プログラミング応用

http://bit.ly/ouyou3d

アルゴリズム（4）

後期 第2週

2018/9/27

本日は・・・

- クイックソートの概要
- 動作の例
- クイックの実装

クイックソートの概要

- 高速（Quick）なソーティングアルゴリズムとして
  よく知られている
- アルゴリズム基本方針

① 基準になるデータを決める

② 基準データよりも大きなグループと小さなグループに分ける

③ ①, ②を再帰的に繰り返す

動作例①

【例】８人の身長を昇順に並べ替える

175	170	160	168	165	170	155	150

動作例②

基準になるデータを決める
（ここでは配列の中央）

175	170	160	168	165	170	155	150

動作例③

基準データよりも大きなグループと小さなグループに分ける

175	170	160	168	165	170	155	150

160	165	155	150

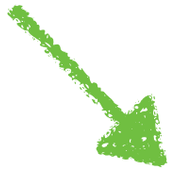
175	170	170

動作例④

分けたグループに対して再帰的に繰り返す

175	170	160	168	165	170	155	150

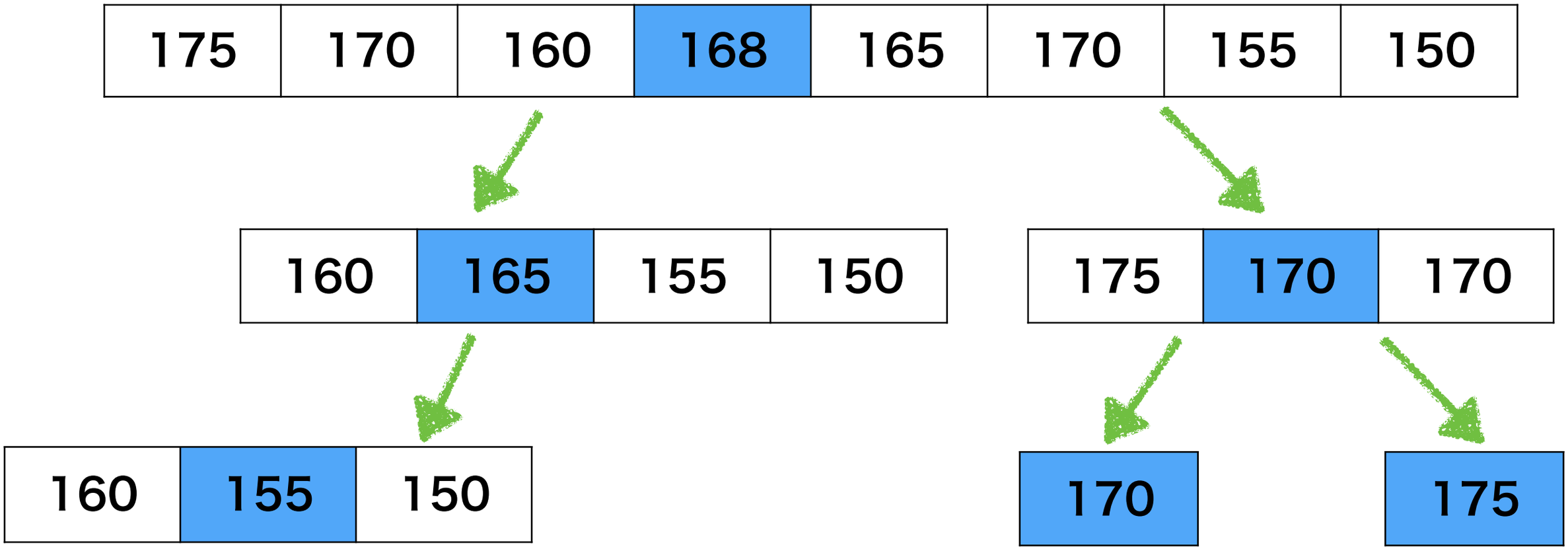
160	165	155	150



175	170	170

動作例⑤

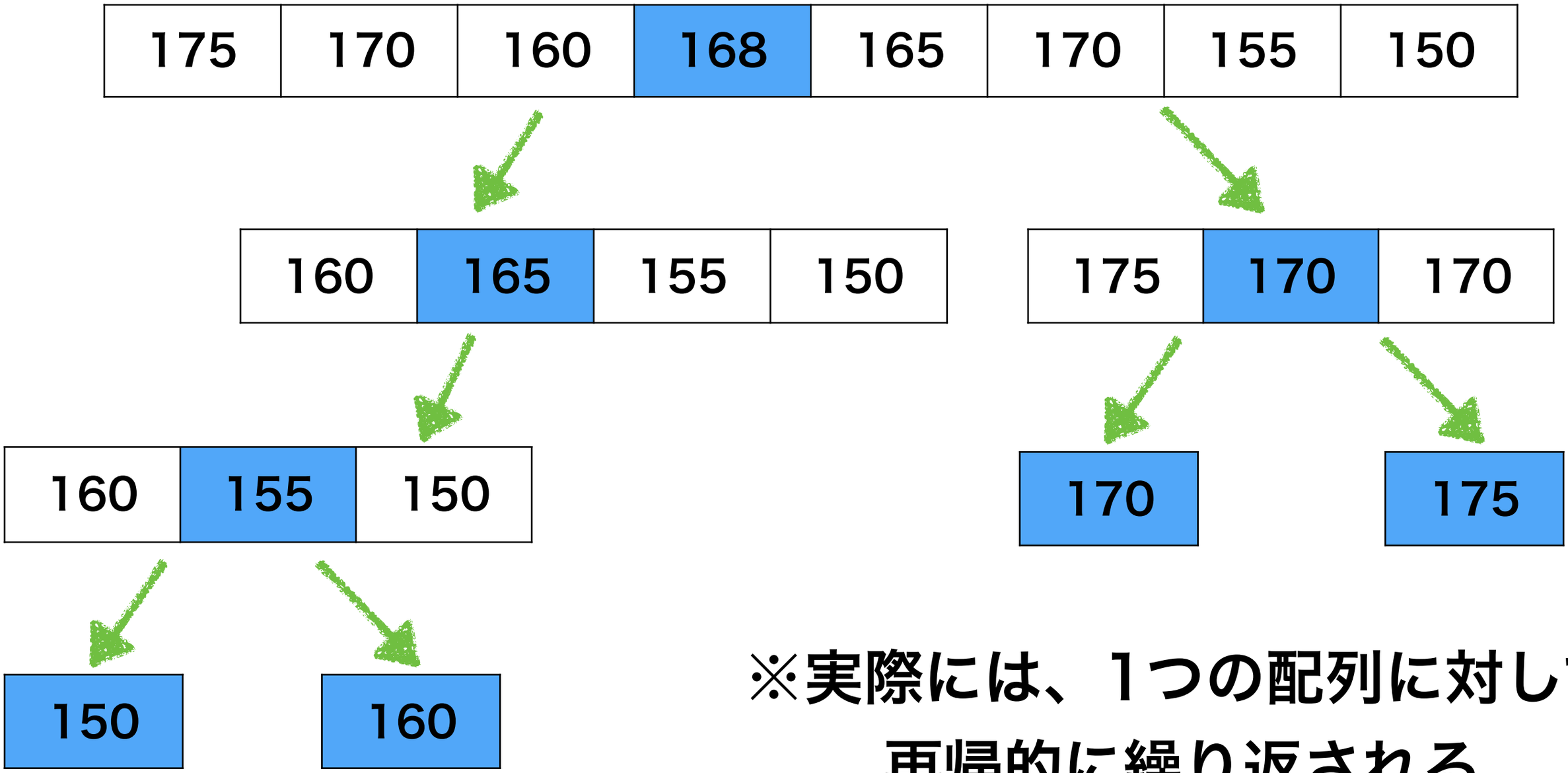
分けたグループに対して再帰的に繰り返す



※実際には、1つの配列に対して
再帰的に繰り返される

動作例⑤

分けたグループに対して再帰的に繰り返す



※実際には、1つの配列に対して
再帰的に繰り返される

アルゴリズムの基本方針

① 基準になるデータを決める

→ 配列の中央の値にする

（他にも基準値の決め方は色々ある）

② 基準データよりも大きなグループと小さなグループに分ける

→ 配列内を基準値でグループに分ける必要がある

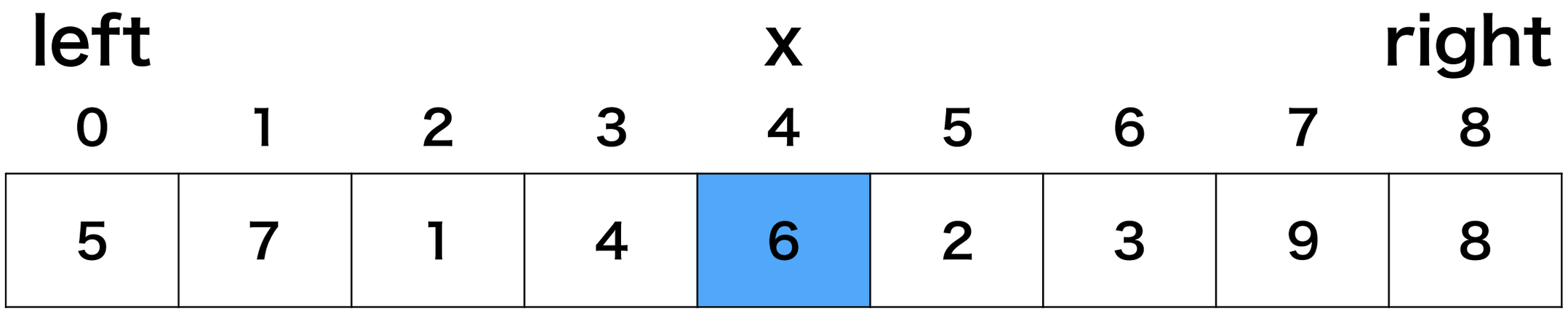
③ ①, ②を再帰的に繰り返す

配列のグループ分け

x = value[(left+ right)/2] ・・・ 基準値（配列中央の値）

left ・・・ グループに分ける配列の左端の添字

right ・・・ グループに分ける配列の右端の添字



x=value[(0+8)/2]　なので、

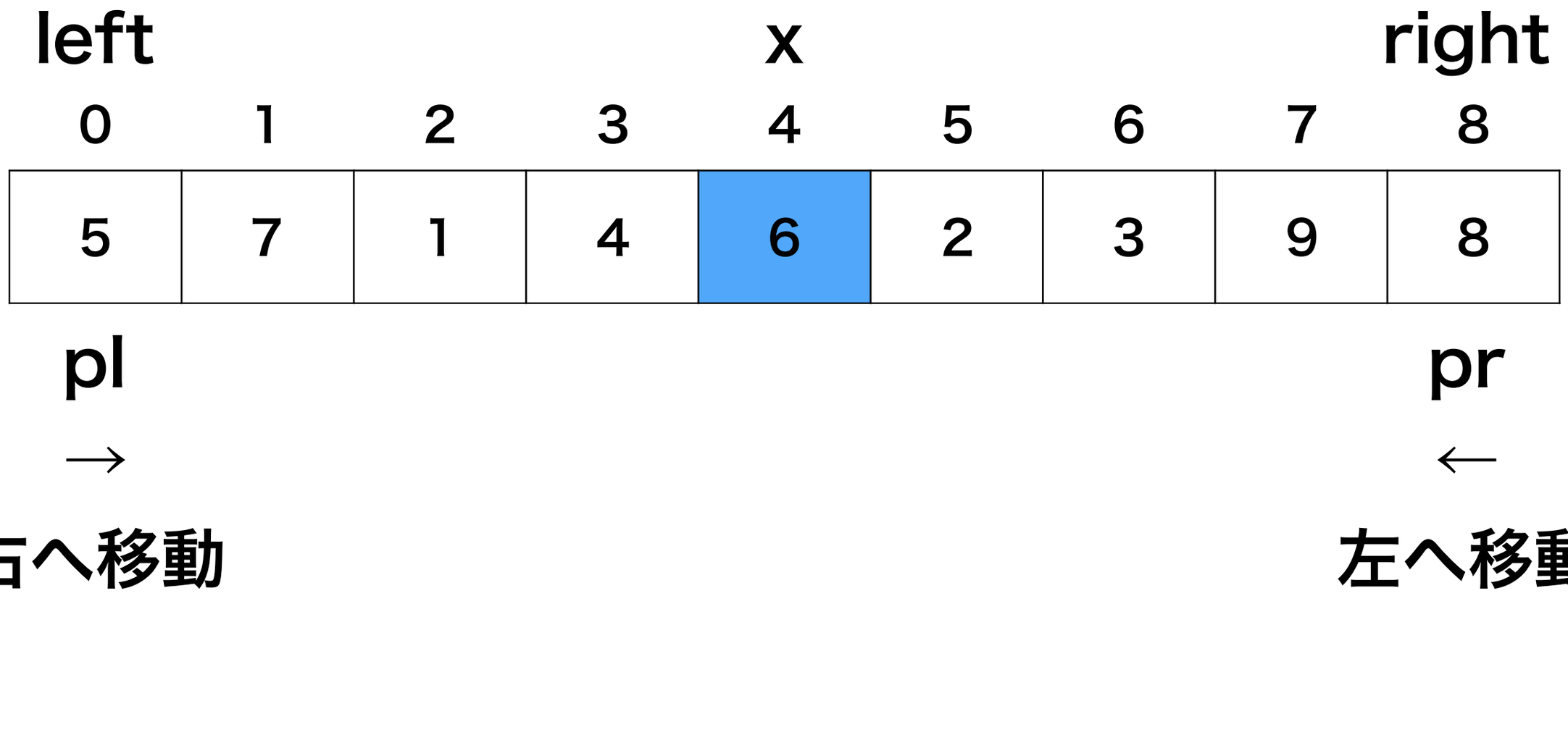
x=value[4]　となり、

このグループ分けの基準値は6となる

配列のグループ分け

基準値6以下の数を左側、6以上の数を右側に移動したい

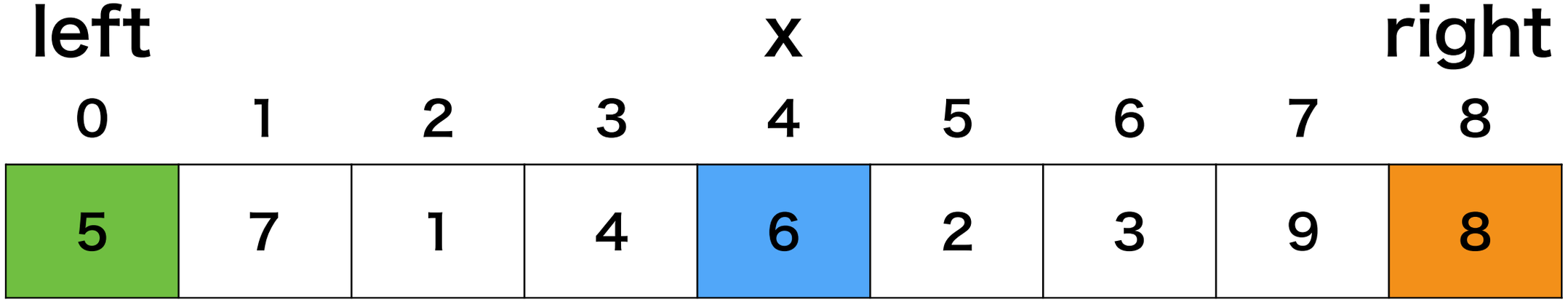
pl ・・・ 左端から右へ移動する添字を表す変数
pr ・・・ 右端から左へ移動する添字を表す変数



配列のグループ分け

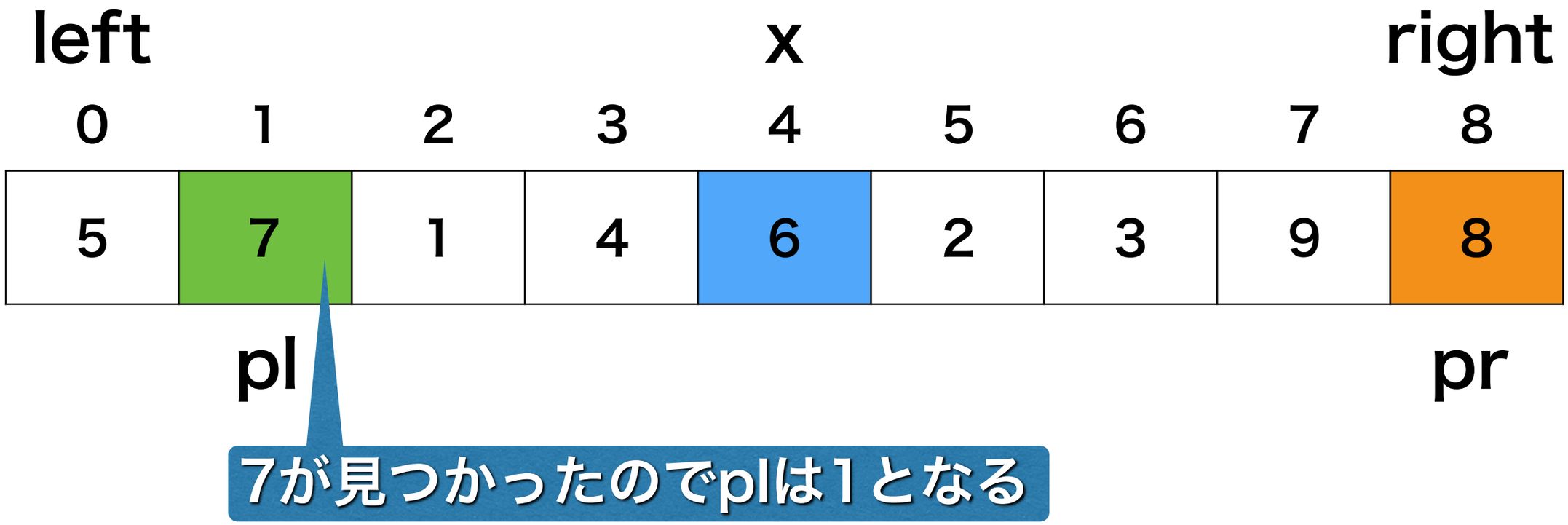
plを左端（left）、prを右端（right）にする

left　　x　　right



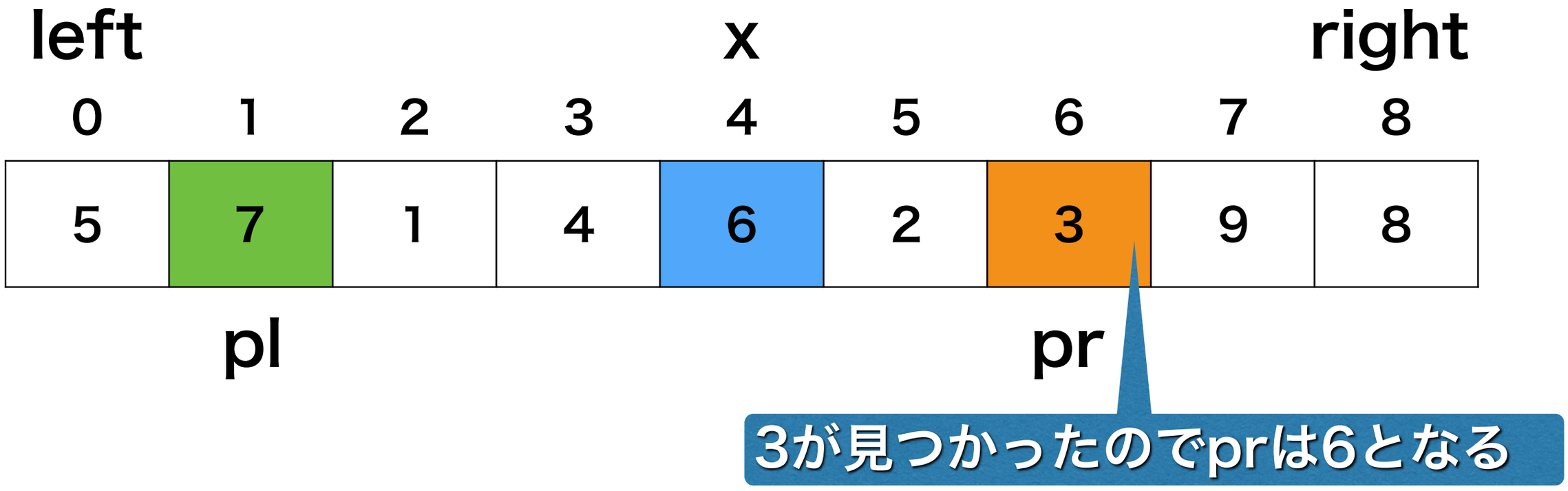
配列のグループ分け

基準値6以上の値が見つかるまでplを右へ移動する
（pl++する）



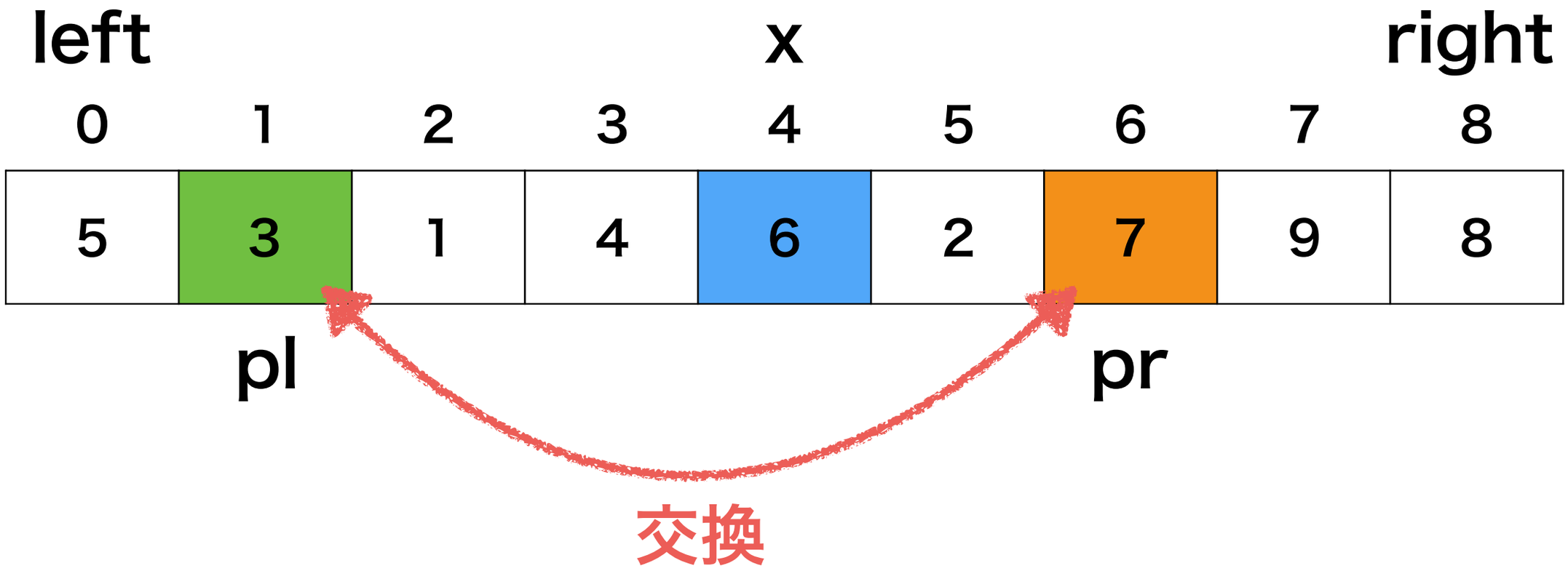
配列のグループ分け

基準値6以下の値が見つかるまでprを左へ移動する
（pr--する）



配列のグループ分け

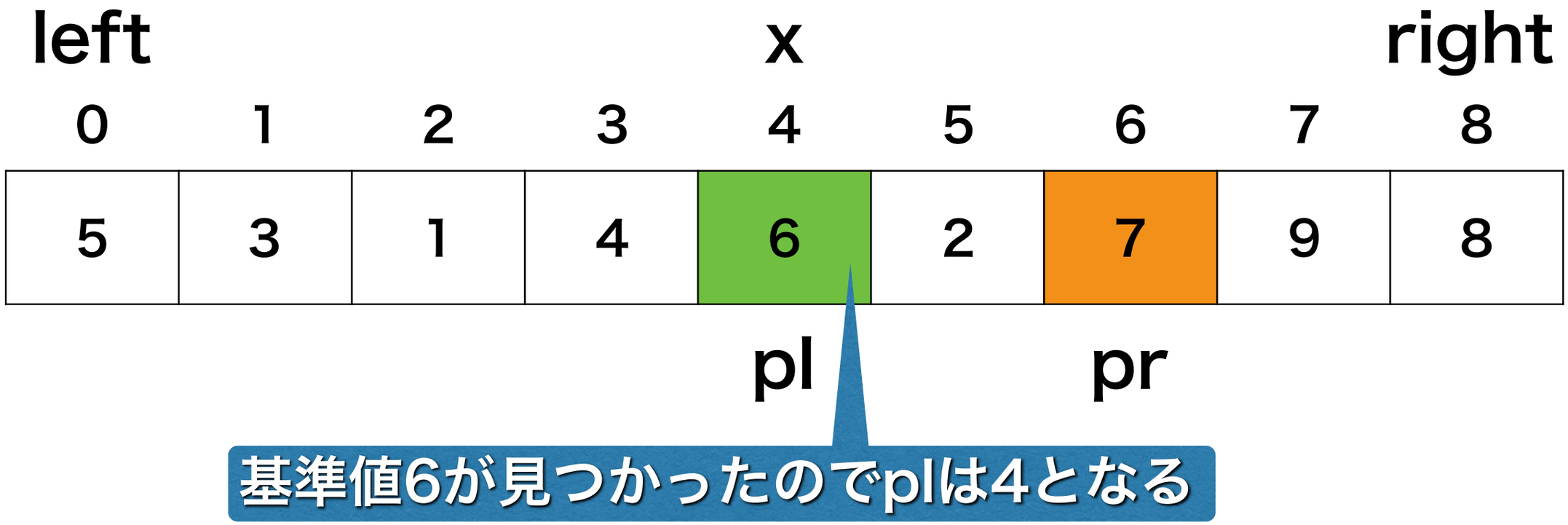
value[pl]とvalue[pr]の値を交換する



この処理を「pl > pr」になるまで繰り返す

配列のグループ分け

基準値6以上の値が見つかるまでplを右へ移動する
（pl++する）



配列のグループ分け

基準値6以下の値が見つかるまでprを左へ移動する
（pr--する）

left　　x　　right

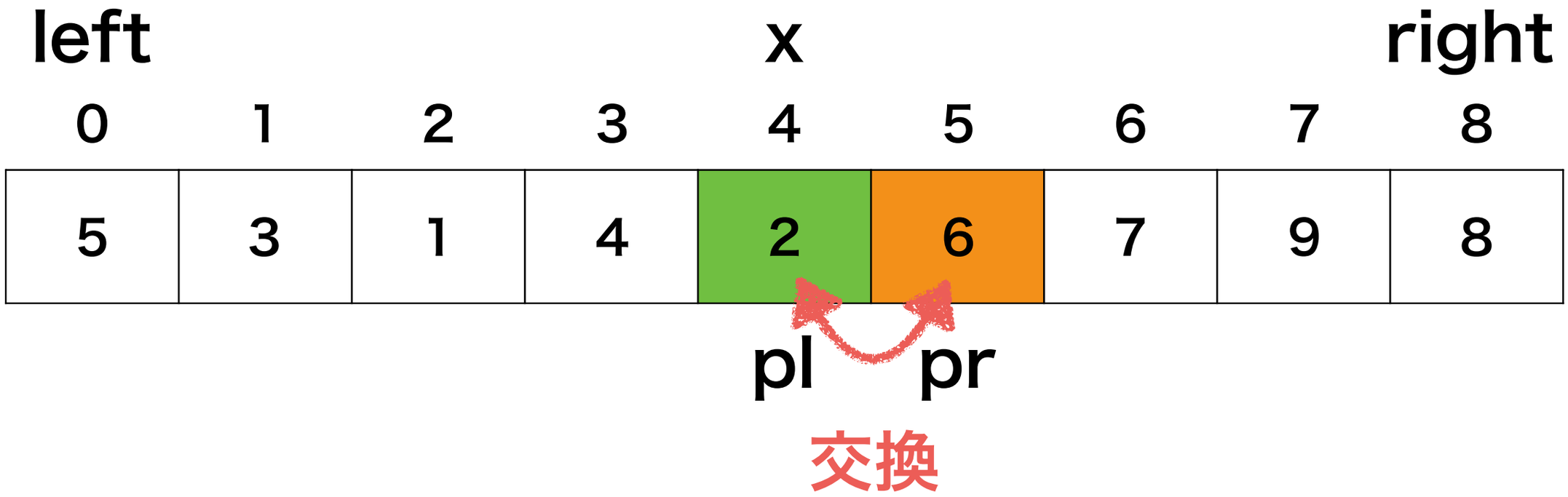
0	1	2	3	4	5	6	7	8
5	3	1	4	6	2	7	9	8

pl　　pr

2が見つかったのでprは5となる

配列のグループ分け

value[pl]とvalue[pr]の値を交換する



この処理を「pl > pr」になるまで繰り返す

配列のグループ分け

基準値6以上の値が見つかるまでplを右へ移動する
（pl++する）

left　　x　　right

0	1	2	3	4	5	6	7	8
5	3	1	4	2	6	7	9	8

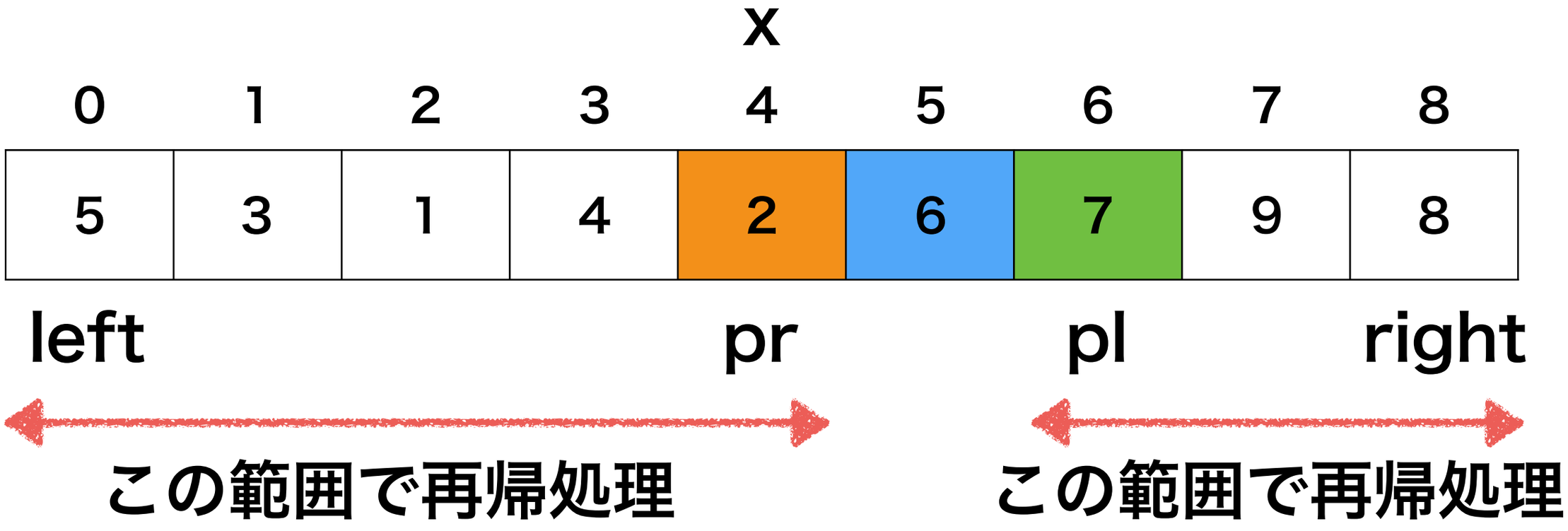
pr　　pl

この処理を「pl > pr」になったので繰り返し終了

- **left～pr番目は、6より小さいグループ**
- **pl～right番目は、6より大きいグループ**

グループ分けの再帰処理

- left～pr番目に対して、再帰処理をする
- pl～right番目に対して、再帰処理をする



<u>範囲の要素数が1になるまで再帰処理を繰り返す</u>

今日の課題は

課題を通して、クイックソートをする
プログラムを少しずつ作っていきます

【準備】

前回の授業で作った関数show(), swap()を今回のプログラムにコピーして下さい。

【課題2-1】

引数で与えられたint型配列の値をクイックソートする関数quick()を作成して下さい。

[この関数のプロトタイプ宣言]

```
void quick(int value[], int left, int right);

/* 配列valueのleft番目からright番目に対してグループ分けをする */
/* 先に示したイラストを参考に作る */
```

【課題2-1】①

先に示したイラストを参考にして、
変数x, pl, prの宣言と初期値を代入して下さい。
（全てint型）

```
void quick(int value[], int left, int right)
{
   //課題①の処理

}
```

【課題2-1】②

先に示したイラストを参考にして、
以下のwhile文の中に、「基準値以上の値が見つかるまで（つまり、基準値未満の値である間）plを右へ移動する（pl++する）」処理を追加して下さい。
（while文で作れます）

```
void quick(int value[], int left, int right)
{
   //課題①の処理
   while(1) { //グループ分けのための繰り返し処理
      //課題②の処理
   }
}
```

【課題2-1】③

先に示したイラストを参考にして、
課題②と同様に「基準値以下の値が見つかるまで（つまり、基準値より大きい値である間）prを左へ移動する（pr—する）」処理を追加して下さい。
（while文で作れます）

```
void quick(int value[], int left, int right)
{
  //課題①の処理
  while(1) { //グループ分けのための繰り返し処理
    //課題②の処理
    //課題③の処理
  }
}
```

【課題2-1】④

先に示したイラストを参考にして、
「pl > prになったら繰り返しを終了する（breakする）」処理を追加して下さい。

```
void quick(int value[], int left, int right)
{
   //課題①の処理
   while(1) { //グループ分けのための繰り返し処理
      //課題②の処理
      //課題③の処理
      //課題④の処理（whileを終了する、つまりグループ分けを終了する）
   }
}
```

【課題2-1】⑤

「value[pl]とvalue[pr]の値を交換する」処理を追加して下さい。前回の関数swap()が使えます。

```
void quick(int value[], int left, int right)
{
  //課題①の処理
  while(1) { //グループ分けのための繰り返し処理
    //課題②の処理
    //課題③の処理
    //課題④の処理（whileを終了する、つまりグループ分けを終了する）
    if(pl <= pr) {   //グループ分け続けるための条件
        //課題⑤の処理
        pl++; pr--;   //交換後、plとprを移動してグループ分けを続行
    }
  }
}
```

【課題2-1】⑥

left～pr番目に対する再帰呼出しと、

pl～right番目に対する再帰呼出しを追加して下さい。

```
void quick(int value[], int left, int right)
{
  //課題①の処理
  while(1) { //グループ分けのための繰り返し処理
    //課題②の処理
    //課題③の処理
    //課題④の処理（whileを終了する、つまりグループ分けを終了する）
    if(pl <= pr) {   //グループ分け続けるための条件
        //課題⑤の処理
        pl++; pr--;   //交換後、plとprを移動してグループ分けを続行
    }
    //課題⑥の処理
  }
}
```

【課題2-1】

[mainの処理]

```
  int v[10] = {9, 7, 6, 0, 1, 5, 2, 3, 4, 8};
  show(v, 10);
  quick(v, 0, 9);
  show(v, 10);
```

[実行結果]

```
9  7  6  0  1  5  2  3  4  8          (←ソート前の出力)
0  1  2  3  4  5  6  7  8  9          (←ソート後の出力)
```