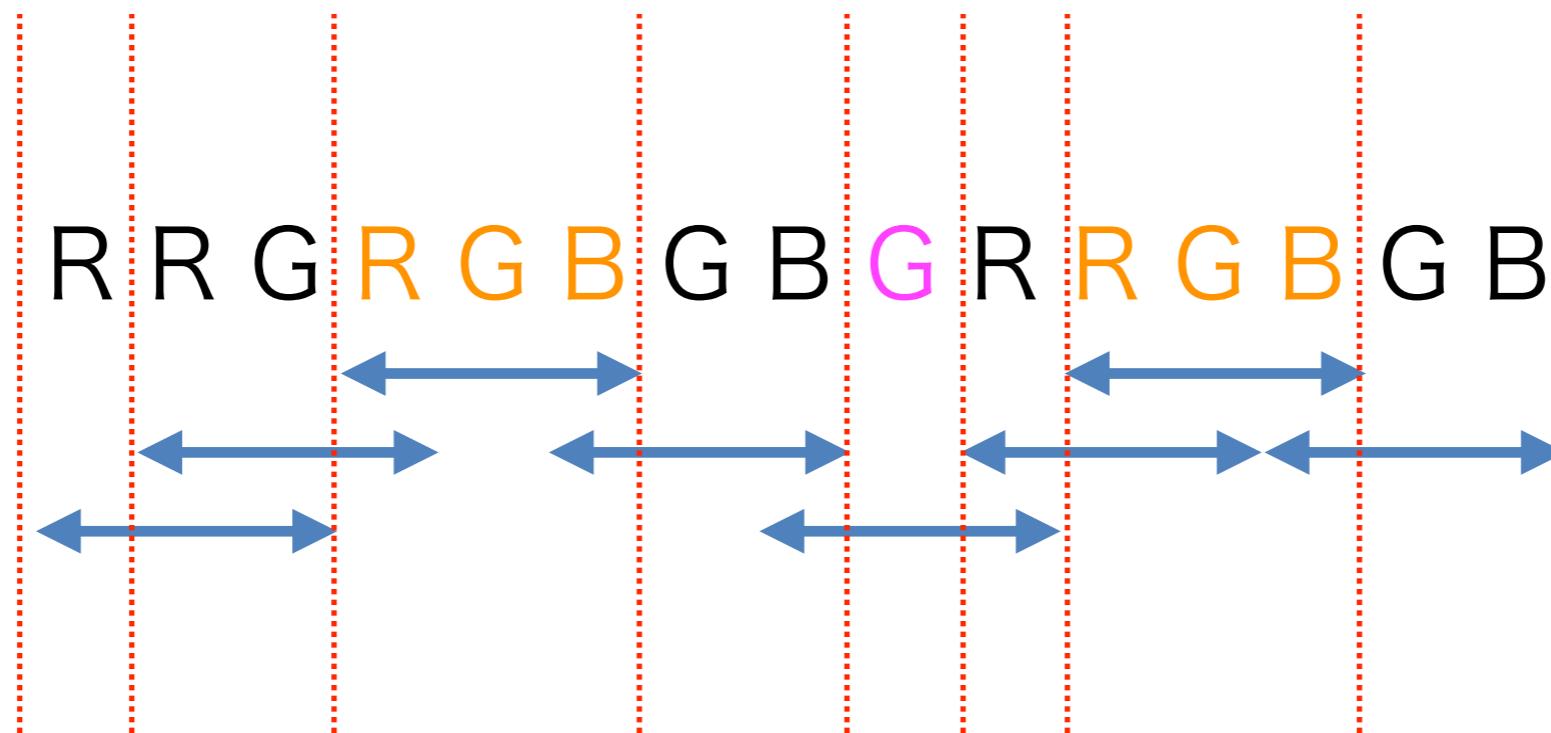
解法1



- RGB の左側へ “R” や “RG” を切り取っていく
- RGB の右側へ “B” や “GB” を切り取っていく
- RGB と RGB の間が消えるか、単発の “G” なら OK

解法2: 実はすごく簡単な特徴づけ

- 以下のすべてを満たせば OK、満たさなかったら NG
 - R で始まり、B で終わる
 - “RB” と “GG” を含まない

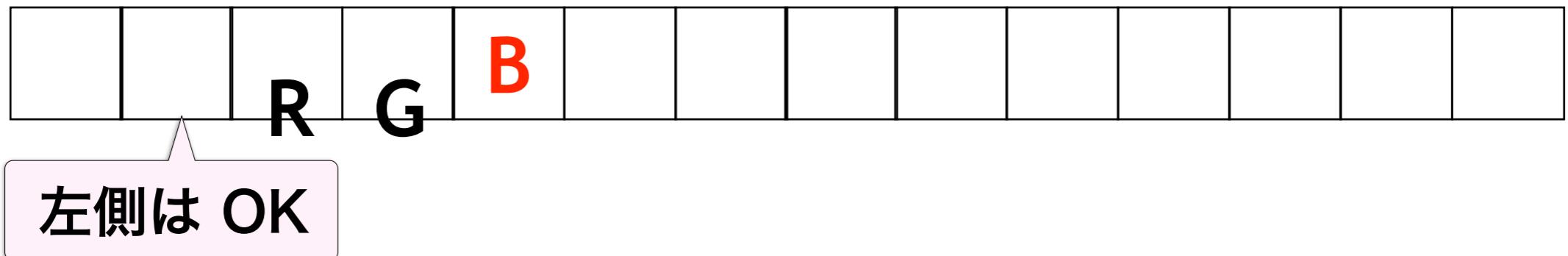


実はすごく簡単な特徴づけ

- 以下のすべてを満たせば OK、満たさなかったら NG
 - R で始まり、B で終わる
 - “RB” と “GG” を含まない

証明

- 最も左の B を考えると、その左は G で、その左は R
- そこで RGB が作られていて、その左は R と G しかなく、G は連続しないので R と RG で区切れる



実はすごく簡単な特徴づけ

- 以下のすべてを満たせば OK、満たさなかったら NG
 - R で始まり、B で終わる
 - “RB” と “GG” を含まない

証明

- 次の B を考えると、前の B から見て「BB」か「BGB」か「B…RGB」になっている
- B と B との間の部分は先頭を除き、R と RG のみで区切れる



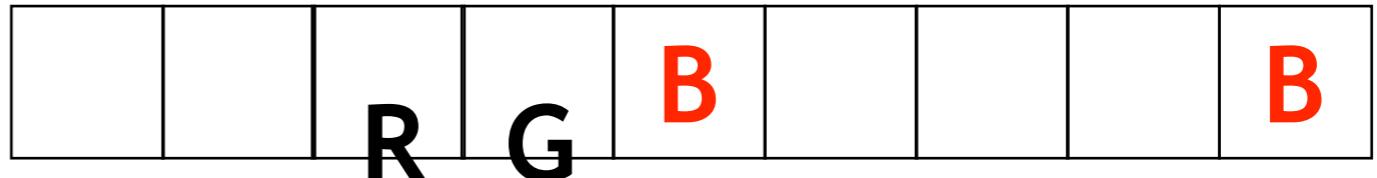
今度はここが G になる可能性がある…が、問題ない

実はすごく簡単な特徴づけ

- 以下のすべてを満たせば OK、満たさなかったら NG
 - R で始まり、B で終わる
 - “RB” と “GG” を含まない

証明

- 以下同様
- 最右端が B なのでそこで終わる



データ

- ジャッジ解
 - drken (C++): 23 行 (解法 2)
 - tempura (Python): 13 行 (解法 2)
 - gojira (C++): 33 行 (解法 1)
- First AC (全体): square1001 (00:08:33)
- First AC (オンライン): cpsco_g_gekihie (00:15:18)
- Accepted: 132
- Submitted: 160

CPSCO 2019 session3

E - Enumerate Xor Sum

原案: drken

解答: drken, tempura, gojira

問題概要

- ・長さ N の数列 A_1, A_2, \dots, A_N が与えられる
- ・各 $k = 1, 2, \dots, N$ に対して、
 - ・ $X = A_1 \text{ xor } \dots \text{ xor } A_k$ として
 - ・ $(A_1 \text{ xor } X) + \dots + (A_k \text{ xor } X)$ を計算せよ

解き方

- ・愚直にやつては $O(N^2)$ となるので間に合わない
- ・XOR の性質として「各桁ごとに独立に考える」という解法が成立することが多い

$$(A_1 \text{ xor } X) + \cdots + (A_k \text{ xor } X)$$

各桁ごとにこの値を計算して、最後にまとめる

- ・ 2^d の位の値が、 k だったら、 $k \times 2^d$ を合算する

0と1の問題に帰着

- ・数列 A が 0 と 1 のみからなるケースに帰着された
- ・各 $k = 1, 2, \dots, N$ に対して、 A_1, A_2, \dots, A_k の中に 1 が奇数個だったら $X = 1$ 、偶数個だったら $X = 0$

$$(A_1 \text{ xor } X) + \dots + (A_k \text{ xor } X)$$



各桁ごとにこの値を計算して、最後にまとめる

- ・ 2^d の位の値が、 k だったら、 $k \times 2^d$ を合算する

0と1の問題に帰着

- ・数列 A が 0 と 1 のみからなるケースに帰着された
- ・各 $k = 1, 2, \dots, N$ に対して、 A_1, A_2, \dots, A_k の中に 1 が奇数個だったら $X = 1$ 、偶数個だったら $X = 0$
- ・ $(A_1 \text{ xor } X) + \dots + (A_k \text{ xor } X)$ の値は、
 - ・ $X = 0$ のときは、 A_1, \dots, A_k の中の 1 の個数
 - ・ $X = 1$ のときは、 A_1, \dots, A_k の中の 0 の個数

累積和で簡単に管理できる！

計算量

- ・ 各桁ごとについては「1 の個数」に関して累積和を求めるだけなので $O(N)$
- ・ 全体としては、登場しうる最大の整数を A として、 $O(N \log A)$

データ

- ・ ジャッジ解
 - ・ drken (C++): 20 行
 - ・ tempura (PyPy): 16 行
 - ・ gojira (C++): 57 行
- ・ First AC (全体): square1001 (00:03:15)
- ・ First AC (オンラインサイト): cpsco_c_114514 (00:11:05)
- ・ Accepted: 90
- ・ Submitted: 106

CPSICO 2019 session3

F - Flexible Permutation

原案: drken

解答: drken, tempura

問題概要

- ・ $1, 2, \dots, N$ の順列のうち
- ・ 以下の条件を満たすものを数え上げよ
 - ・ $P_i > i$ となる i がちょうど A 個
 - ・ $P_i < i$ となる i がちょうど B 個
- ・ $N \leq 300$ (部分点: $N \leq 15$)

部分点解法: bit DP

- 順列について、条件を満たすものを数え上げたり、最適な順列を求めたり (TSP など) するような問題で、制約が小さかったら大体 bitDP で解ける

$dp[\text{(使った数字の集合)}][a][b]$

= (使った数字の集合) の分だけ左から順に並べる方法のうち、 $P_i > i$ となるのが a 箇所、 $P_i < i$ となるのが b 箇所あるようなものの個数

部分点解法: bit DP

$dp[(\text{使った数字の集合})][a][b]$
= (使った数字の集合) の分だけ左から順に並べる方法
のうち、 $P_i > i$ となるのが a 箇所、 $P_i < i$ となるのが
 b 箇所あるようなものの個数

```
// bit で表される数の集合を左から順に並べる
for (int bit = 0; bit < (1<<N); ++bit) {
    int con = __builtin_popcount(bit); // 何個並べてあるか
    for (int a = 0; a <= A; ++a) {
        for (int b = 0; b <= B; ++b) {
            // n: 次に使う数
            for (int n = 0; n < N; ++n) {
                if (bit & (1<<n)) continue; // n が既に使われている
                int nbit = bit | (1<<n);

                // i を con 番目に並べる
                if (n > con) add(dp[nbit][a+1][b], dp[bit][a][b]);
                else if (n < con) add(dp[nbit][a][b+1], dp[bit][a][b]);
                else add(dp[nbit][a][b], dp[bit][a][b]);
            }
        }
    }
}
```

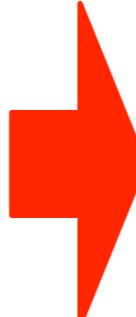
満点解法:挿入(?)DP

- 「 $1, \dots, N+1$ の順列は、 $1, \dots, N$ の順列に対して $N+1$ をどこかに挿入してつくられる」ということを利用した DP

$dp[N][\text{(条件)}]$

= $1, \dots, N$ の順列のうち、(条件)を満たすものの個数

3, 4, 1, 5, 2



6, 3, 4, 1, 5, 2

3, 6, 4, 1, 5, 2

3, 4, 6, 1, 5, 2

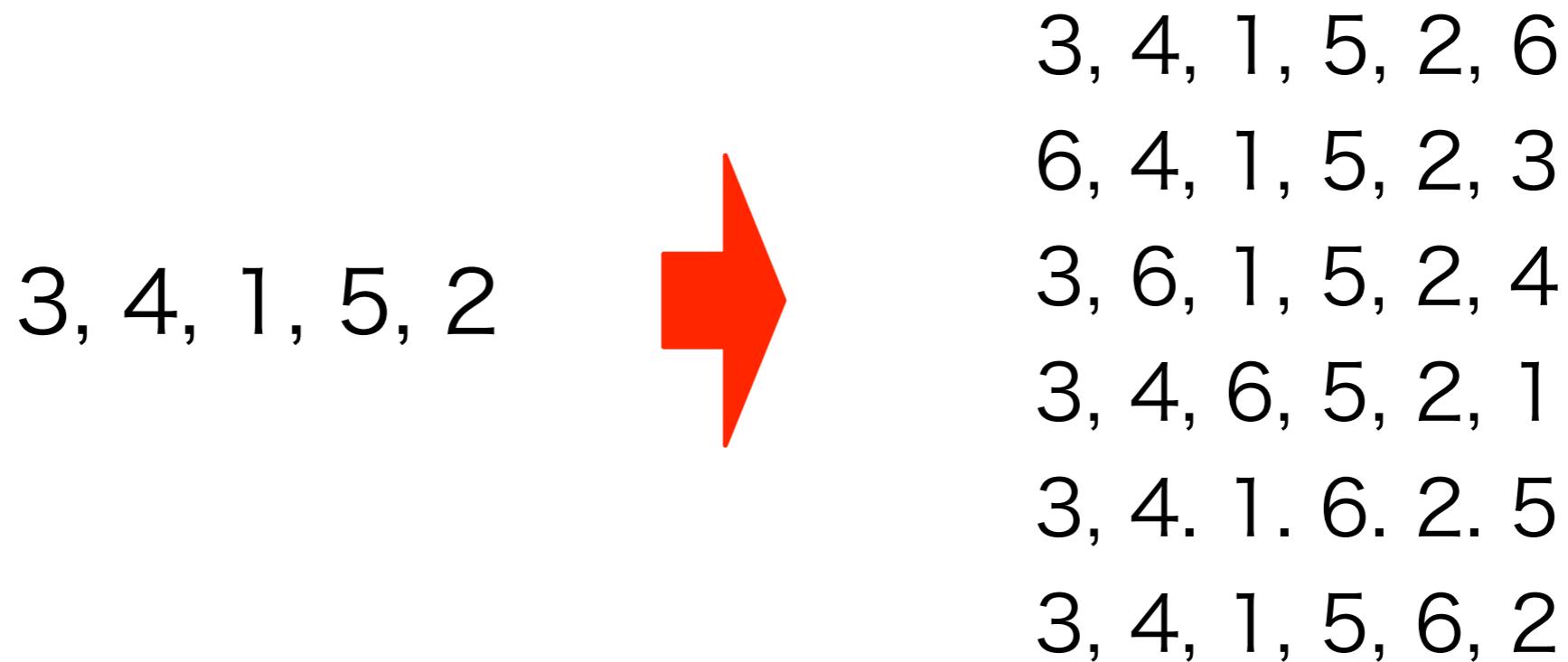
3, 4, 1, 6, 5, 2

3, 4, 1, 5, 6, 2

3, 4, 1, 5, 2, 6

今回は挿入でなく swap

- 1, …, N, N+1 の順列は、1, …, N の順列の右に暫定的に N+1 を置いて、以下のようにして作れる
 - そのまま
 - 1~N のいずれかと swap



swap する場所を考える DP

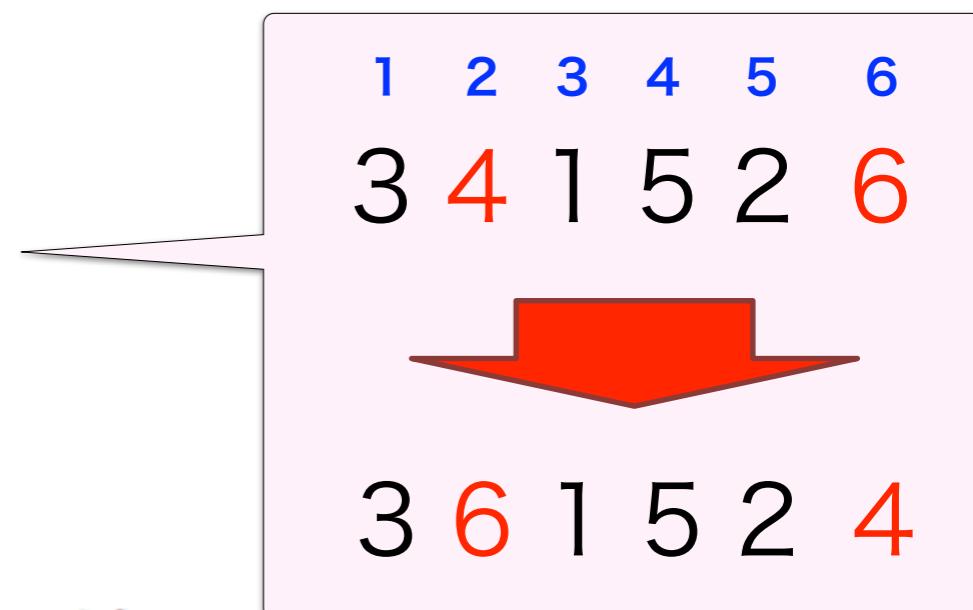
- $dp[n][a][b] := 1 \sim N$ の順列のうち、 $P_i > i$ となる i が a 箇所で、 $P_i < i$ となる i が b 箇所なものの個数

```
// i+1 を i+1 番目に
dp[n+1][a][b] += dp[n][a][b];

// i+1 を a 族と swap
dp[n+1][a][b+1] += dp[n][a][b] * a;

// i+1 を b 族と swap
dp[n+1][a+1][b] += dp[n][a][b] * b;

// i+1 を c 族と swap
dp[n+1][a+1][b+1] += dp[n][a][b] * (n-a-b);
```



- 計算量は $O(N^3)$

さらに $O(N^2)$

- ・ 前処理として、 $P_i = i$ となる場所を選ぶ $_N C_{N-A-B}$ 通りの方法を後で掛けることにして、 $N = A + B$ の場合に帰着させておくことで $O(N^2)$ にするこ
ともできる

データ

- ・ ジャッジ解
 - ・ drken (C++): 43 行
 - ・ tempura (C++): 22 行
 - ・ tempura (C++): 41 行 (箱根駅伝 ver)
- ・ First AC (全体): square1001 (00:23:06)
- ・ First AC (オンラインサイト): - (00:00:00)
- ・ Accepted: 26
- ・ Submitted: 45

CPSCO 2019 session3

G - Grand Election

原案: drken

解答: drken, tempura

問題概要

正の整数 a_1, a_2, \dots, a_N ($A = a_1 + \dots + a_N$) が与えられて、

$$\left| \frac{x_1}{a_1} - \frac{X}{A} \right| + \dots + \left| \frac{x_N}{a_N} - \frac{X}{A} \right|$$

が最小となる整数 x_1, x_2, \dots, x_N ($x_1 + \dots + x_N = K$) を求めよ

想定誤解法

- ・ 概ね、 x と a が比例するようにしたい
- ・ 各 i に対して $\frac{x_i}{a_i} - \frac{X}{A} \leq 0$ となる範囲で最大限 x_i を増やしてあげて、その後は x_i のうち影響が少ないところを少しづつ増やせばよさそう？

$$\left| \frac{x_1}{a_1} - \frac{X}{A} \right| + \cdots + \left| \frac{x_N}{a_N} - \frac{X}{A} \right| \quad (x_1 + \cdots + x_N = K)$$

コーナーケース 1

- ・ある i に対しては x_i を中途半端な値で止めた方がいい
- ・ある i に対しては x_i を一人だけ大きく上げた方がいい
ようなケースがありうる

$$a = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 90), X = 151$$

嘘解法だと、 $x = (1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 135)$ 4.99

正解法だと、 $x = (2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 131)$ 4.95444…

コーナーケース 2

- ・ある i に対しては x_i を中途半端な値で止めた方がいい
- ・ある i に対しては x_i を一人だけ大きく上げた方がいい
ようなケースがありうる

$$a = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 90), X = 149$$

嘘解法だと、 $x = (1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 135)$ 4.99

正解法だと、 $x = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 139)$ 4.95444…

$$\left| \frac{x_1}{2} - \frac{13}{10} \right| + \left| \frac{x_2}{3} - \frac{13}{10} \right| + \left| \frac{x_3}{5} - \frac{13}{10} \right|$$

K 回操作する問題

- 今回の問題を、 x_1, x_2, \dots, x_N のうちどれか 1 つ選んで +1 する操作を合計 K 回行う問題と考える。ここで +1 することで $f(x)$ (最小化したい関数) がどれくらい減るか考える
- 例えば $a = (2, 3, 5)$, $X = 13$ のときは、例えば x_1 について $0 \rightarrow 1$ で $f(x)$ は $1/2$ 減少、 $1 \rightarrow 2$ で $1/2$ 減少、…

$$x_1 : \frac{1}{2}, \frac{1}{2}, \frac{1}{10}, -\frac{1}{2}, -\frac{1}{2}, \dots$$

$$x_2 : \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{4}{15}, -\frac{1}{3}, -\frac{1}{3}, \dots$$

$$x_3 : \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0, -\frac{1}{5}, -\frac{1}{5}, \dots$$

これらから左詰めに 13 個
選んで総和を最大に

$$\left| \frac{x_1}{2} - \frac{13}{10} \right| + \left| \frac{x_2}{3} - \frac{13}{10} \right| + \left| \frac{x_3}{5} - \frac{13}{10} \right|$$

重要な性質

- x_i を 1 増やしたときの $f(x)$ の減少分は、
 i について **単調非増加** である
- 従って、 $f(x_i = t+1) - f(x_i = t)$ を各 (i, t) について計算した
値をまるごとまとめて、大きい順に K 個とればよい

$$x_1 : \frac{1}{2}, \frac{1}{2}, \frac{1}{10}, -\frac{1}{2}, -\frac{1}{2}, \dots$$

$$x_2 : \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{4}{15}, -\frac{1}{3}, -\frac{1}{3}, \dots$$

$$x_3 : \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0, -\frac{1}{5}, -\frac{1}{5}, \dots$$

これらを全部まとめて
大きい順に K 個とする

$$\left| \frac{x_1}{2} - \frac{13}{10} \right| + \left| \frac{x_2}{3} - \frac{13}{10} \right| + \left| \frac{x_3}{5} - \frac{13}{10} \right|$$

高速化

- この時点で $O(K \log K)$
- 各 x_i について、 x_i を 1 増やしたときの $f(x)$ の減少量は、3 パターンしかないので、同じ値が続く箇所ではまとめて更新することができる。これで $O(N \log N)$ になる。

$$x_1 : \frac{1}{2}, \frac{1}{2}, \frac{1}{10}, -\frac{1}{2}, -\frac{1}{2}, \dots$$

$$x_2 : \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{4}{15}, -\frac{1}{3}, -\frac{1}{3}, \dots$$

$$x_3 : \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, 0, -\frac{1}{5}, -\frac{1}{5}, \dots$$

これらを全部まとめて
大きい順に K 個

誤差に注意

- $f(x)$ の値が同じ最適値を達成する解が複数ある場合は
辞書順最小のものを出力したい
- 複数の i に対して x_i をインクリメントしたときの $f(x)$ の
減少分が一致したときは、 i が大きいものを優先する
- このとき、減少分が一致するかどうかの判定は、有理数を
用いる無難。実数でも EPS をきちんと設定すること
で通すことができる

データ

- ジャッジ解
 - drken (C++): 71 行 (long double 型)
 - drken (C++): 162 行 (有理数型)
 - tempura (C++): 78 行 (工夫した有理数型)
- First AC (全体): luma (01:14:54)
- First AC (オンラインサイト): - (00:00:00)
- Accepted: 5
- Submitted: 21