# 第１１週
# ソート（３）
# ヒープソート、マージソート
# クイックソート

# 必須問題11-1:ヒープソート

●アルゴリズム
選択法（最小値選択を反復して整列）に
ヒープの概念を取り入れて効率よく整列を行う手法

ヒープソートの手順

●ヒープ木を作成する
- 10-3のinsertを利用

●ヒープ木のルート取り出す
- 10-4deleteminを反復実行して、ヒープの再構成を反復して行い、各時点での最小値を取り出す

●取り出した最小値を保存
- 昇順（小さい順）なら ⇐ 今回はこっち
  - 配列の前から保存、
  - または、キューに保存して、後から取り出す
- 降順（大きい順）スタックに格納し、
  - 配列の後ろから保存、
  - または、スタックに保存して、後から取り出す

# 参考プログラム：11-1(ヒープ：１基準)

```
#define HEAP_SIZE
int main(){
    int numbers[HEAP_SIZE];
    int heap[HEAP_SIZE+1];
    int heap_size = 0;

    /* 略：変数宣言、numbers.datのファイル読込み */
    printArray(numbers, n);

    return 0;
}
```

配列のサイズを定義

ヒープの一次元配列：１基準なので＋１

ソート前の配列の表示

ヒープ木の作成

insertを利用しヒープ木の作成
10-3を参照

値の保存

deletminで値を取り出しnumbersに保存
昇順：
　配列（numbers）の先頭から順に保存
降順：
　配列（numbers）の後ろから順に保存
10-4を参照

```
    printArray(numbers, n);
```

ソート処理の確認用
printArrayは次のページ

# 参考プログラム：printArray

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

# オプション課題11-2：マージソート

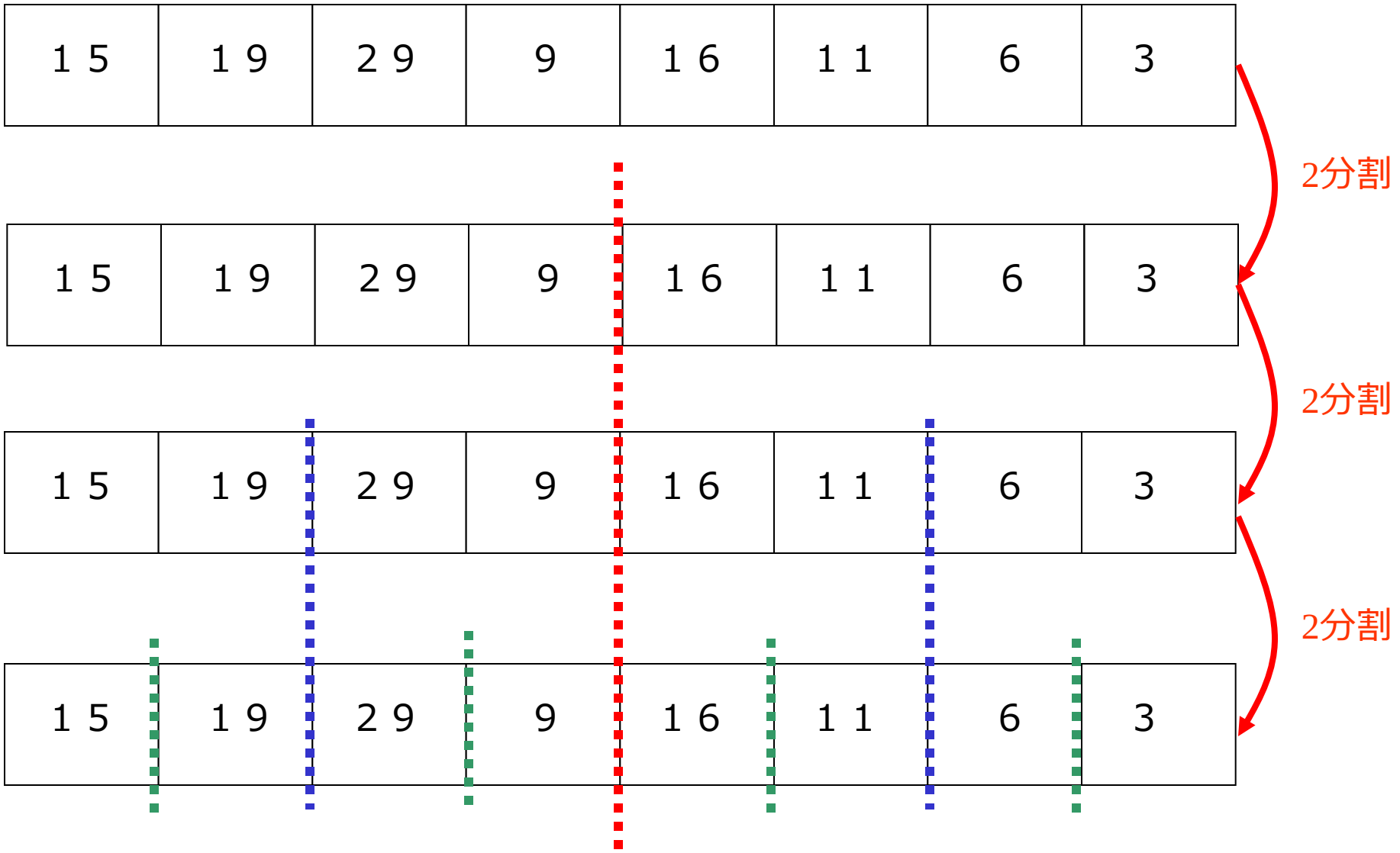
アルゴリズム

入力系列を2つに分割して、各々の系列を整列し、その後、統合（マージ）することで整列を行う手法（分割された系列の整列にもマージソートを使う。よってこれは再帰的に整列を行っている。）

マージソートの手順

- ●入力を分割
  - 再帰で入力系列を2分割を繰り返す
- ●統合（マージ）：昇順の場合
  - 統合する２つの入力の先頭から比較を行い、統合後小さい値を先頭から順に代入していく

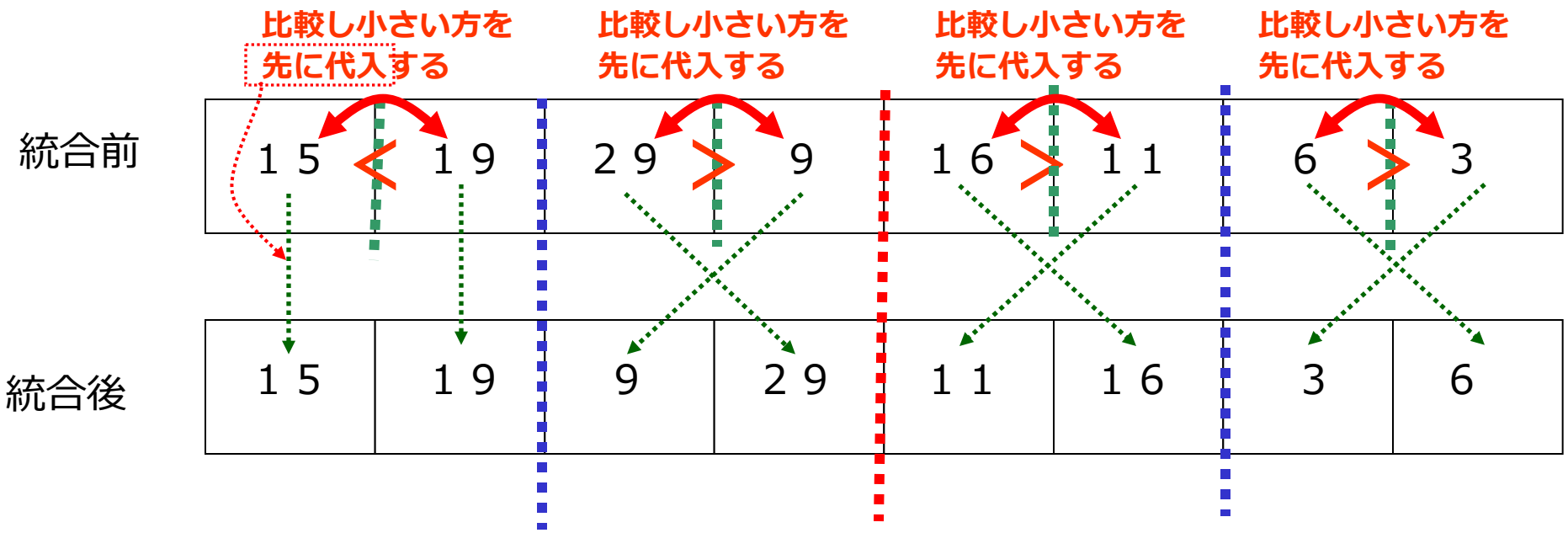
# マージソートの例：分割


15 19 29 9 16 11 6 3
2分割
15 19 29 9 16 11 6 3
2分割
15 19 29 9 16 11 6 3
2分割
15 19 29 9 16 11 6 3

# マージソートの例：統合

●統合（マージ）：昇順の場合

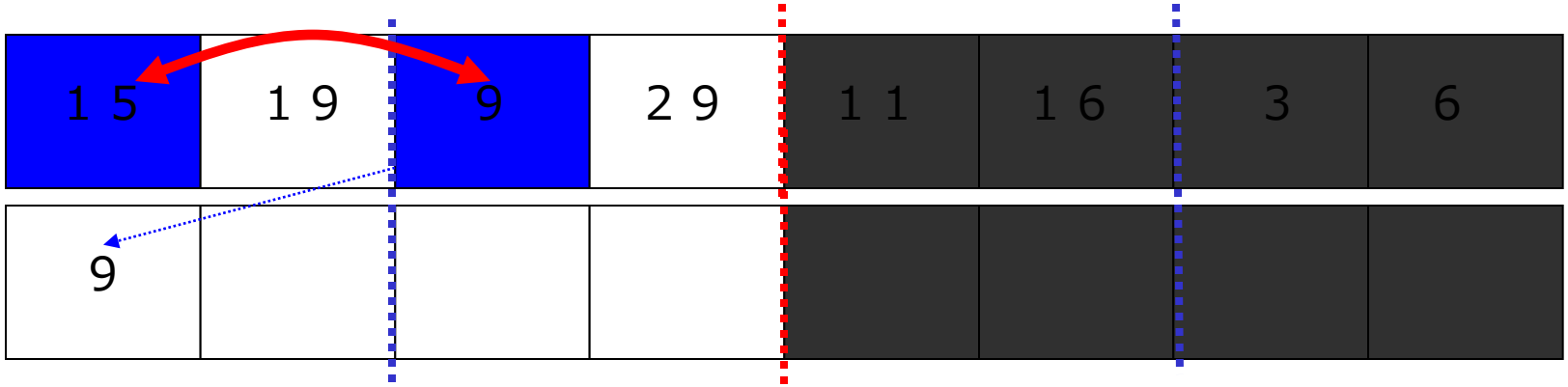
- 2つの入力の先頭から比較を行い、統合後小さい値を先頭から代入していく



# マージソートの例：統合

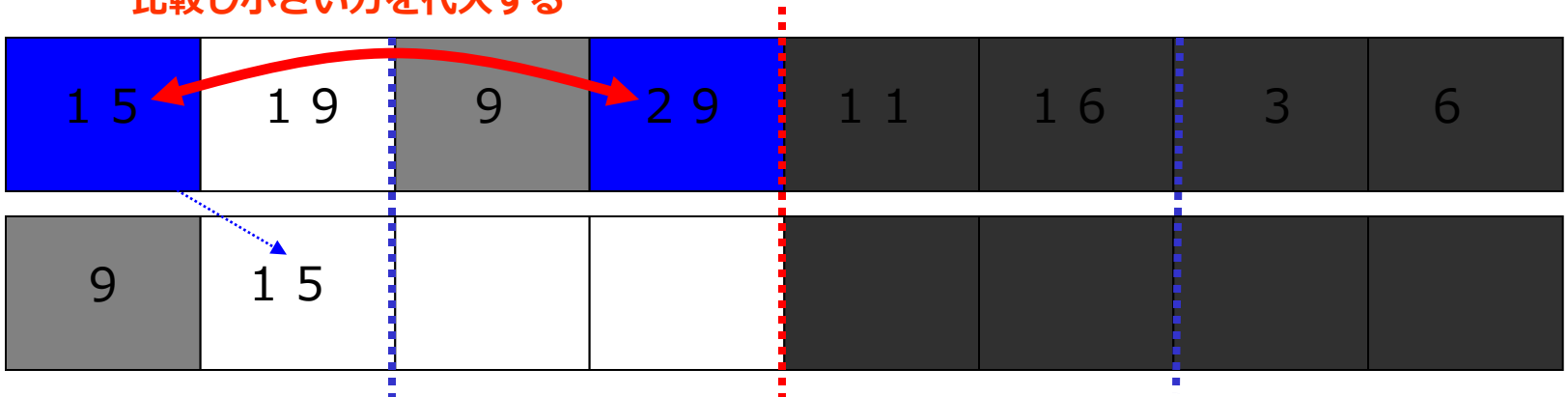
2段階目（左半分）：ステップ１：
左側の先頭（１５）と右側の先頭（９）を比較し、
小さい9が統合後の先頭の値になる

比較し小さい方を代入する



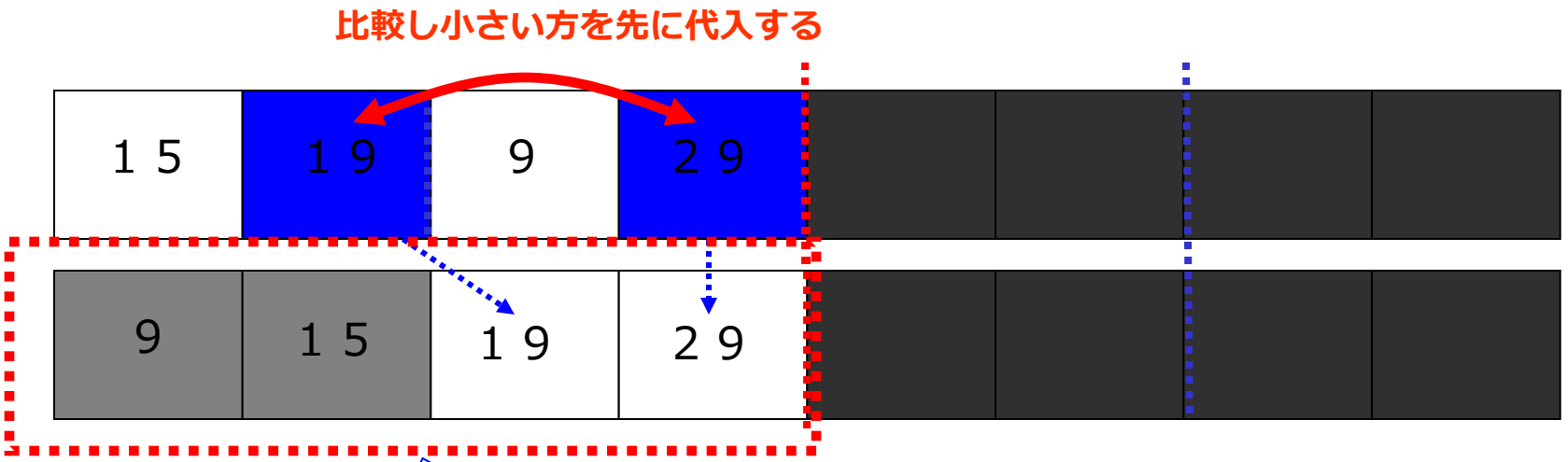

2段階目（左半分）：ステップ２：
左側の先頭（１５）と右側の２番目の値（２９）を比較し、
小さい15が統合後の2番目の値になる

比較し小さい方を代入する



# マージソートの例：統合

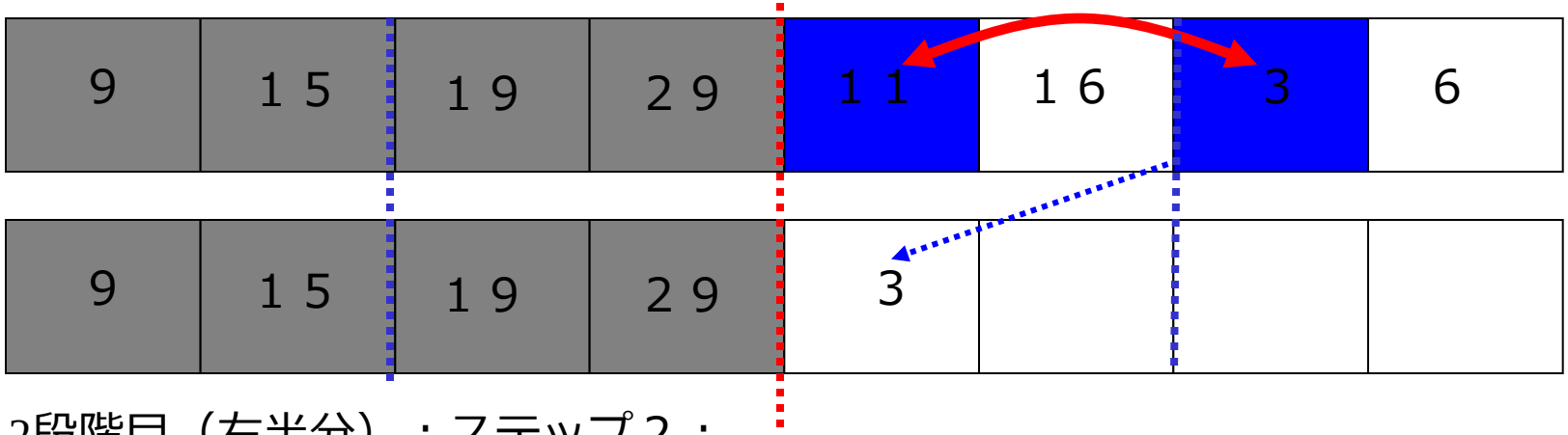
2段階目（左半分）：ステップ３：
左側の２番目（１９）と右側の2番目（２９）を比較し、
小さい１９が統合後の３番目の値になる
大きい２９が統合後の４番目の値になる



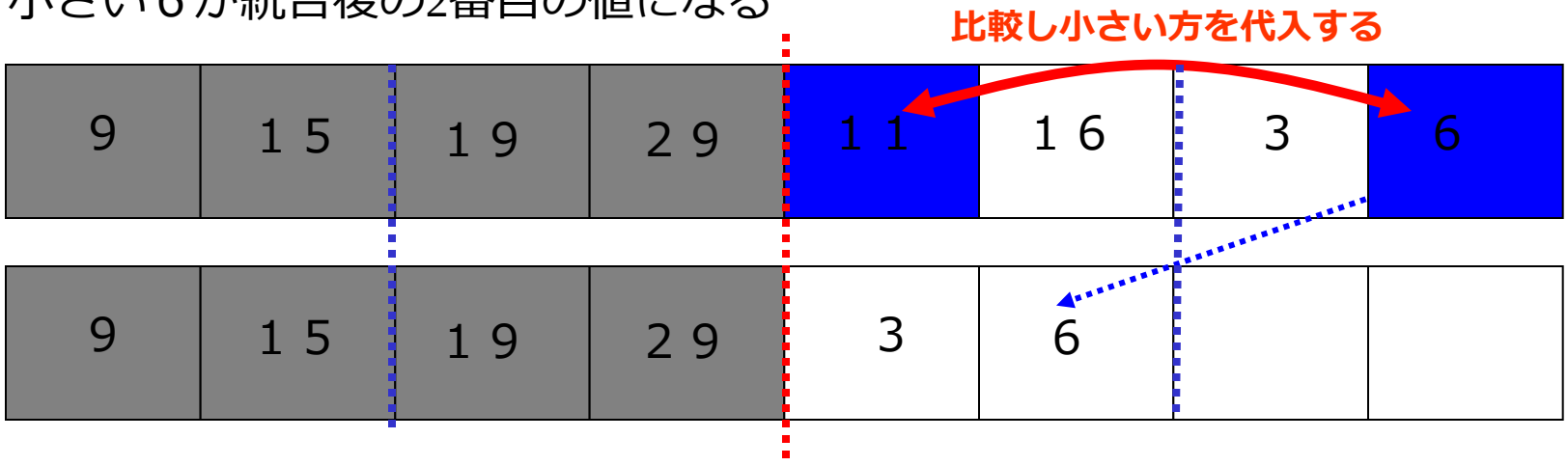

ポイント
統合後は、配列の値が昇順に並んでいる

# マージソートの例：統合

2段階目（右半分）：ステップ１：
左側の先頭（１１）と右側の先頭（３）を比較し、
小さい３が統合後の先頭の値になる

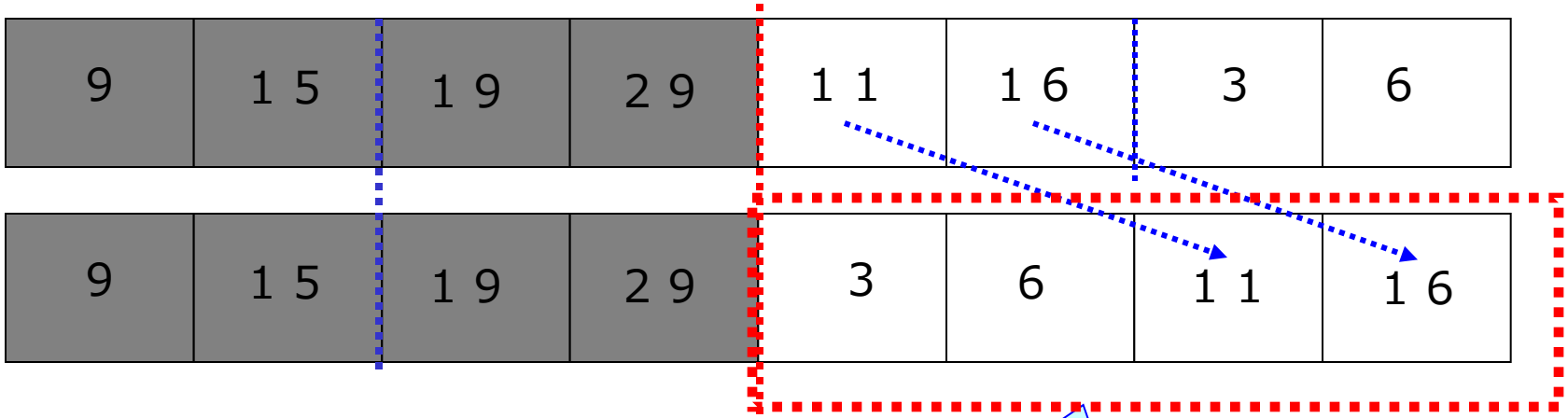



2段階目（右半分）：ステップ２：
左側の先頭（１１）と右側の２番目の値（６）を比較し、
小さい６が統合後の2番目の値になる



# マージソートの例：統合

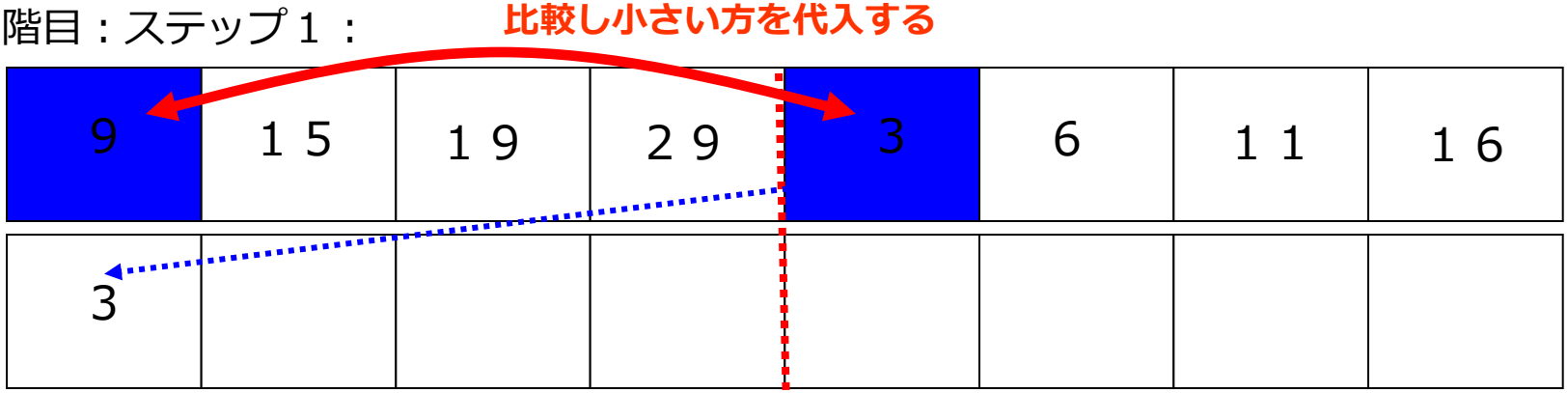
2段階目（右半分）：ステップ３：
左側の先頭（１１）右側の2番目を（１６）を比較し、
それぞれ統合後の３番目と4番目の値になる



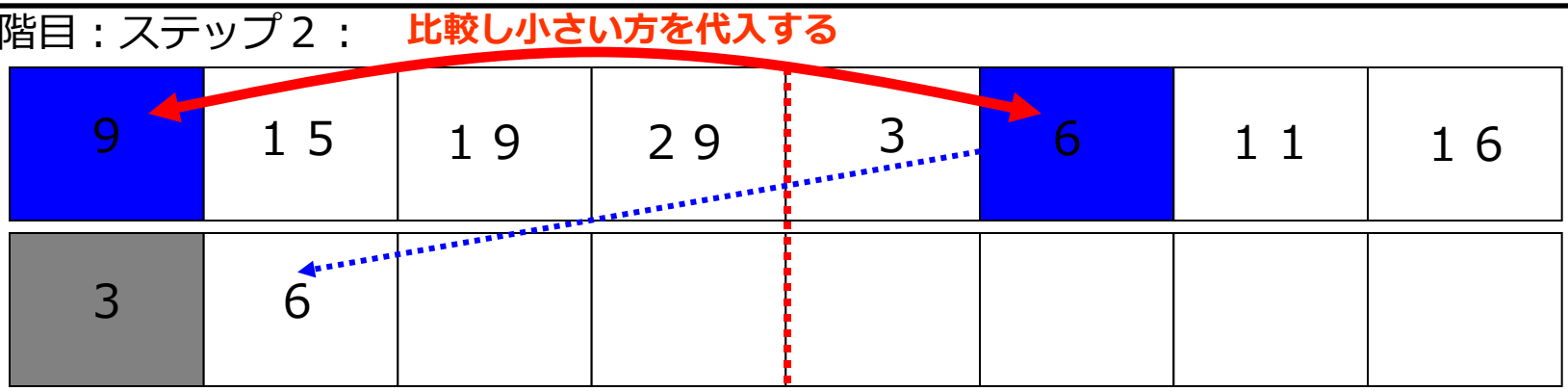

ポイント
統合後は、配列の値が昇順に並んでいる

# マージソートの例：統合

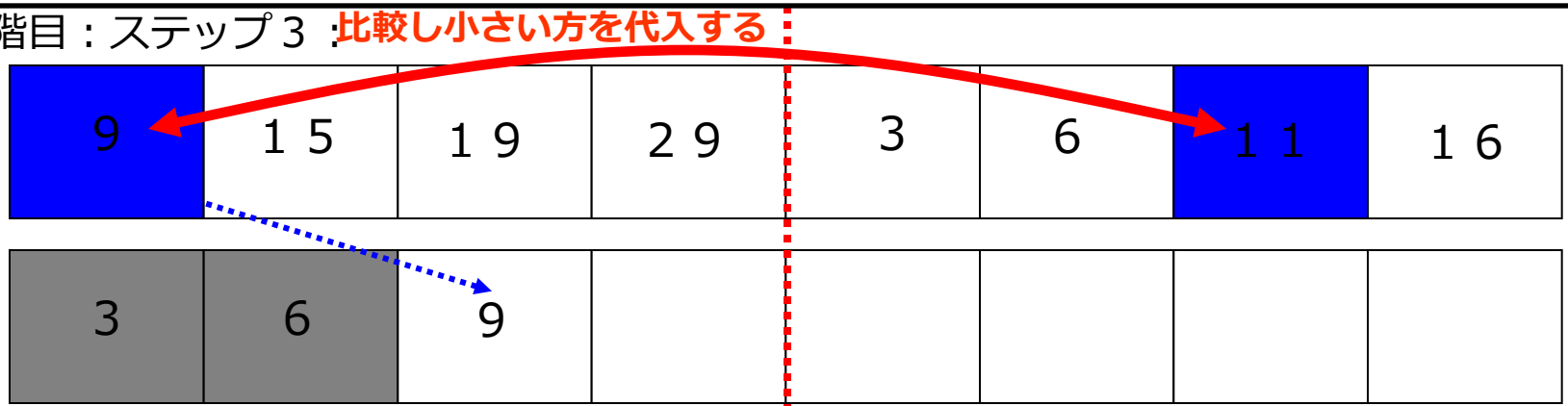
1段階目：ステップ1：




1段階目：ステップ2：

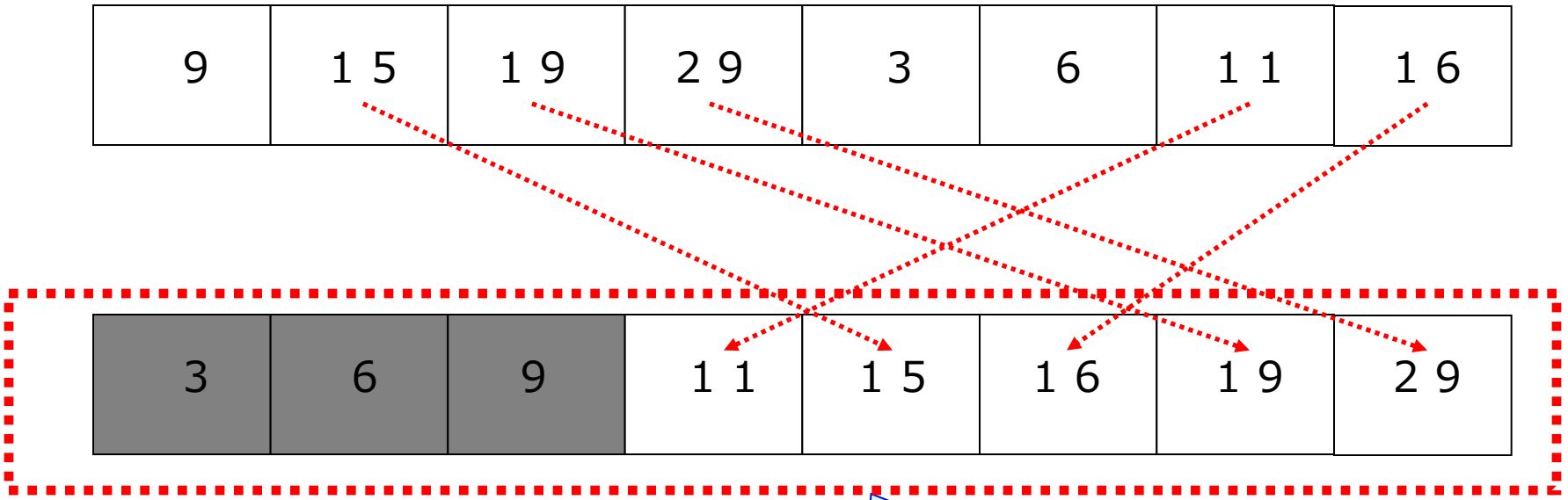



1段階目：ステップ3：



# マージソートの例：統合

残りのステップも同様に比較を行い小さい方を代入していく




全て統合し終わると昇順に値が並んでいる

# オプション課題11-3：クイックソート

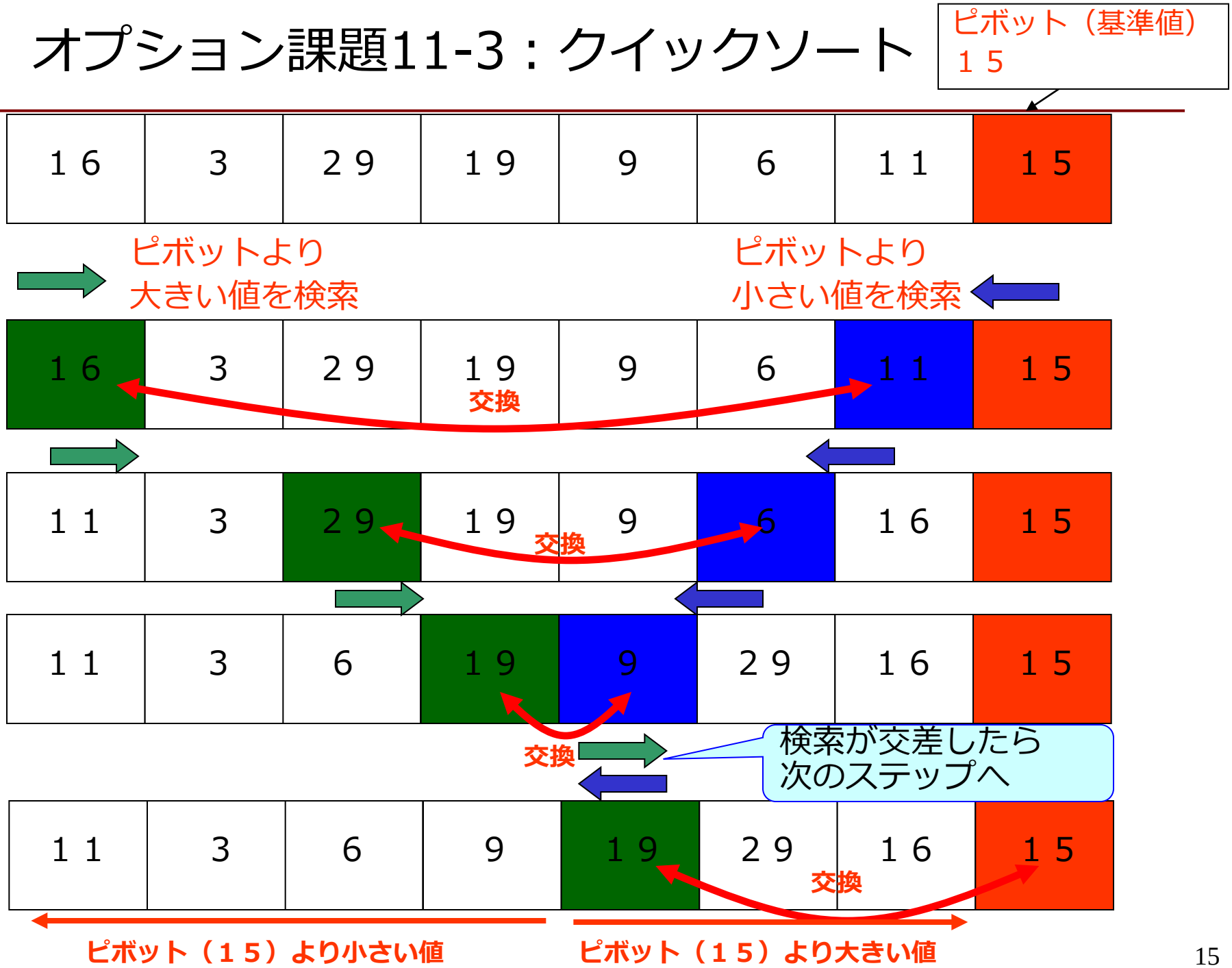
- アルゴリズム
- 入力系列を2つに分割（必ずしも等分割ではない）して、各々の系列を整列し、その後、あわせる。
  また整列された部分集合を統合する必要がない。
- クイックソートの手順（昇順の場合）

Step1:ピボット（基準値）を決める。
ピボットの決め方は、様々（例えば、一番後ろの値）

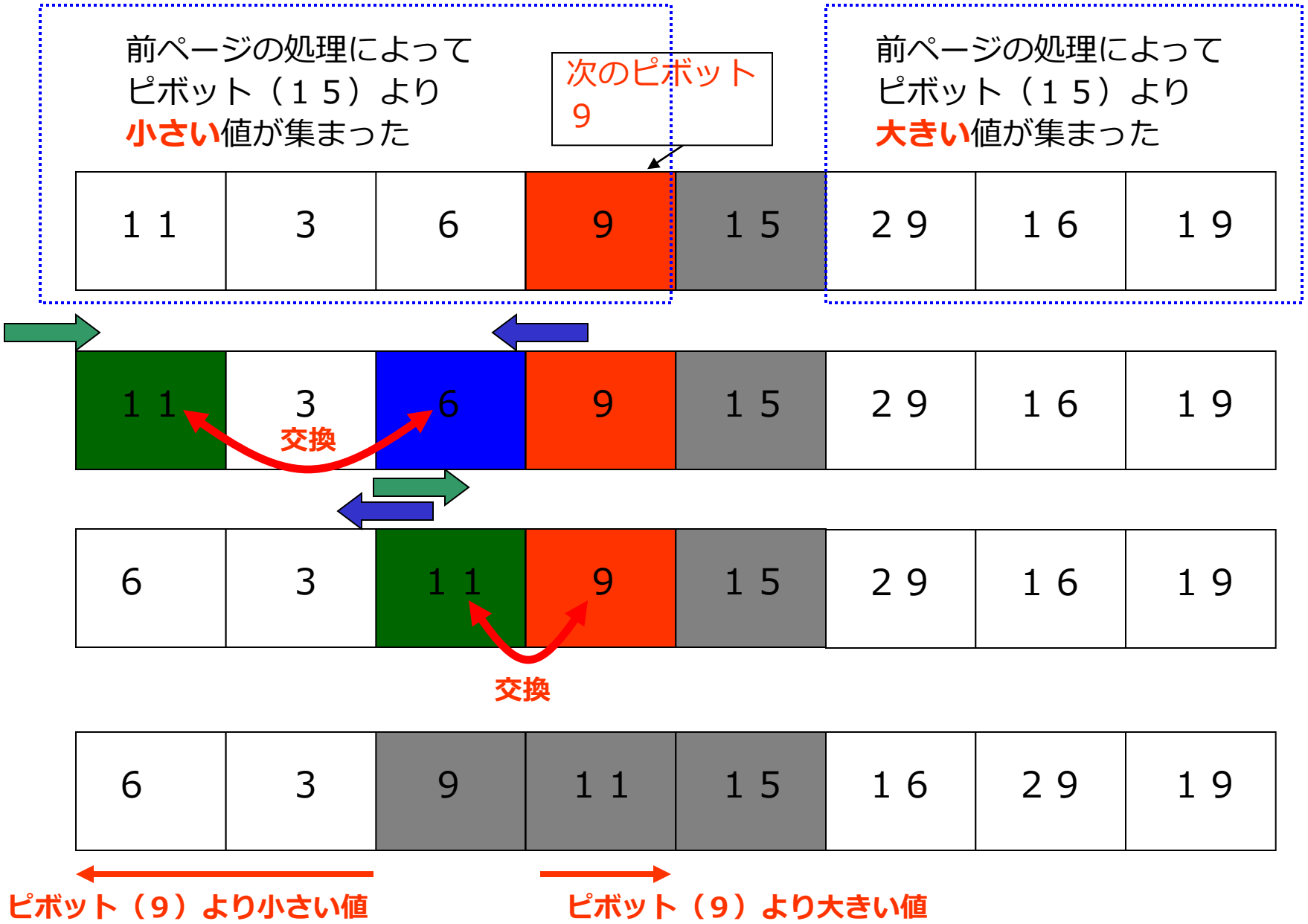
Step2:ピボットより小さい値を左に、
ピボットより大きい値を右に集める。
（左からピボットより大きい値し、
右からピボットより小さい値を探し、入れ替える）

左右に分けた入力について、
新たにピボットを決め,
分割できなくなるまで、
Step1&Step２を繰り返す

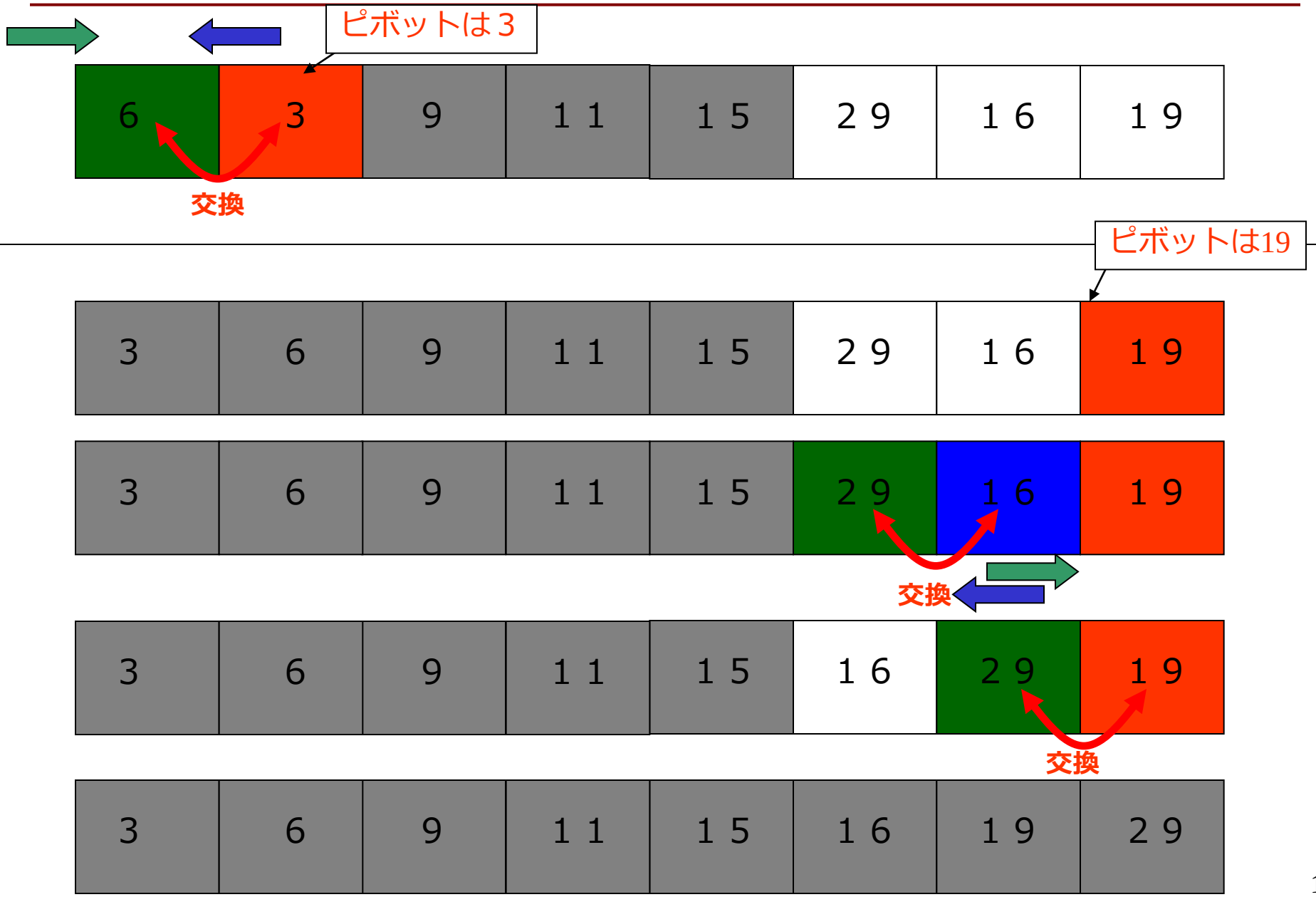
# オプション課題11-3：クイックソート



# オプション課題11-3：クイックソート



# オプション課題11-3：クイックソート



# 著者リスト

1. 安積 卓也（情報システム学科）