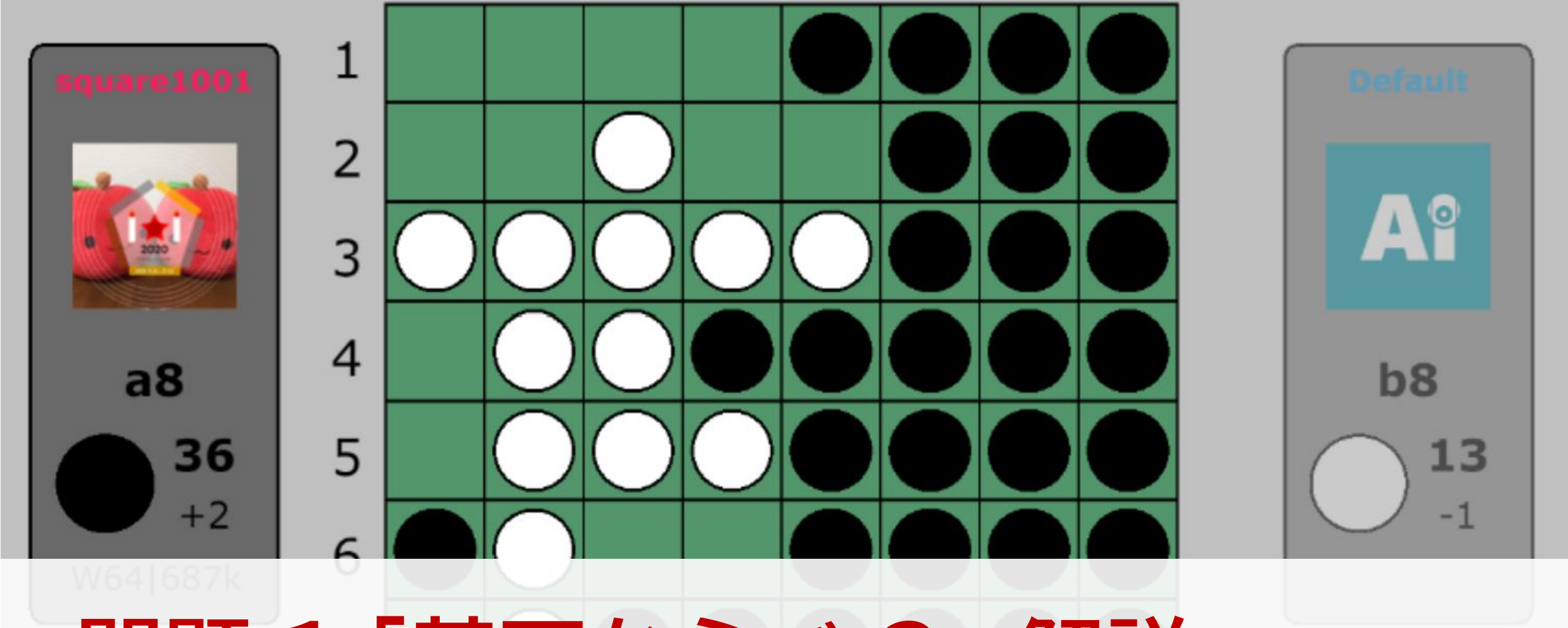


問題 1「碁石ならべ 2」解説

Problème 1 “Disposer des Pierres 2”

解説: 米田 寛峻（よねだ ひろたか）/ square1001

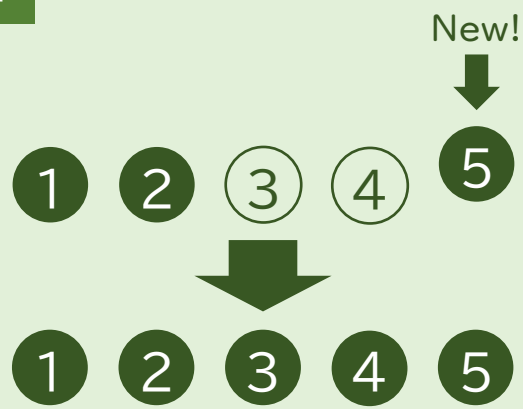
1 問題の説明

N 個の碁石を左から順に並べます

1

i 番目に置く石の色は A_i です

2



同色の 2 つの石に挟まれると同じ色に変化します

3

最終的な各石の色を求めよ

1 制約

$N \leq 2000$
$A_i \leq 10^9$

小課題 1 (25 点)

$N \leq 200\,000$
$A_i = 1, 2$

小課題 2 (60 点)

$N \leq 200\,000$
$A_i \leq 10^9$

小課題 3 (100 点)

Chapter II

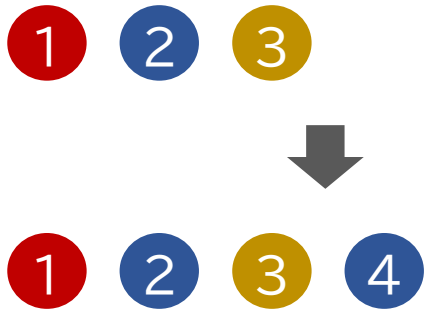
小課題 1 の解法

Sous-tâche 1

2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう



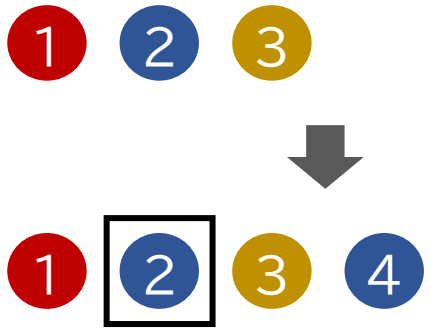
各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう



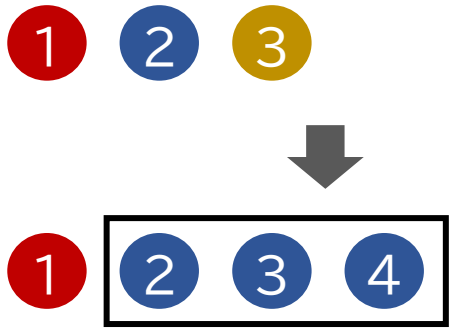
各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう



各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

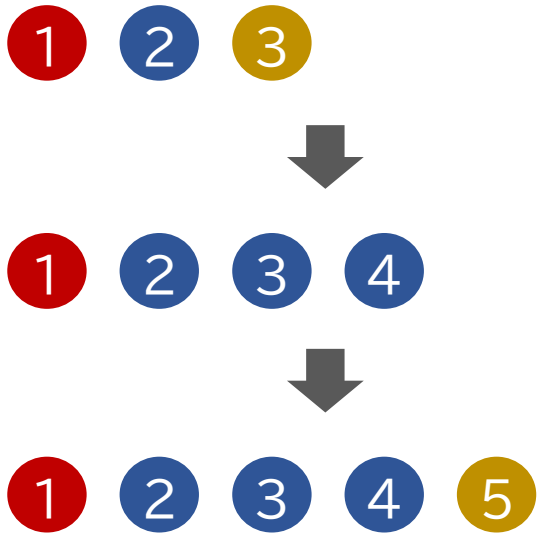
2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える



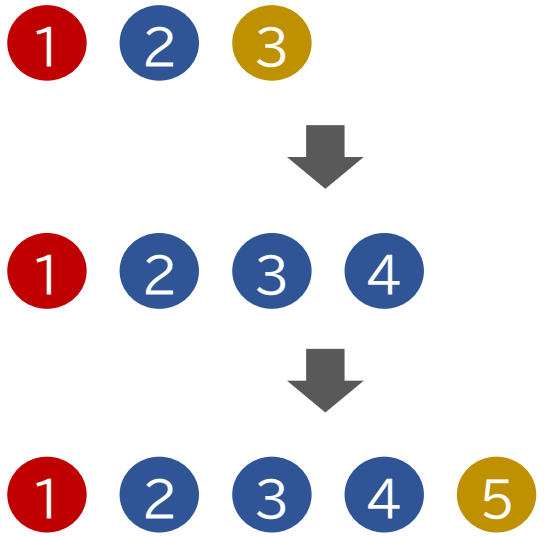
2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える



黄色の石は存在しない!

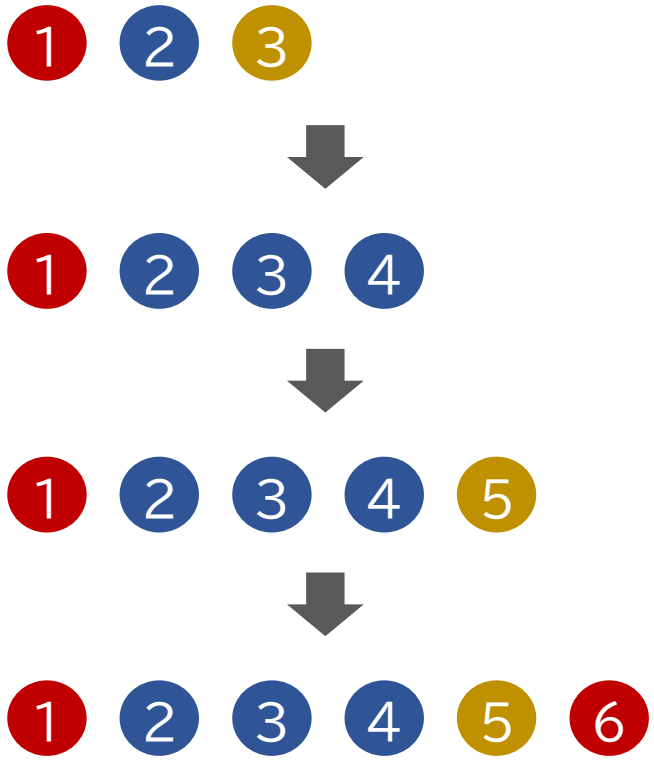
2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える



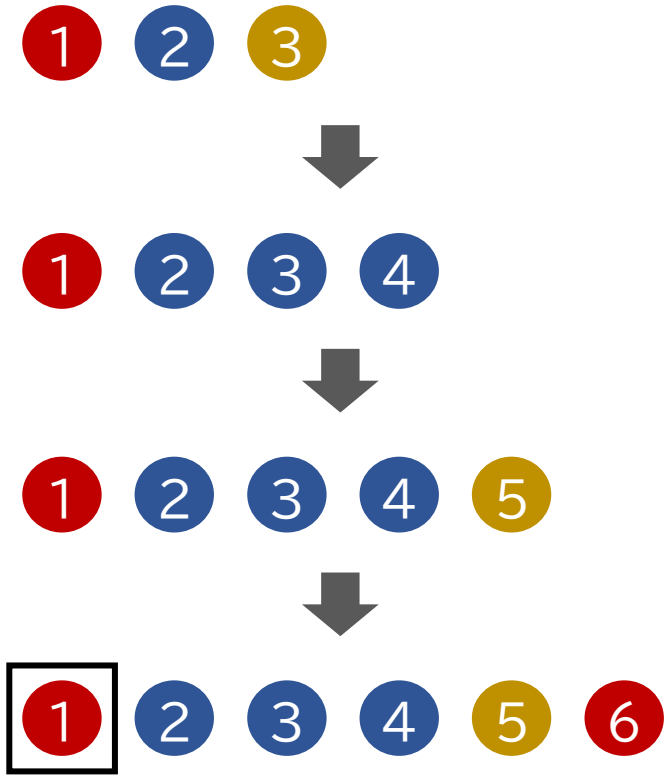
2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える



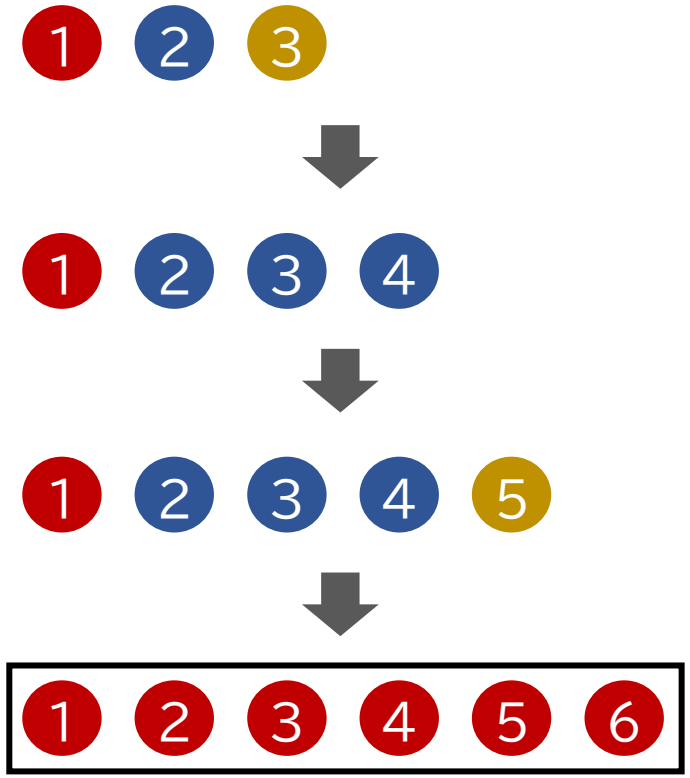
2 小課題 1 の解法

小課題 1 の制約は $N \leq 2000$ です

単純にシミュレーションすることを考えましょう

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える



2 小課題 1 の解法

解答例を右に示します

手順ごとに分けて考えると実装しやすいです

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

```
int main() {
    // step #1. read input
    int N;
    cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }

    // step #2. simulation
    vector<int> stones;
    for (int i = 0; i < N; i++) {
        int pos = -1;
        for (int j = 0; j < i; j++) {
            if (stones[j] == A[i]) {
                pos = max(pos, j);
            }
        }
        if (pos != -1) {
            for (int j = pos + 1; j < i; j++) {
                stones[j] = A[i];
            }
        }
        stones.push_back(A[i]);
    }

    // step #3. output
    for (int i = 0; i < N; i++) {
        cout << stones[i] << endl;
    }

    return 0;
}
```

2 小課題 1 の解法

解答例を右に示します

手順ごとに分けて考えると実装しやすいです

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

入力部分

```
int main() {
    // step #1. read input
    int N;
    cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }

    // step #2. simulation
    vector<int> stones;
    for (int i = 0; i < N; i++) {
        int pos = -1;
        for (int j = 0; j < i; j++) {
            if (stones[j] == A[i]) {
                pos = max(pos, j);
            }
        }
        if (pos != -1) {
            for (int j = pos + 1; j < i; j++) {
                stones[j] = A[i];
            }
        }
        stones.push_back(A[i]);
    }

    // step #3. output
    for (int i = 0; i < N; i++) {
        cout << stones[i] << endl;
    }

    return 0;
}
```

2 小課題 1 の解法

解答例を右に示します

手順ごとに分けて考えると実装しやすいです

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

```
int main() {
    // step #1. read input
    int N;
    cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }

    // step #2. simulation
    vector<int> stones;
    for (int i = 0; i < N; i++) {
        int pos = -1;
        for (int j = 0; j < i; j++) {
            if (stones[j] == A[i]) {
                pos = max(pos, j);
            }
        }
        if (pos != -1) {
            for (int j = pos + 1; j < i; j++) {
                stones[j] = A[i];
            }
        }
        stones.push_back(A[i]);
    }

    // step #3. output
    for (int i = 0; i < N; i++) {
        cout << stones[i] << endl;
    }

    return 0;
}
```

手順 1 の部分

2 小課題 1 の解法

解答例を右に示します

手順ごとに分けて考えると実装しやすいです

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

```
int main() {
    // step #1. read input
    int N;
    cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }

    // step #2. simulation
    vector<int> stones;
    for (int i = 0; i < N; i++) {
        int pos = -1;
        for (int j = 0; j < i; j++) {
            if (stones[j] == A[i]) {
                pos = max(pos, j);
            }
        }
        if (pos != -1) {
            for (int j = pos + 1; j < i; j++) {
                stones[j] = A[i];
            }
        }
        stones.push_back(A[i]);
    }

    // step #3. output
    for (int i = 0; i < N; i++) {
        cout << stones[i] << endl;
    }

    return 0;
}
```

手順 2 の部分

2 小課題 1 の解法

解答例を右に示します

手順ごとに分けて考えると実装しやすいです

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

```
int main() {
    // step #1. read input
    int N;
    cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }

    // step #2. simulation
    vector<int> stones;
    for (int i = 0; i < N; i++) {
        int pos = -1;
        for (int j = 0; j < i; j++) {
            if (stones[j] == A[i]) {
                pos = max(pos, j);
            }
        }
        if (pos != -1) {
            for (int j = pos + 1; j < i; j++) {
                stones[j] = A[i];
            }
        }
        stones.push_back(A[i]);
    }
```

出力部分

```
    // step #3. output
    for (int i = 0; i < N; i++) {
        cout << stones[i] << endl;
    }

    return 0;
}
```

2 小課題 1 の解法

解答例を右に示します

手順ごとに分けて考えると実装しやすいです

25 点獲得！

各ステップで行う操作

1. 色 A_i の石の中で最も右のものを求める
2. 1 で求めた場所より右を色 A_i で塗り替える

```
int main() {
    // step #1. read input
    int N;
    cin >> N;
    vector<int> A(N);
    for (int i = 0; i < N; i++) {
        cin >> A[i];
    }

    // step #2. simulation
    vector<int> stones;
    for (int i = 0; i < N; i++) {
        int pos = -1;
        for (int j = 0; j < i; j++) {
            if (stones[j] == A[i]) {
                pos = max(pos, j);
            }
        }
        if (pos != -1) {
            for (int j = pos + 1; j < i; j++) {
                stones[j] = A[i];
            }
        }
        stones.push_back(A[i]);
    }

    // step #3. output
    for (int i = 0; i < N; i++) {
        cout << stones[i] << endl;
    }

    return 0;
}
```

出力部分

Chapter III

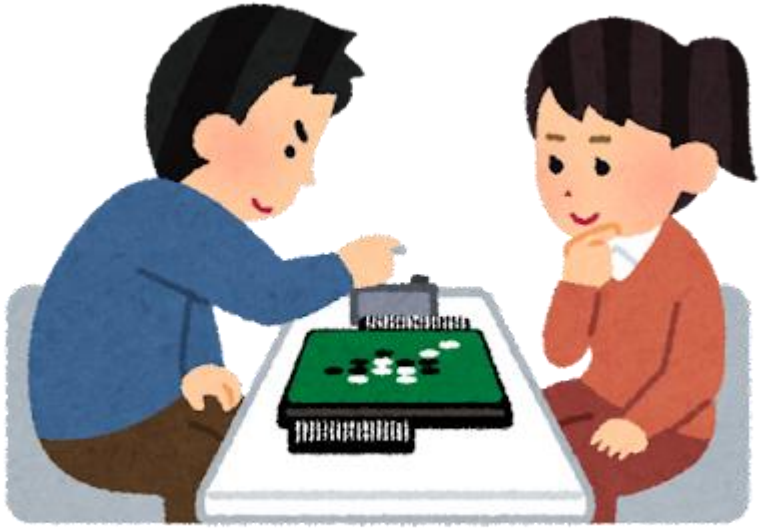
小課題 2 の解法

Sous-tâche 2

3 小課題 2 の解法

小課題 2 の制約は「$A_i = 1, 2$」です

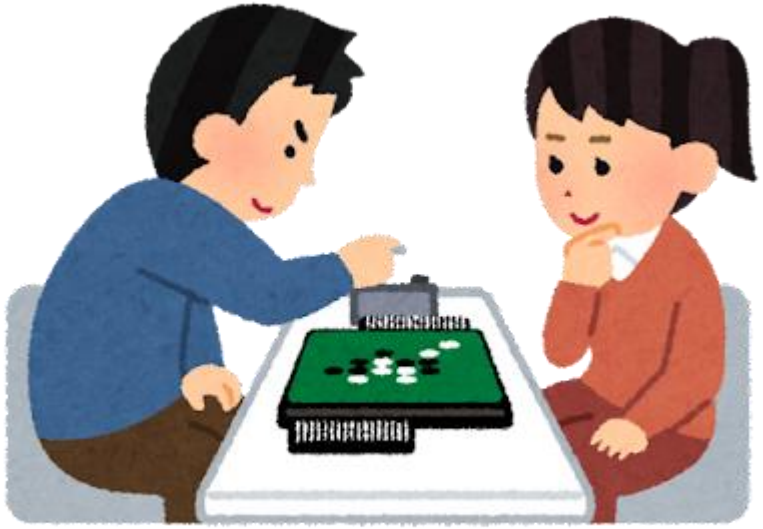
つまり、囲碁やオセロのように「黒と白しかない」ということです



3 小課題 2 の解法

どのような性質があるでしょうか？

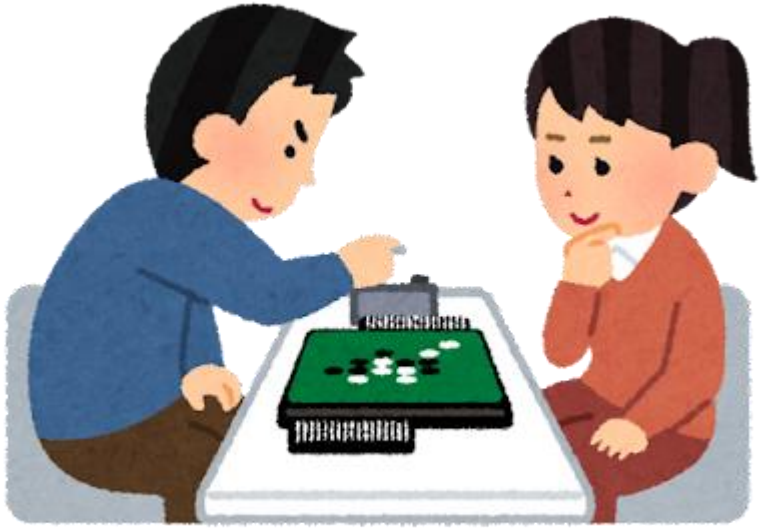
これを探るため、小さいケースで実験してみましょう



3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



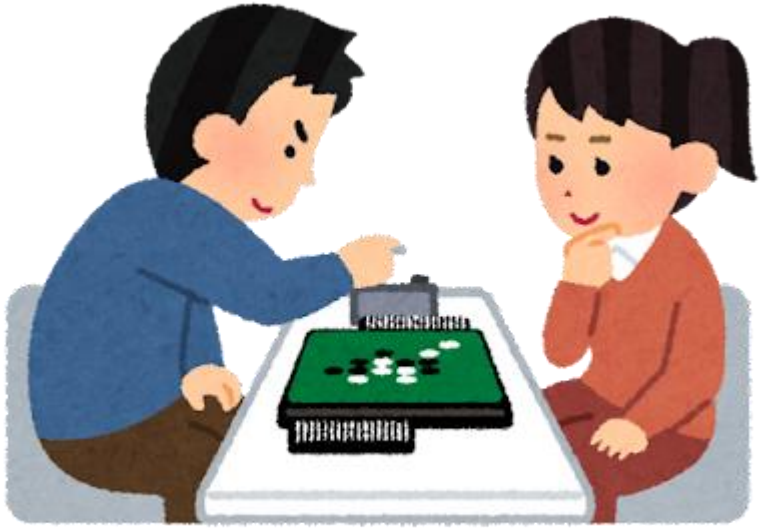
例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

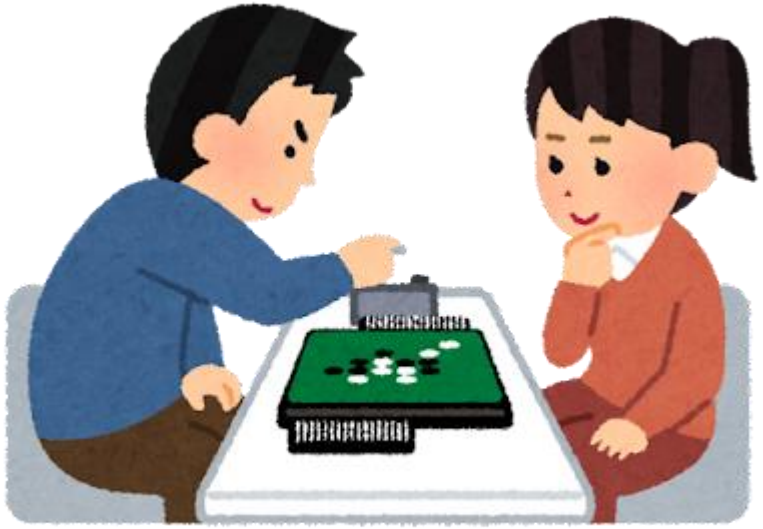
❶

❶ ❷

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

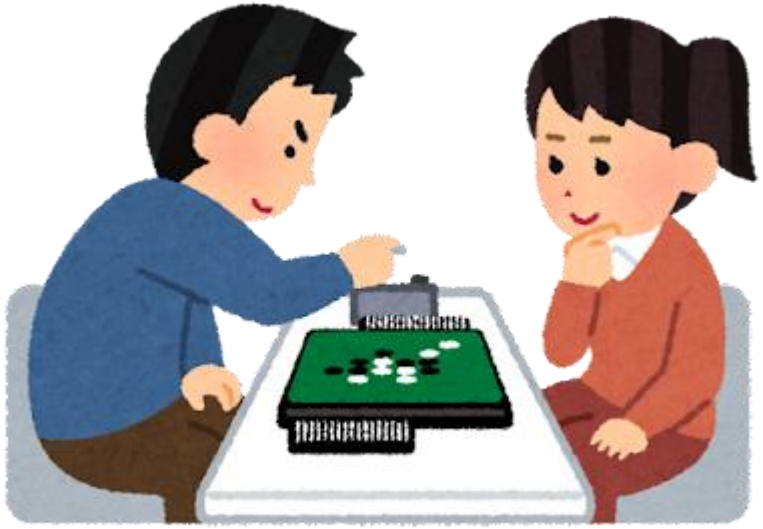
❶ ❷

❶ ❷ ③

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶ ❷

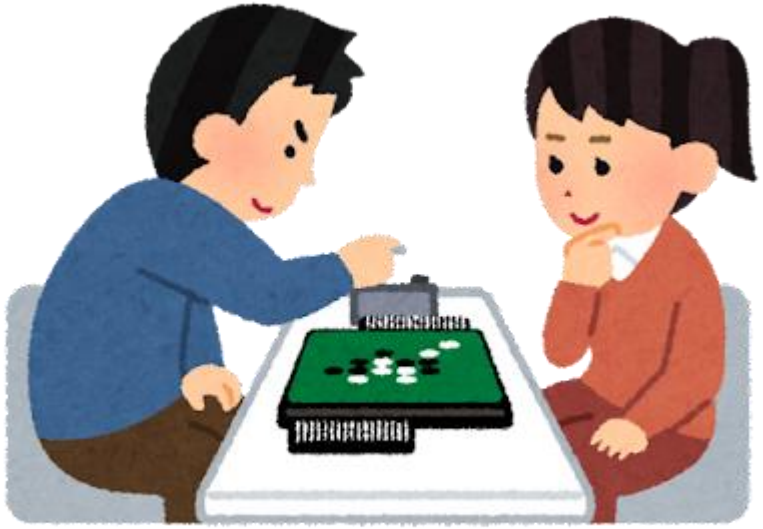
❶ ❷ ③

❶ ❷ ③ ④

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶ ❷

❶ ❷ ③

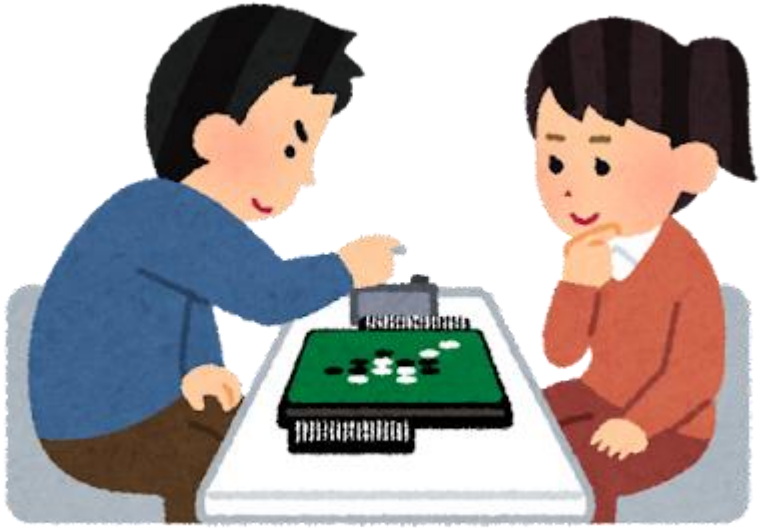
❶ ❷ ③ ④

❶ ❷ ❸ ❹ ❺

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶ ❷

❶ ❷ ③

❶ ❷ ③ ④

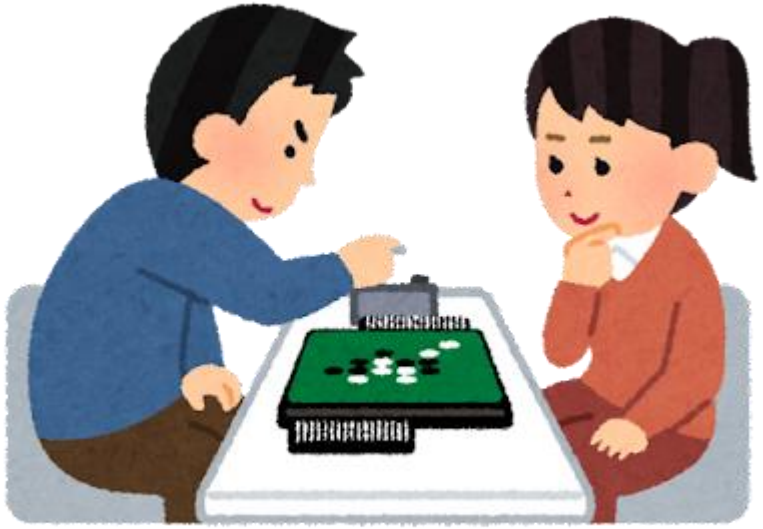
❶ ❷ ❸ ❹ ❺

❶ ❷ ❸ ❹ ❺ ⑥

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶ ❷

❶ ❷ ③

❶ ❷ ③ ④

❶ ❷ ❸ ❹ ❺

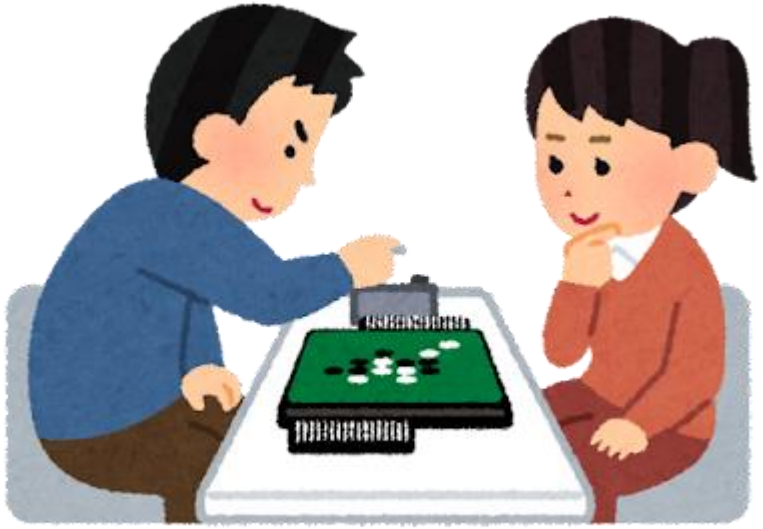
❶ ❷ ❸ ❹ ❺ ⑥

❶ ❷ ❸ ❹ ❺ ❻ ❼

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶ ❷

❶ ❷ ③

❶ ❷ ③ ④

❶ ❷ ❸ ❹ ❺

❶ ❷ ❸ ❹ ❺ ⑥

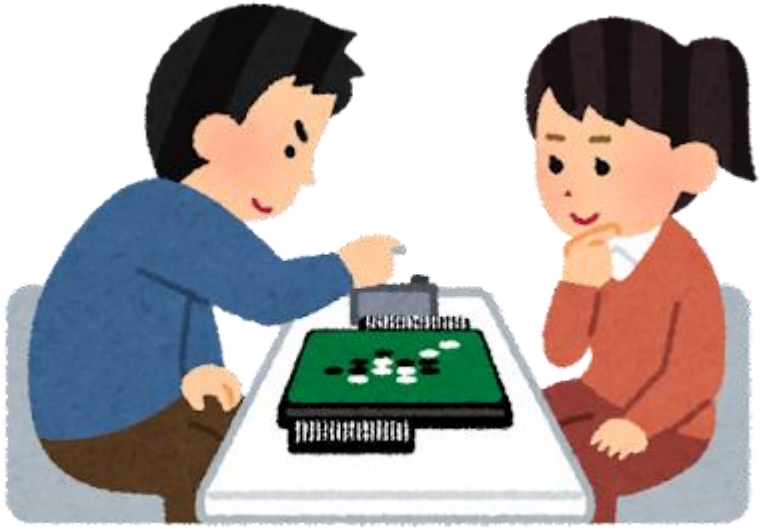
❶ ❷ ❸ ❹ ❺ ❻ ❼

❶ ❷ ❸ ❹ ❺ ❻ ❼ ⑧

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう



例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶ ❷

❶ ❷ ③

❶ ❷ ③ ④

❶ ❷ ❸ ❹ ❺

❶ ❷ ❸ ❹ ❺ ⑥

❶ ❷ ❸ ❹ ❺ ❻ ❼

❶ ❷ ❸ ❹ ❺ ❻ ❼ ⑧

❶ ❷ ❸ ❹ ❺ ❻ ❼ ⑧ ⑨

3 小課題 2 の解法

どのような性質があるでしょうか？

これを探るため、小さいケースで実験してみましょう

結論

どんな場合でも「黒, 黒, …, 黒 の後に白, 白, …, 白」という状態になる！

例：$A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶❷

❶❷③

❶❷③④

❶❷❸❹❺

❶❷❸❹❺⑥

❶❷❸❹❺❻❼

❶❷❸❹❺❻❼⑧

❶❷❸❹❺❻❼⑧⑨

3 小課題 2 の解法

結論

どんな場合でも「黒, 黒, …, 黒 の後に
白, 白, …, 白」という状態になる!

どうしてそんなことになるのか?

k 個石を置いたとき「黒, …, 黒, 白, …, 白」となっていると仮定する

1. もし白が置かれると「黒, …, 黒, 白, …, 白, 白」となる
2. もし黒が置かれると「黒, …, 黒, 黒, …, 黒, 黒」となる

→ $k+1$ 個でも「黒, …, 黒, 白, …, 白」に!

例: $A = (1, 1, 2, 2, 1, 2, 1, 2, 2)$ の場合

❶

❶❷

❶❷③

❶❷③④

❶❷❸❹❺

❶❷❸❹❺⑥

❶❷❸❹❺❻❼

❶❷❸❹❺❻❼⑧

❶❷❸❹❺❻❼⑧⑨

3 小課題 2 の解法

解法

一番左が黒であれば、一番右の黒まで全部黒になる（それより右は全部白）

入力：　❶❷③④❺⑥❼⑧⑨

↓

出力：　❶❷❸❹❺❻❼⑧⑨

3 小課題 2 の解法

解法

一番左が黒であれば、一番右の黒まで全部黒になる（それより右は全部白）

入力：　❶❷③④❺⑥❼⑧⑨

↓

出力：　❶❷❸❹❺❻❼⑧⑨

実際にプログラムを書くときは…

> $A_1 = A_i$ となる最大の i まで、
> 全部色 A_1 で塗られる

ということになります

当然、計算量は $O(N)$

3　小課題 2 の解法

解法

一番左が黒であれば、一番右の黒まで全部黒に
なる（それより右は全部白）

入力：　❶ ❷ ③ ④ ❺ ⑥ ❼ ⑧ ⑨

⬇

出力：　❶ ❷ ❸ ❹ ❺ ❻ ❼ ⑧ ⑨

実際にプログラムを書くときは…

$A_1 \geq 4$ となる最大の i まで、
全部色 A_i で塗られる

ということになります

当然、計算量は $O(N)$

60 点獲得！

Chapter IV

小課題 3（満点）の解法

Sous-tâche 3

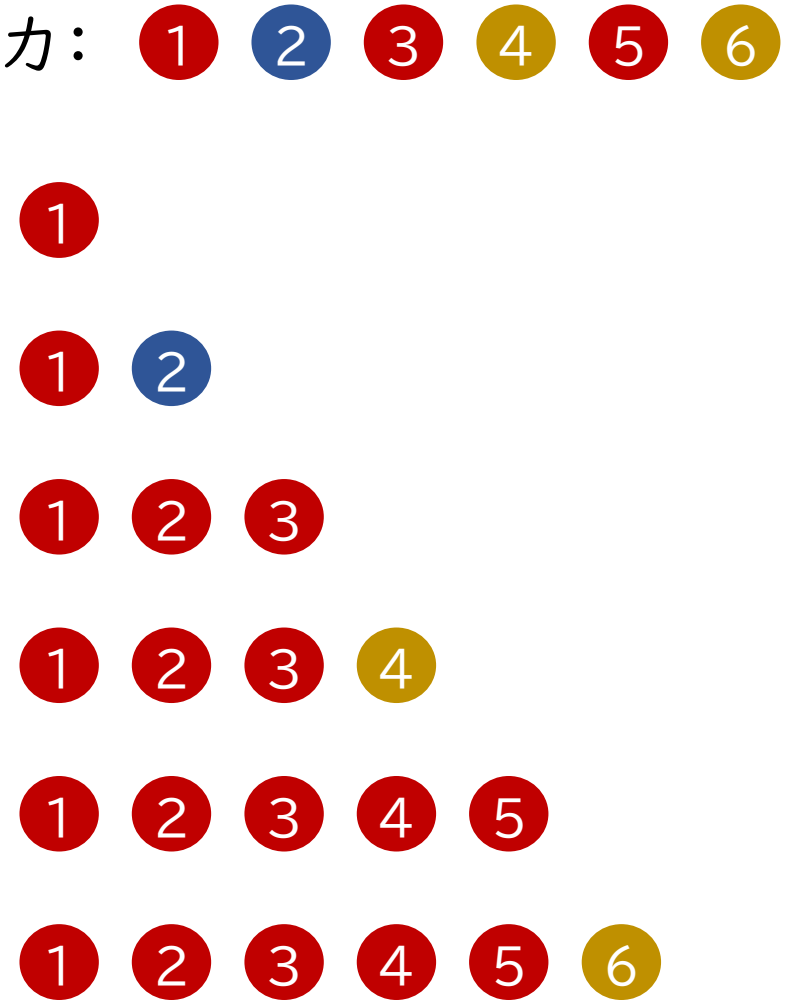
4 満点解法

実は、一般の場合でも、小課題 2 と同じような性質がある

性質

最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

入力：



4 満点解法

性質 最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

この性質を使って、答えを求めよう！

A B C D E F G H I J K L M N O P Q R S T

4 満点解法

性質 最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

この性質を使って、答えを求めよう！

A B C D $\boxed{\text{E}}$ F G H I J K L M N O P Q R S T

最も右の赤色はここなので…

4 満点解法

性質 最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

この性質を使って、答えを求めよう！

これより左が全部赤とわかる！

4 満点解法

性質 最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

この性質を使って、答えを求めよう！

これより左が全部赤とわかる！

Question. 残りの部分はどうやって求める？

4 満点解法

性質 最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

一旦 5 番目までの石の存在を忘れることにしよう（後の結果には関係ない）

A B C D E F G H I J K L M N O P Q R S T

4 満点解法

性質 最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

一旦 5 番目までの石の存在を忘れることにしよう（後の結果には関係ない）

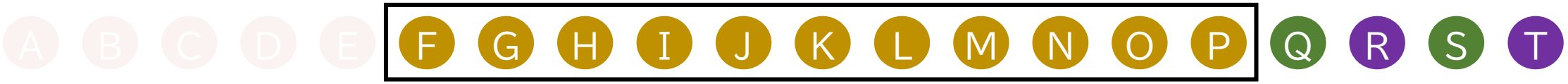
A B C D E F G H I J K L M N O P Q R S T

最も右の黄色はここなので…

4 満点解法

性質 最も右の色 A_1 の石までが、全部色 A_1 に塗り替わる

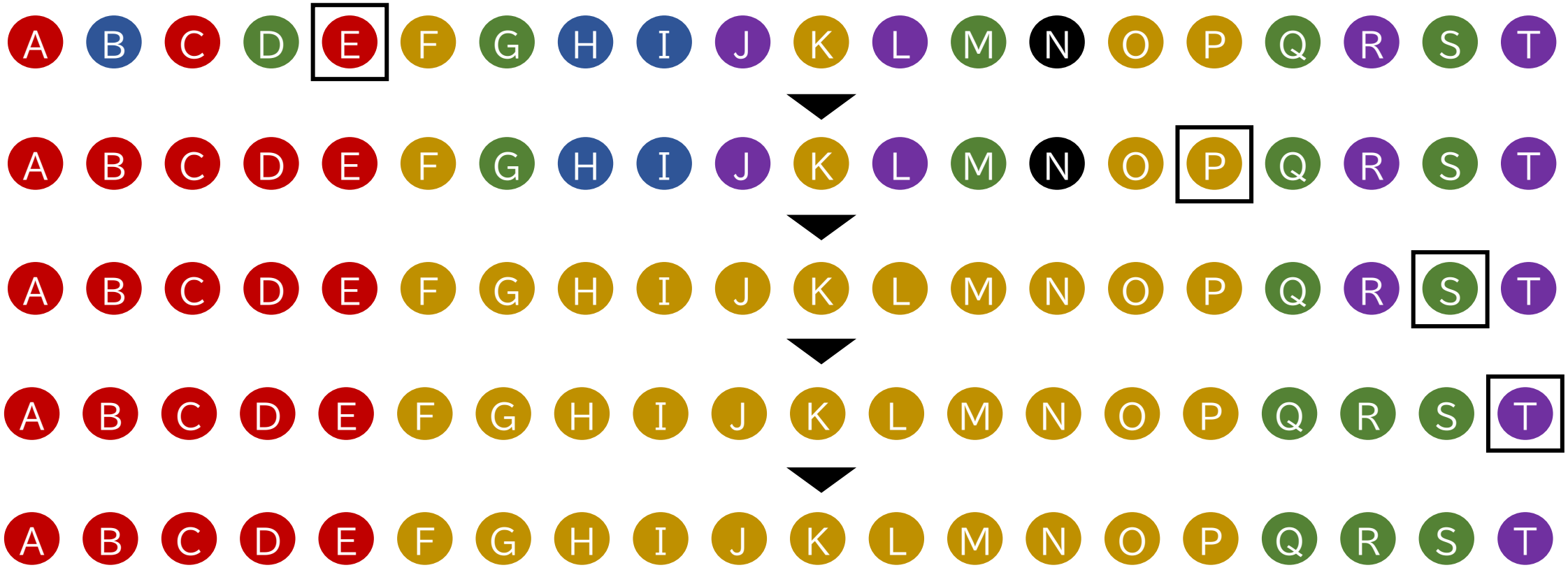
一旦 5 番目までの石の存在を忘れることにしよう（後の結果には関係ない）



これより左が全部黄色とわかる！

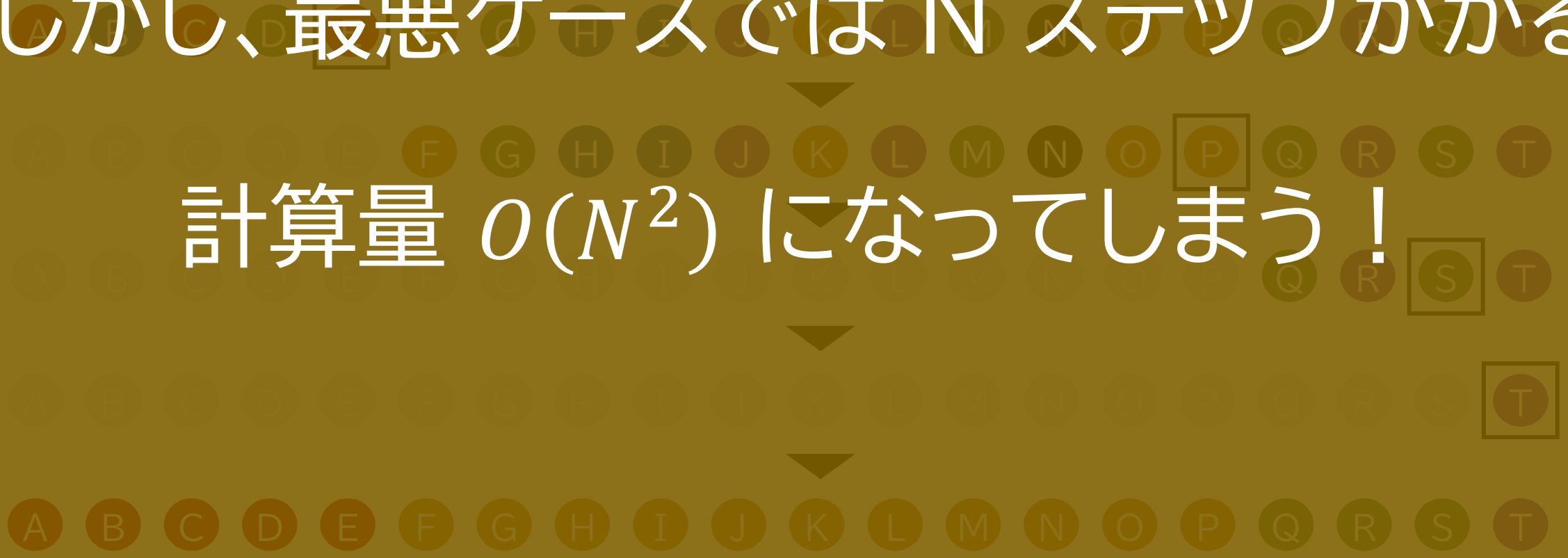
4 満点解法

残りも同様にやっていくと、次のようになります



4 満点解法

残りも同様にやっていくと、次のようになります



しかし、最悪ケースでは N ステップかかる

計算量 $O(N^2)$ になってしまう！

4 満点解法

Question. 各ステップで「どこまで塗り替わるか」は、どうやって高速計算できるか？

4 満点解法

Question. 各ステップで「どこまで塗り替わるか」は、どうやって高速計算できるか？

アイデア 各色に対して、最も右のものを前計算する

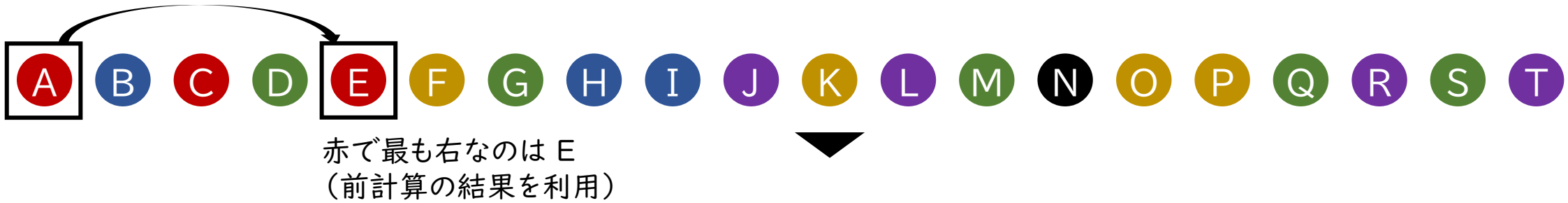
A B C D E F G H I J K L M N O P Q R S T

（この例だと、赤: E、青: I、黄: P、緑: S、紫: T、黒: N）

C++ では map を使って実装すると便利

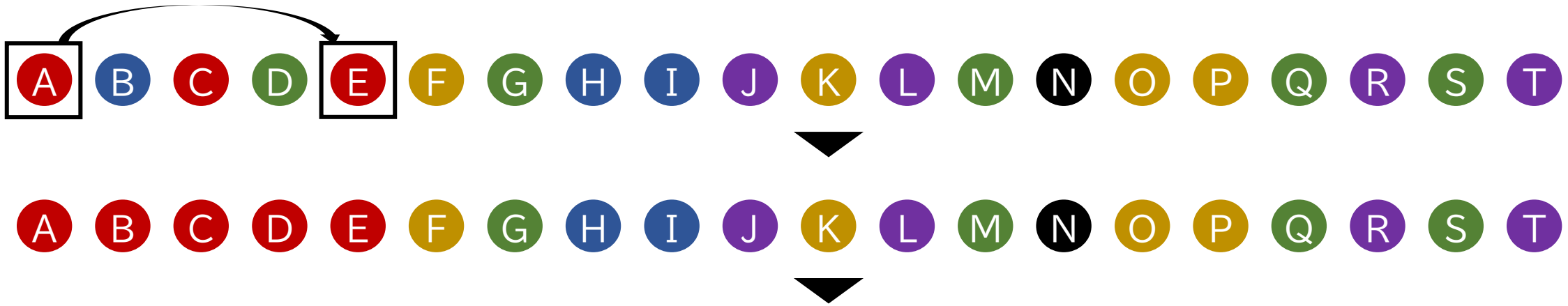
4 満点解法

すると…？（最も右は 赤: E、青: I、黄: P、緑: S、紫: T、黒: N）



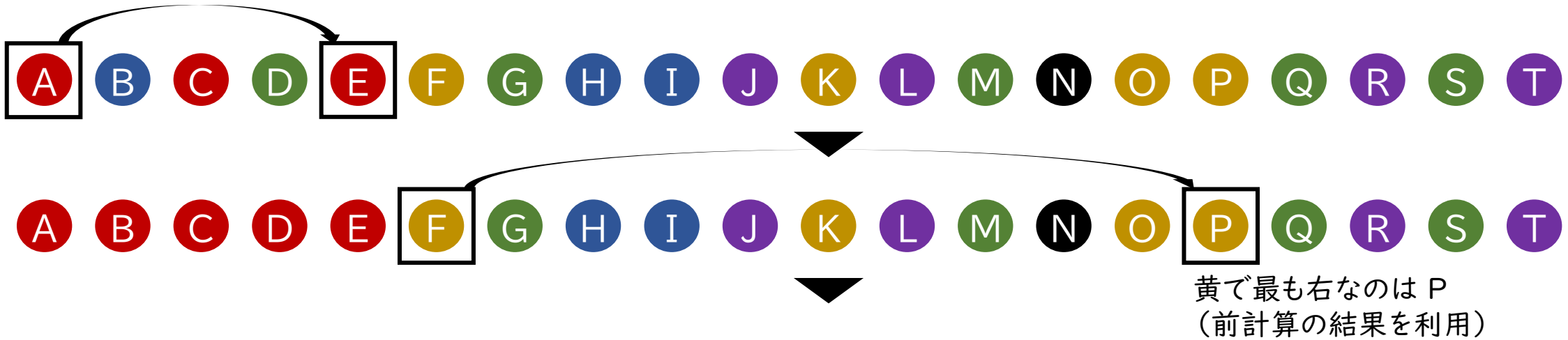
4 満点解法

すると…？（最も右は 赤: E、青: I、黄: P、緑: S、紫: T、黒: N）



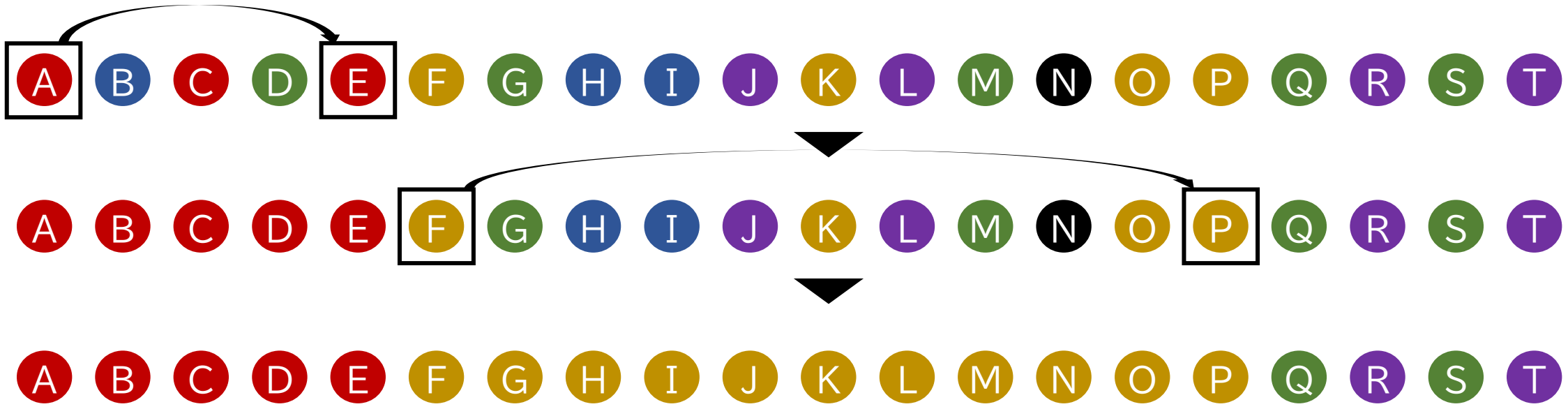
4 満点解法

すると…？（最も右は 赤: E、青: I、黄: P、緑: S、紫: T、黒: N）



4 満点解法

すると…？（最も右は 赤: E、青: I、黄: P、緑: S、紫: T、黒: N）



4 満点解法

すると…？（最も右は 赤: E、青: I、黄: P、緑: S、紫: T、黒: N）

前計算の結果を利用すれば

１ステップをすぐに終えられる！

4 満点解法

計算量について

- map を使うとアクセスに $O(\log n)$ かかります
- そのため、全体計算量 $O(N \log N)$ で解けます

別解について

- ハッシュマップ (unordered_map) を使うと計算量 $O(N)$ になります
- 圧縮を使って同じ計算量で解く解法もあります

4 満点解法

計算量について

- map を使うとアクセスに $O(\log n)$ かかります
- そのため、全体計算量 $O(N \log N)$ で解けます

100 点獲得！

別解について

- ハッシュマップ (unordered_map) を使うと計算量 $O(N)$ になります
- 圧縮を使って同じ計算量で解く解法もあります

Chapter V

得点分布

Répartition des scores

5 得点分布

得点分布

- 100 点：135 人
- 60 点（小課題 1+2）：25 人
- 35 点（小課題 2）：3 人
- 25 点（小課題 2）：6 人

5 宣伝

CodinGame というサイトでオセロ AI を作って遊べます

- 打倒 square1001 ！