第９週
ソート（１）
挿入ソート、選択ソート、バブルソート

必須問題９－１：選択ソート（データ構造とアルゴリズムを参照）

●アルゴリズム

- 要素の中から最小値、2番目に小さい値、…と残っている要素の中での最小値選択を反復して整列する手法

選択ソートのポイント

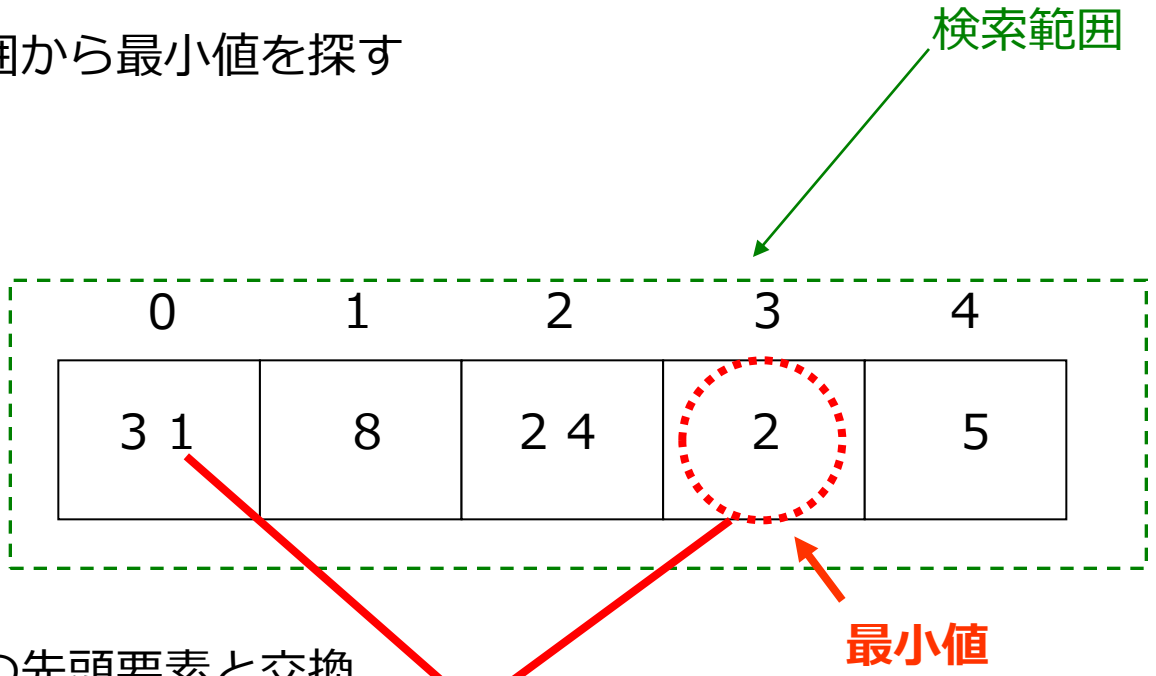
●最小値を探す（step1）：その時点で整列が済んでいない配列要素の中から最小値を探す。

●入れ替えを行なう(step2)：最小値と、値を確定させる配列要素とを入れ替える。

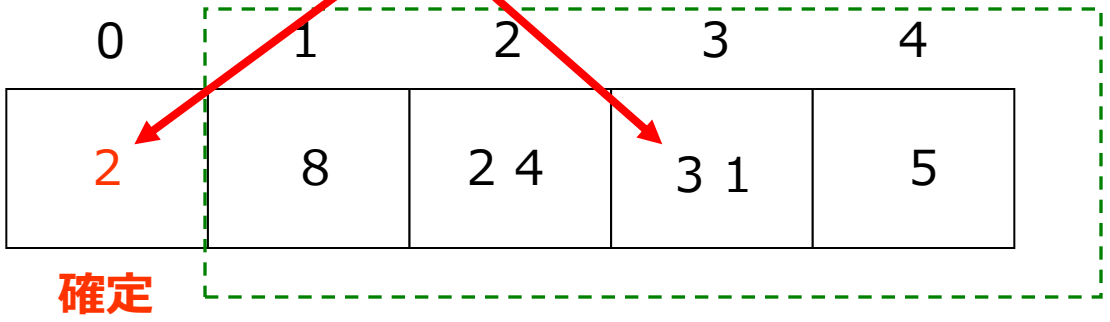
●整列処理の終了：上記のstep1、step２を繰り返し、最後の要素の一つ手前が確定したら、整列処理を終了する。

必須問題９－１：選択ソートの例（昇順）１／４

ＳＴＥＰ１：対象範囲から最小値を探す

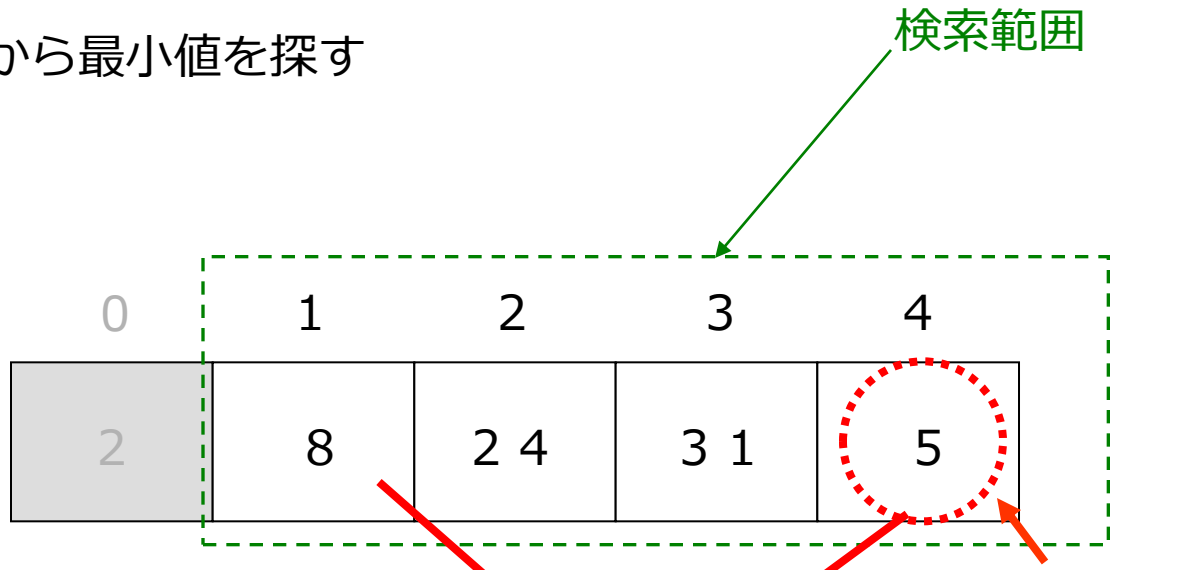


ＳＴＥＰ２：未整列の先頭要素と交換

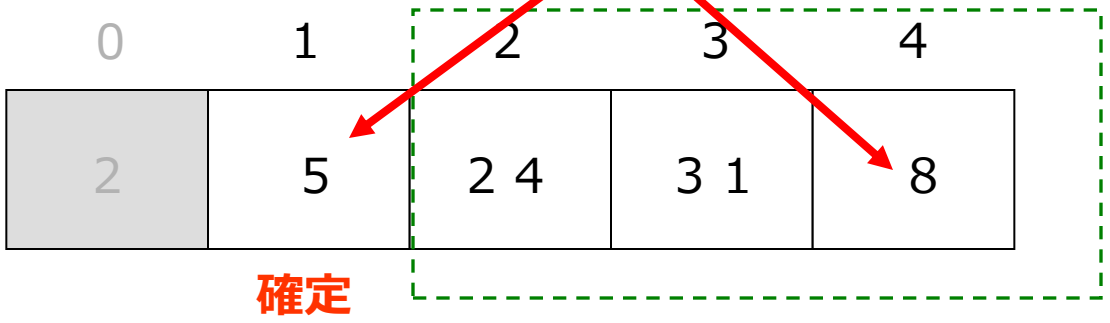


必須問題９－１：選択ソートの例（昇順）２／４

ＳＴＥＰ１：対象範囲から最小値を探す

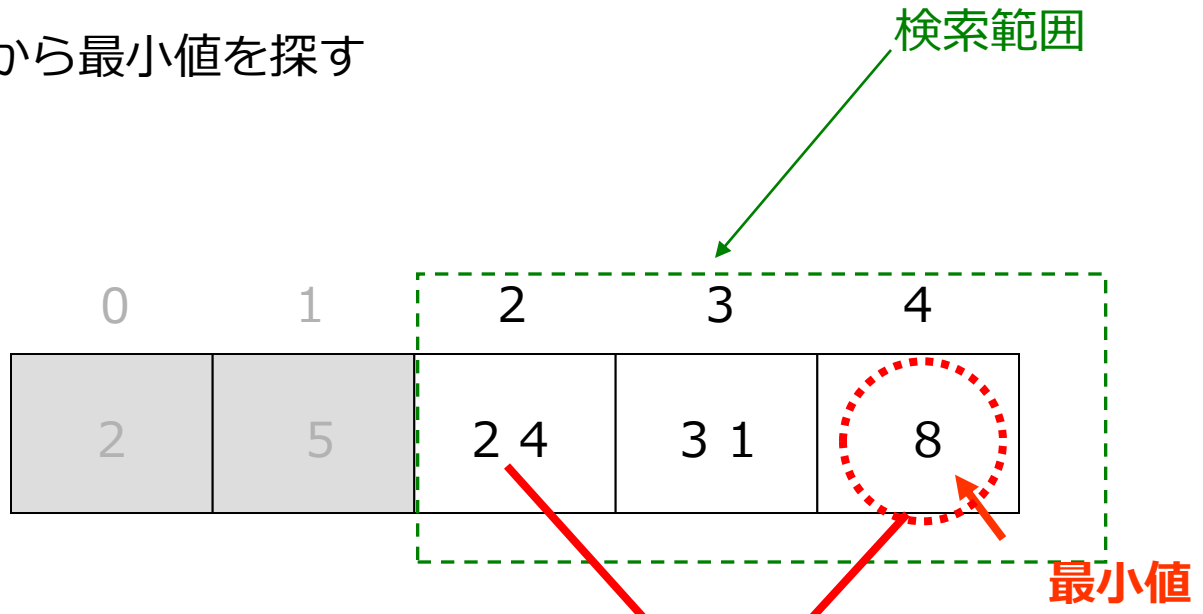


ＳＴＥＰ２：未整列の先頭要素と交換

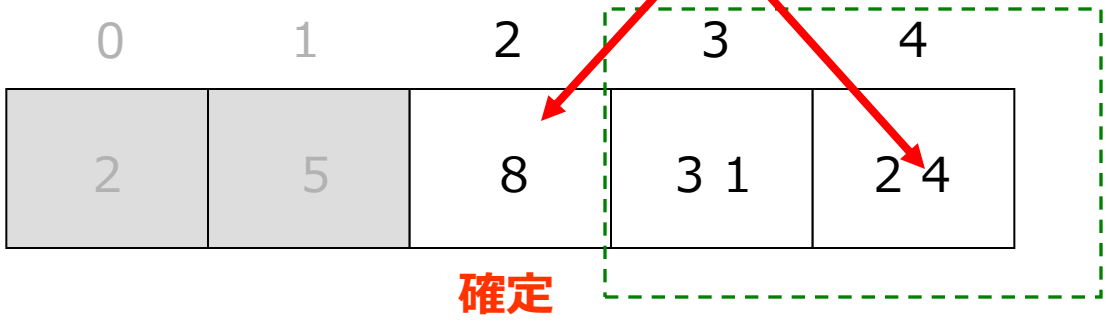


必須問題９－１：選択ソートの例（昇順） ３／４

ＳＴＥＰ１：対象範囲から最小値を探す

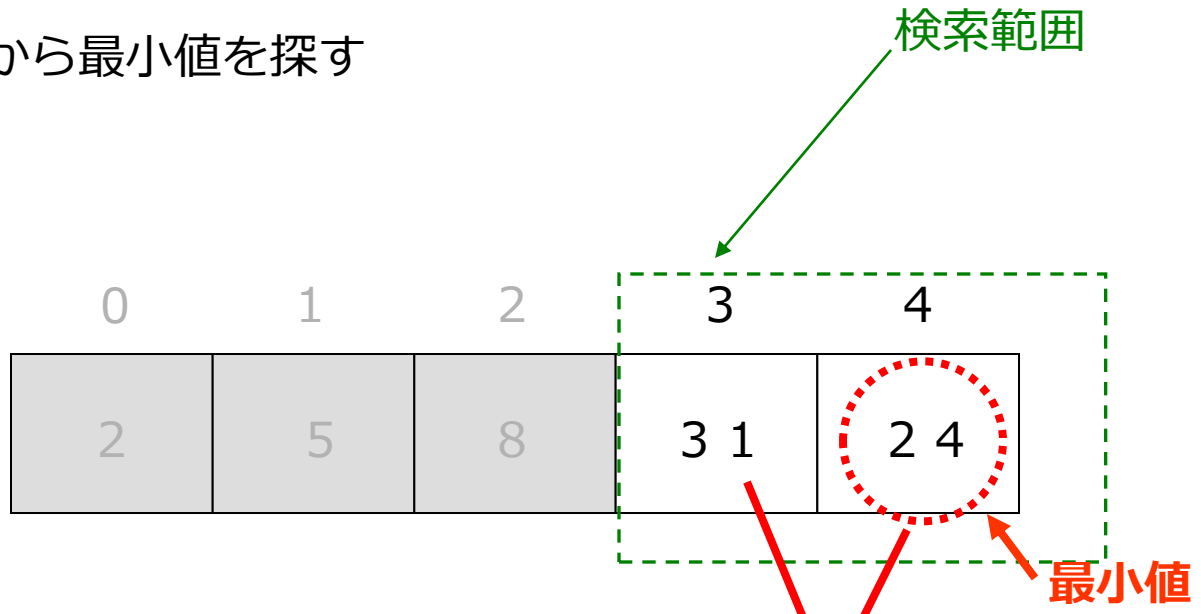


ＳＴＥＰ２：未整列の先頭要素と交換

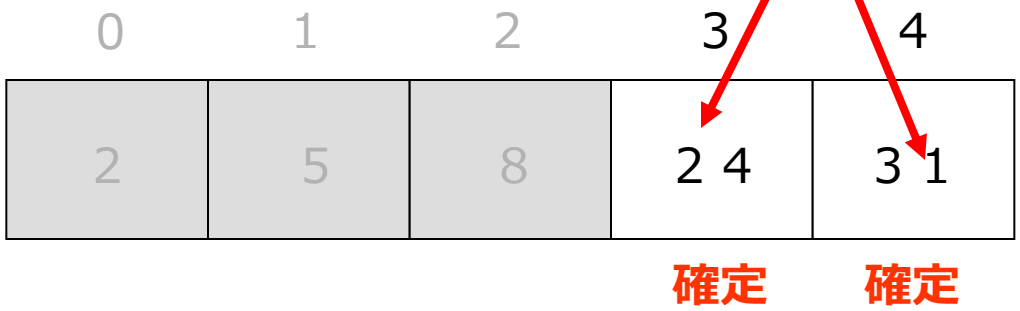


必須問題９－１：選択ソートの例（昇順）４／４

ＳＴＥＰ１：対象範囲から最小値を探す



ＳＴＥＰ２：未整列の先頭要素と交換



必須問題９－２：挿入ソート（データ構造とアルゴリズムを参照）

●アルゴリズム

- 大きい順または小さい順に並んでいる数列に、ある数を順に比較しながらその数列に**挿入**し並び替えることで整列する手法

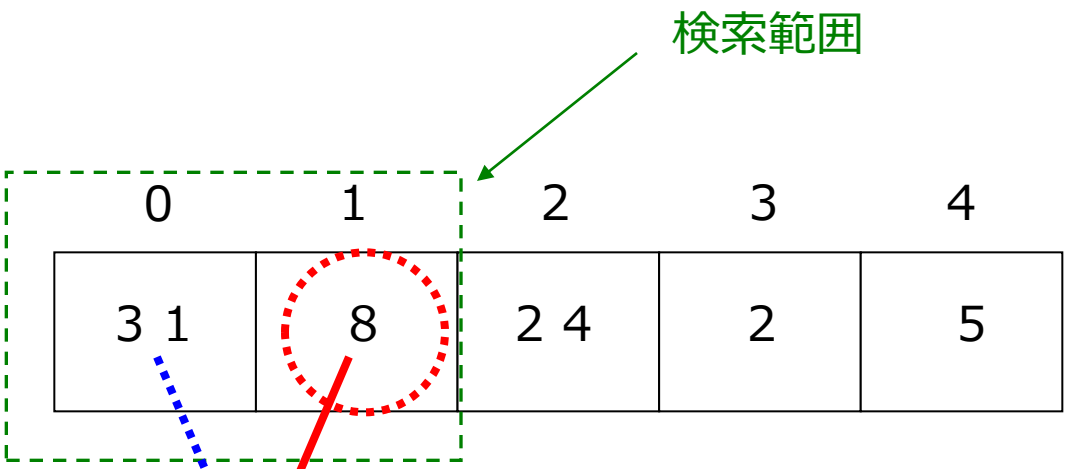
挿入ソートのポイント

●値を入れ替えるのではなく、挿入する：その時点での整列済みの配列に対象要素を挿入。

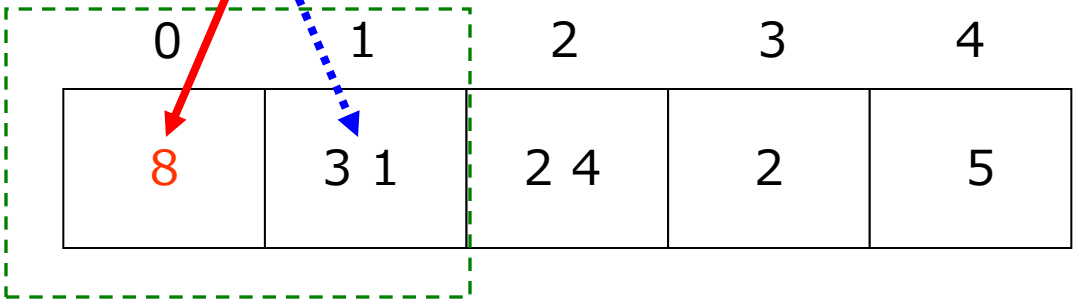
●整列処理の終了：最後の要素が確定したら、整列処理を終了する

必須問題９－２：挿入ソートの例（昇順） １／４

ＳＴＥＰ１：対象範囲の最後の要素を適切な場所に挿入

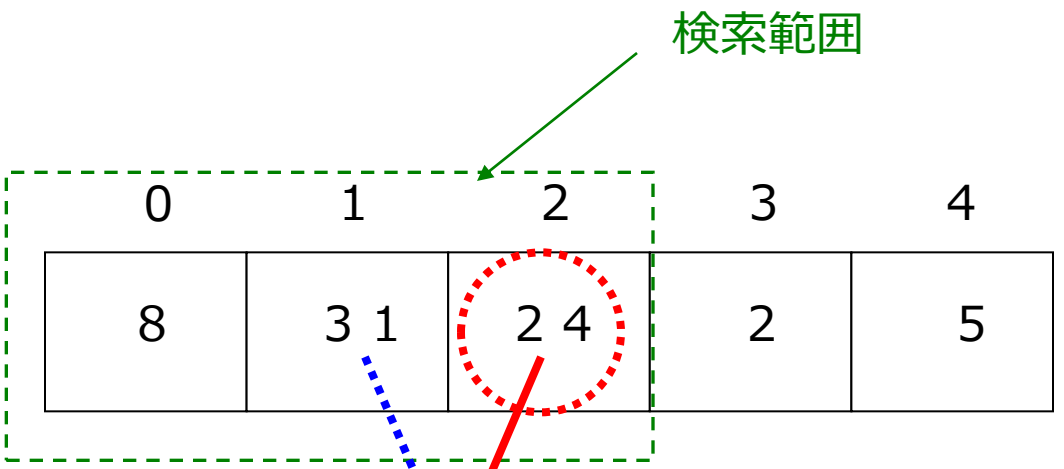


ＳＴＥＰ２：挿入された要素より後ろの要素を移動

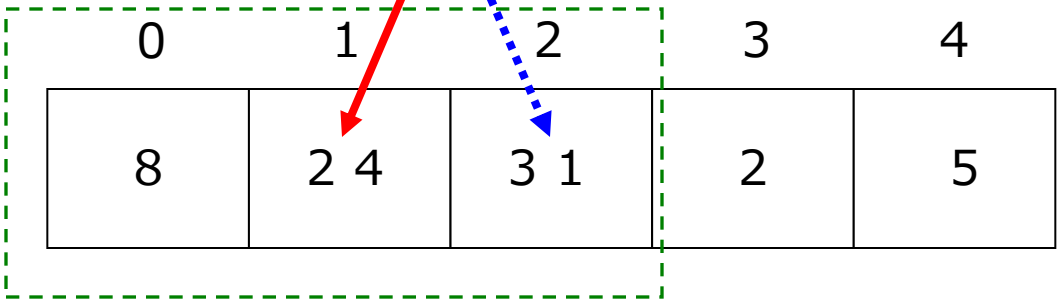


必須問題９－２：挿入ソートの例（昇順）２／４

ＳＴＥＰ１：対象範囲の最後の要素を適切な場所に挿入

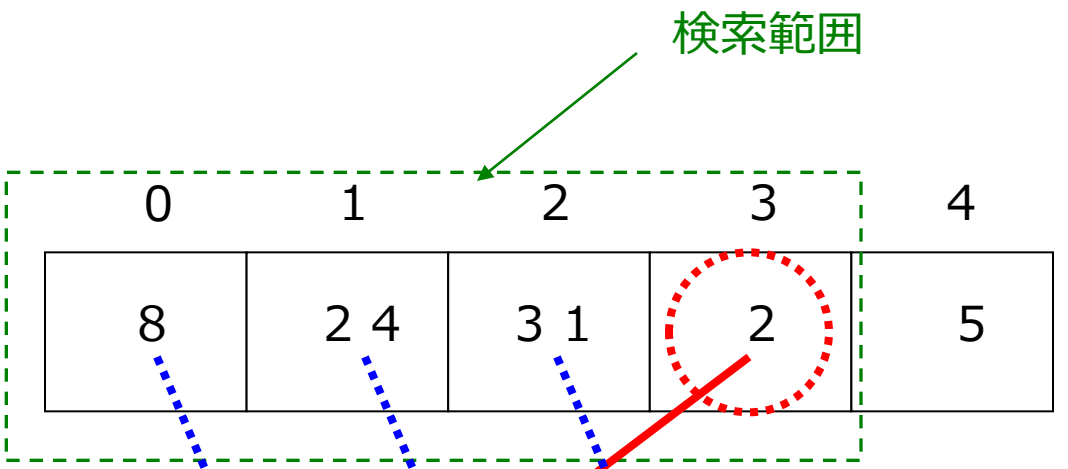


ＳＴＥＰ２：挿入された要素より後ろの要素を移動

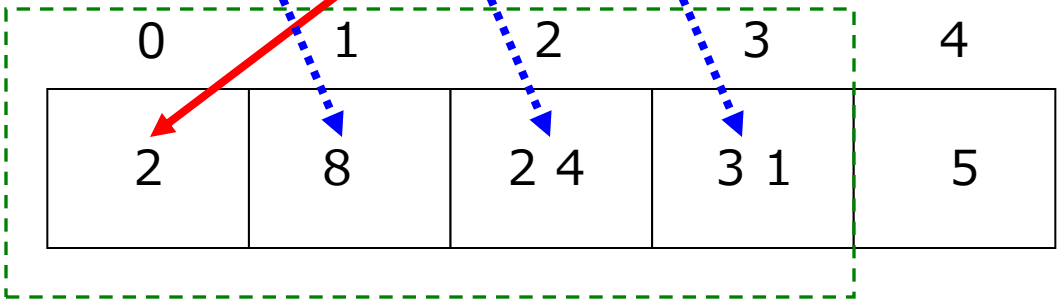


必須問題９－２：挿入ソートの例（昇順） ３／４

ＳＴＥＰ１：対象範囲の最後の要素を適切な場所に挿入

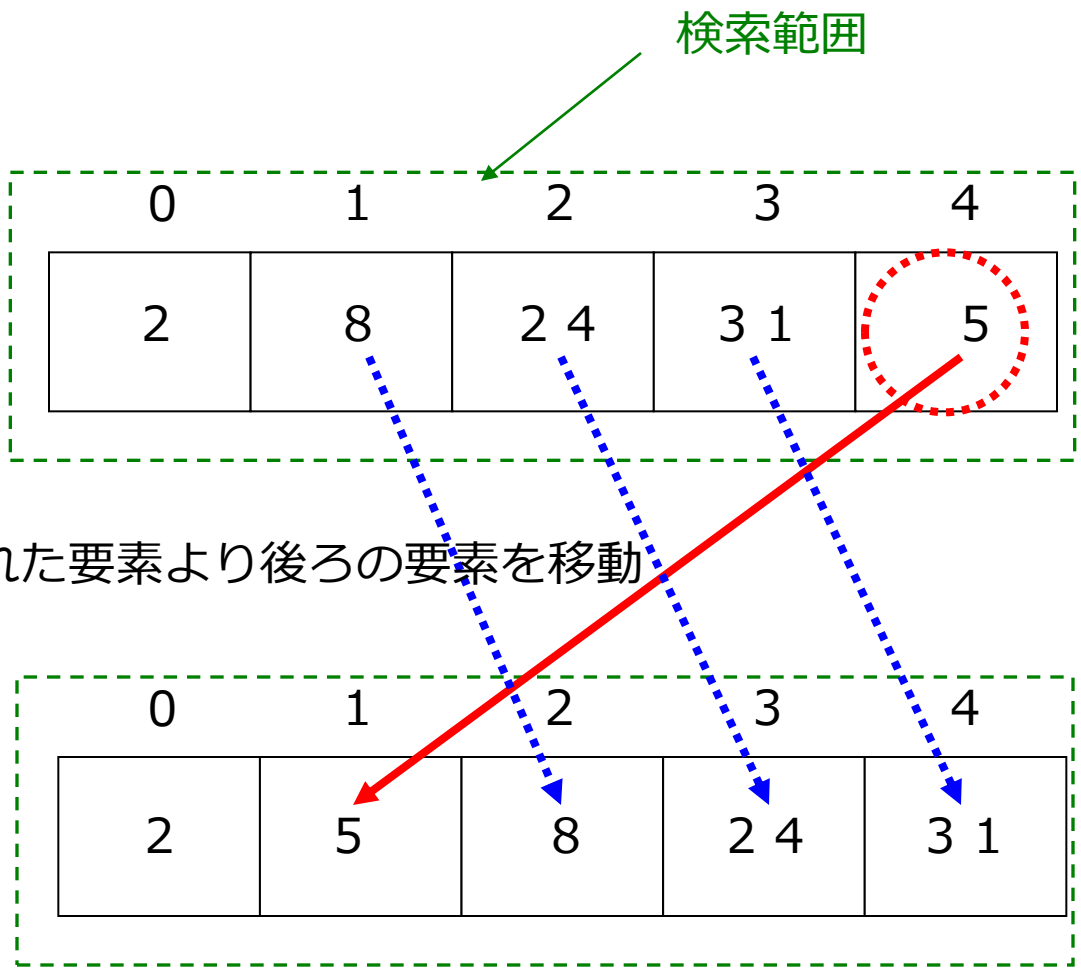


ＳＴＥＰ２：挿入された要素より後ろの要素を移動



必須問題９－２：挿入ソートの例（昇順）４／４

ＳＴＥＰ１：対象範囲の最後の要素を適切な場所に挿入



オプション課題９－３：バブルソート

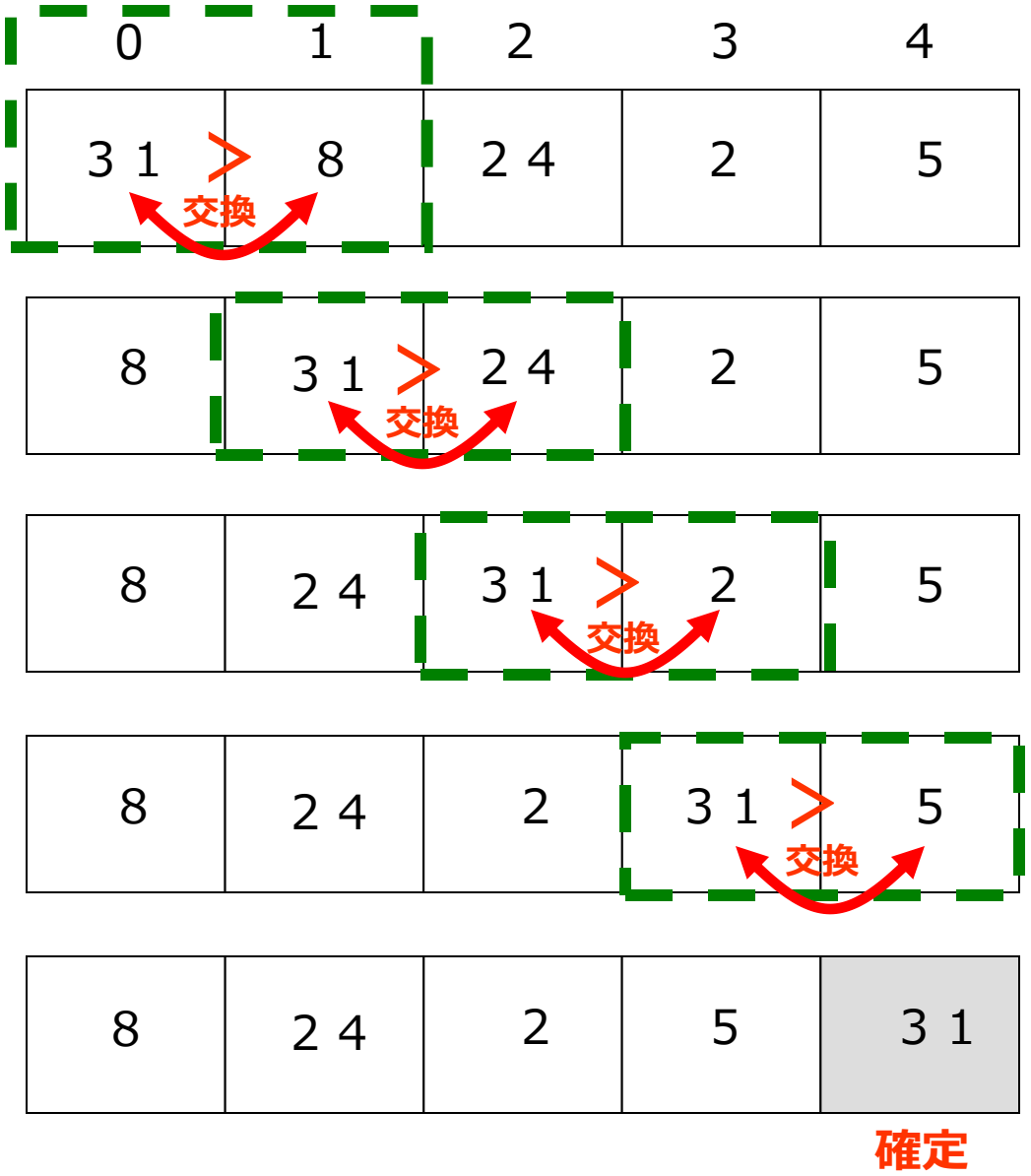
●アルゴリズム

- **隣どうし**の要素の大小を比較して、それらを交換しながら整列する手法

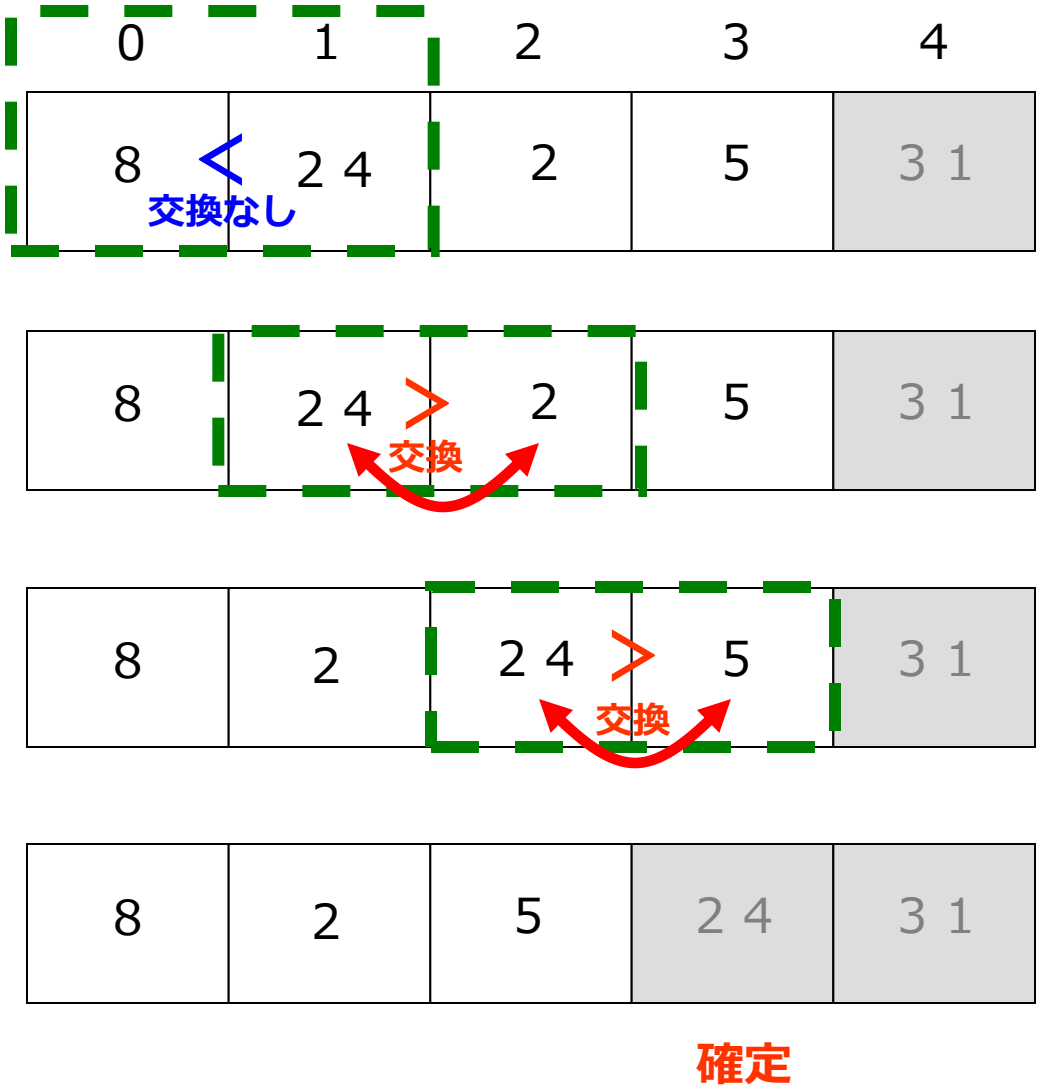
●バブルソートのポイント

- 隣どうしの配列要素を比較する：比較する要素は、配列の先頭から順に後ろへずれて行く。
- 入れ替えを行なう：隣どうしが逆順(大または小)になっていたら、要素を交換する。
- 整列処理の終了：２番目(要素番号１)の要素の値が確定したら、整列処理は終了する。

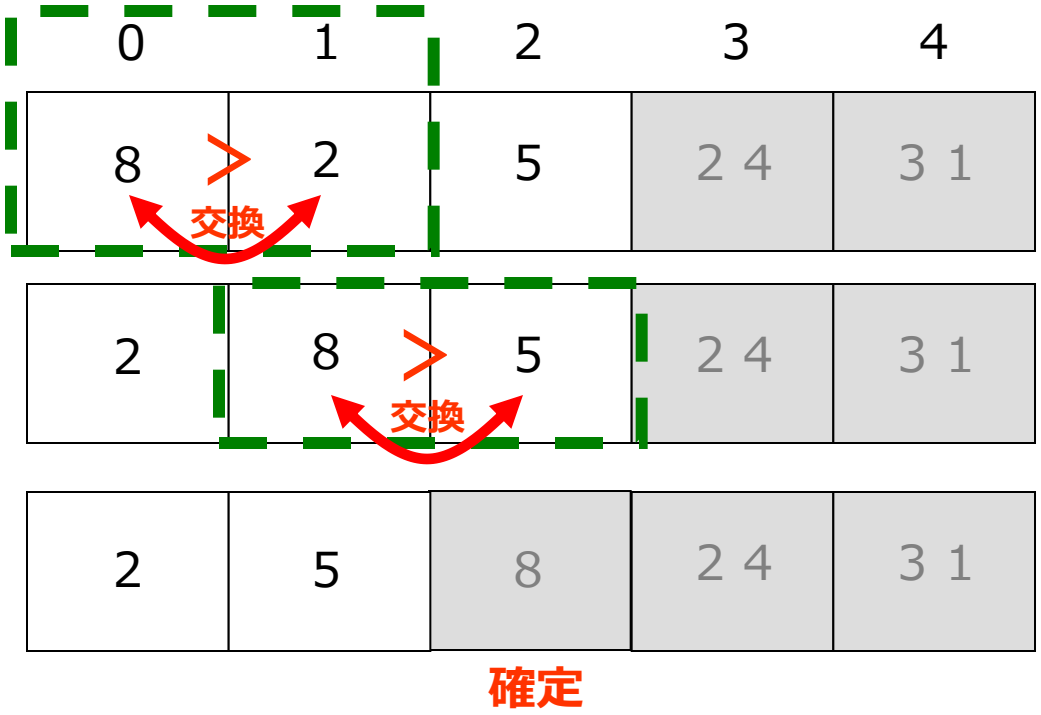
オプション課題９－３：バブルソートの例１／3

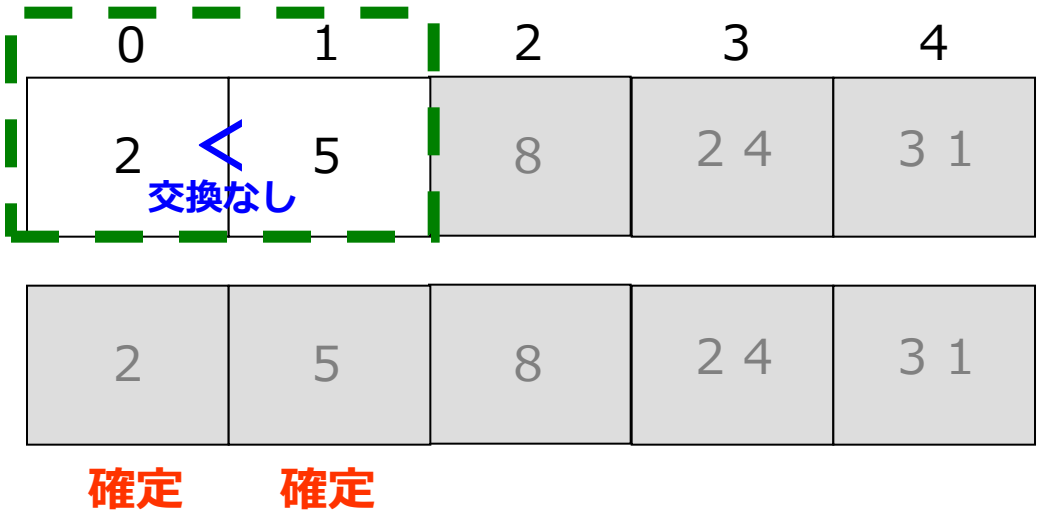


オプション課題９－３：バブルソートの例２／3



オプション課題９－３：バブルソートの例３／3





ソートの中間状態を出力

●確認用にソートの中間状態を出力しましょう。
（参考プログラムは次ページ）

必須問題9-1のチェック用出力例

```
Array = [ 91 63 71 14 60 1 24 13 80 15 ]
-- start selection sort --
Array = [ 1 63 71 14 60 91 24 13 80 15 ]
Array = [ 1 13 71 14 60 91 24 63 80 15 ]
Array = [ 1 13 14 71 60 91 24 63 80 15 ]
Array = [ 1 13 14 15 60 91 24 63 80 71 ]
Array = [ 1 13 14 15 24 91 60 63 80 71 ]
Array = [ 1 13 14 15 24 60 91 63 80 71 ]
Array = [ 1 13 14 15 24 60 63 91 80 71 ]
Array = [ 1 13 14 15 24 60 63 71 80 91 ]
Array = [ 1 13 14 15 24 60 63 71 80 91 ]
Array = [ 1 13 14 15 24 60 63 71 80 91 ]
Array = [ 1 13 14 15 24 60 63 71 80 91 ]
```

参考プログラム：9-1,9-2,9-3共通

```
#define MAX_DATA
int main(){
    int numbers[MAX_DATA];
    /* 略：変数宣言、numbers.datのファイル読込み */

    printArray(numbers, n);

    for (          ) {
        ソート処理

        for (          ) {
            ソート処理

        }

        ソート処理

        printArray(numbers, n);
    }
    return 0;
}
```

配列のサイズを定義

ソート前の配列の表示

nは読み込んだ要素数

ループ条件

ここの処理が必要ない
ソートもあります

ループ条件

ソート処理

ここの処理が必要ない
ソートもあります

ソート処理の確認用
printArrayは次のページ

参考プログラム：9-1,9-2,9-3共通(printArray)

```
void printArray(int numbers[], int length){
    int i=0;
    printf("Array = [ ");
    for (i = 0; i < length; i++) {
        printf("%d ", numbers[i]);
    }
    printf("]¥n");
}
```

その他の注意

- 同じアルゴリズムでも、多少実現方法に違いがあることがある
  - 昇順・降順
  - 前から/後ろからソート済みにする

著者リスト

1. 安積 卓也（情報システム学科）
2. 大森 隆行（情報システム学科）