プログラミング応用
http://bit.ly/kosen02

Week14

前期期末試験

- 試験日時：8/4 13:00-14:30(60分退出可)
- 試験範囲
  - 再帰的アルゴリズム
  - 探索アルゴリズム
    - 線形探索/番兵法/二分探索
  - ソートアルゴリズム
    - バブルソート/単純挿入ソート/クイックソート

本日の内容

- クイックソート
  - クイックソートの動作
  - C言語での実現方法
- 演習
  - クイックソートの実装
  - バブルソート、単純挿入ソート、クイックソートの速度比較

クイックソート(概要)

- 高速(Quick)なアルゴリズムとしてよく知られている
- アルゴリズムの基本方針
  1. 基準になるデータを決める
  2. 基準データよりも大きなグループ、小さなグループに分ける
  3. 1. 2. を繰り返す(再帰を使う)

クイックソート(動作例)

- 例) 8人の身長を昇順に並べ替える例を考える

175	170	160	168	165	170	155	150

クイックソート(動作例)

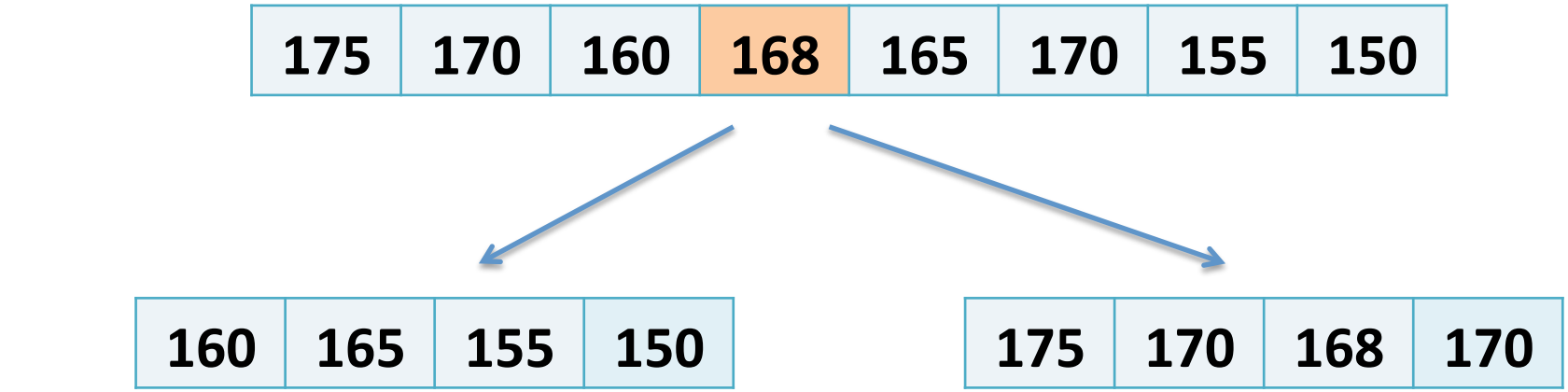
- 例) 8人の身長を昇順に並べ替える例を考える

175	170	160	168	165	170	155	150

1. 基準点を決める
(ここでは配列の中央)

クイックソート(動作例)

- 例) 8人の身長を昇順に並べ替える例を考える

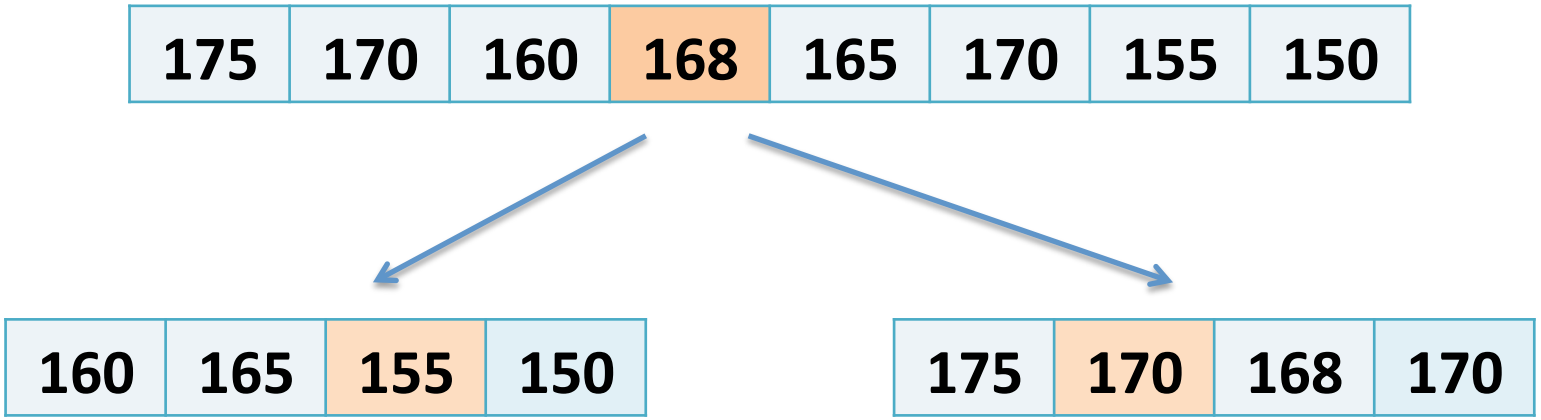


2. 基準点よりも小さなグループ
(ここでは168以下)

2. 基準点よりも大きなグループ
(ここでは168以上)

クイックソート(動作例)

- 例) 8人の身長を昇順に並べ替える例を考える

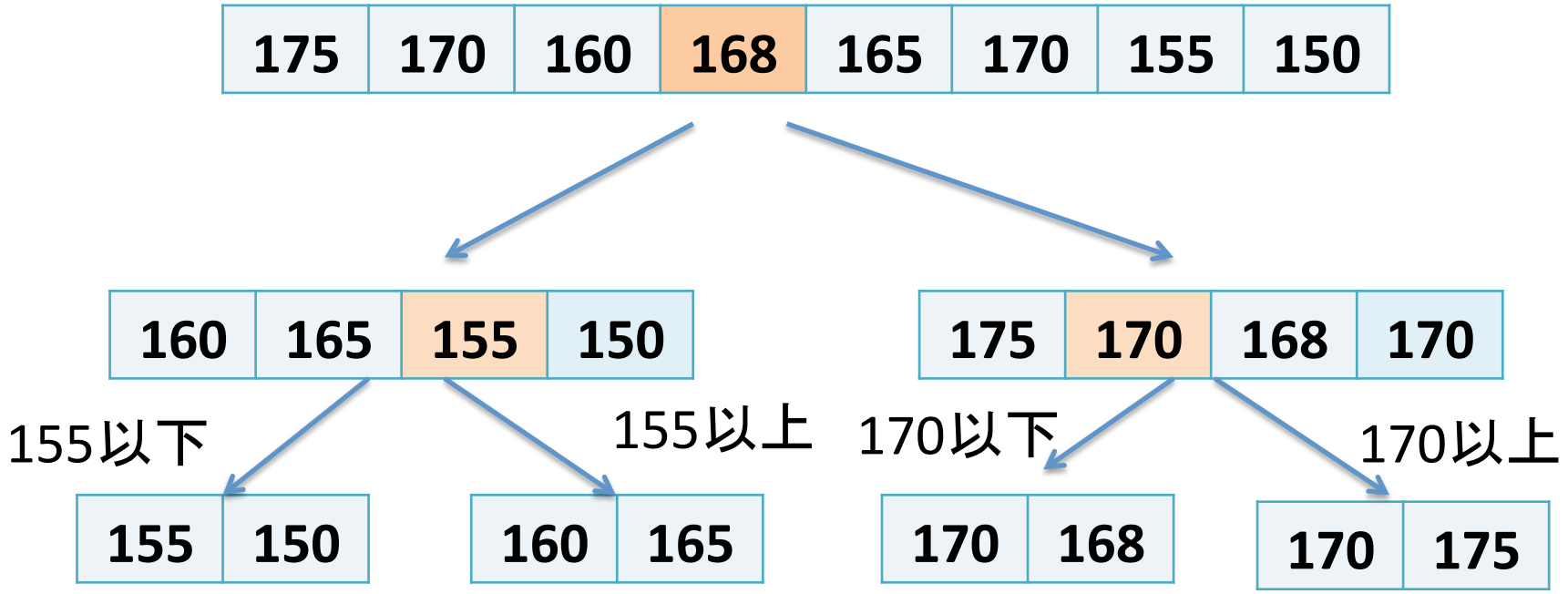


1. 基準点を決める
（配列の中央）

1. 基準点を決める
（配列の中央）

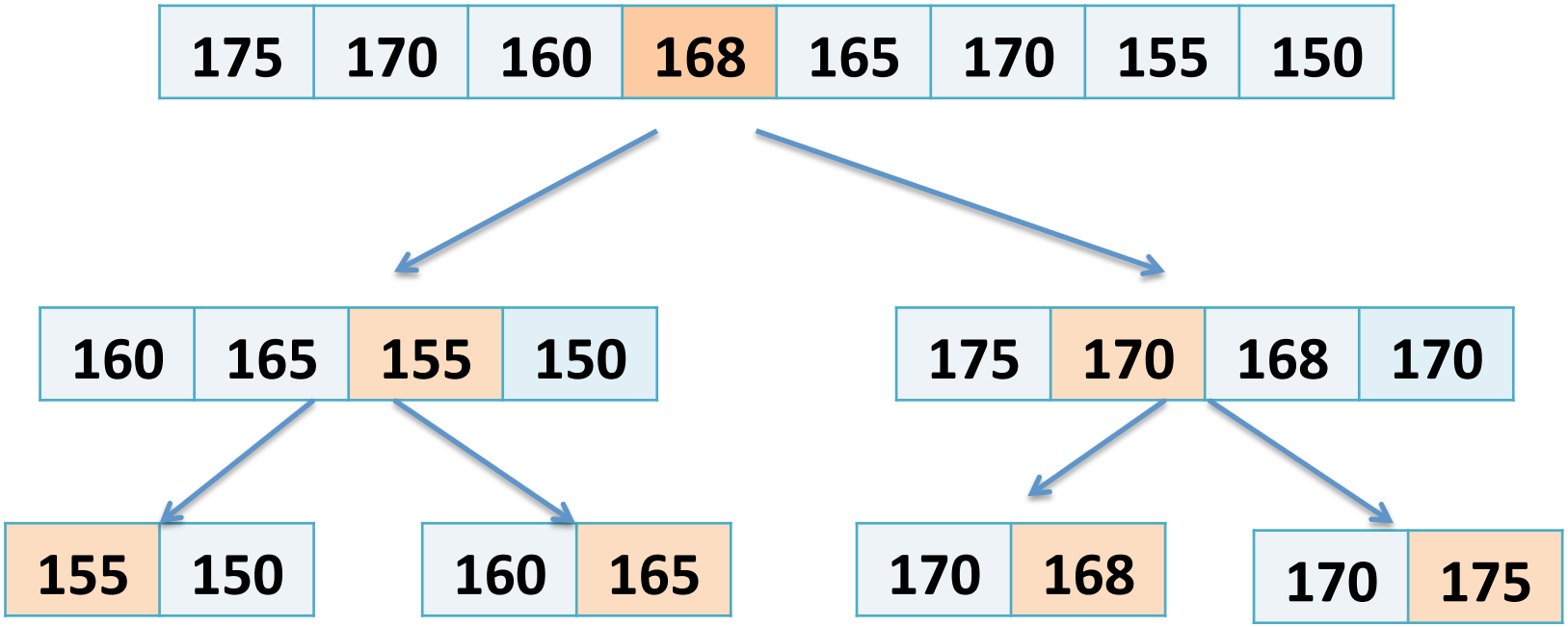
クイックソート(動作例)

- 例) 8人の身長を昇順に並べ替える例を考える



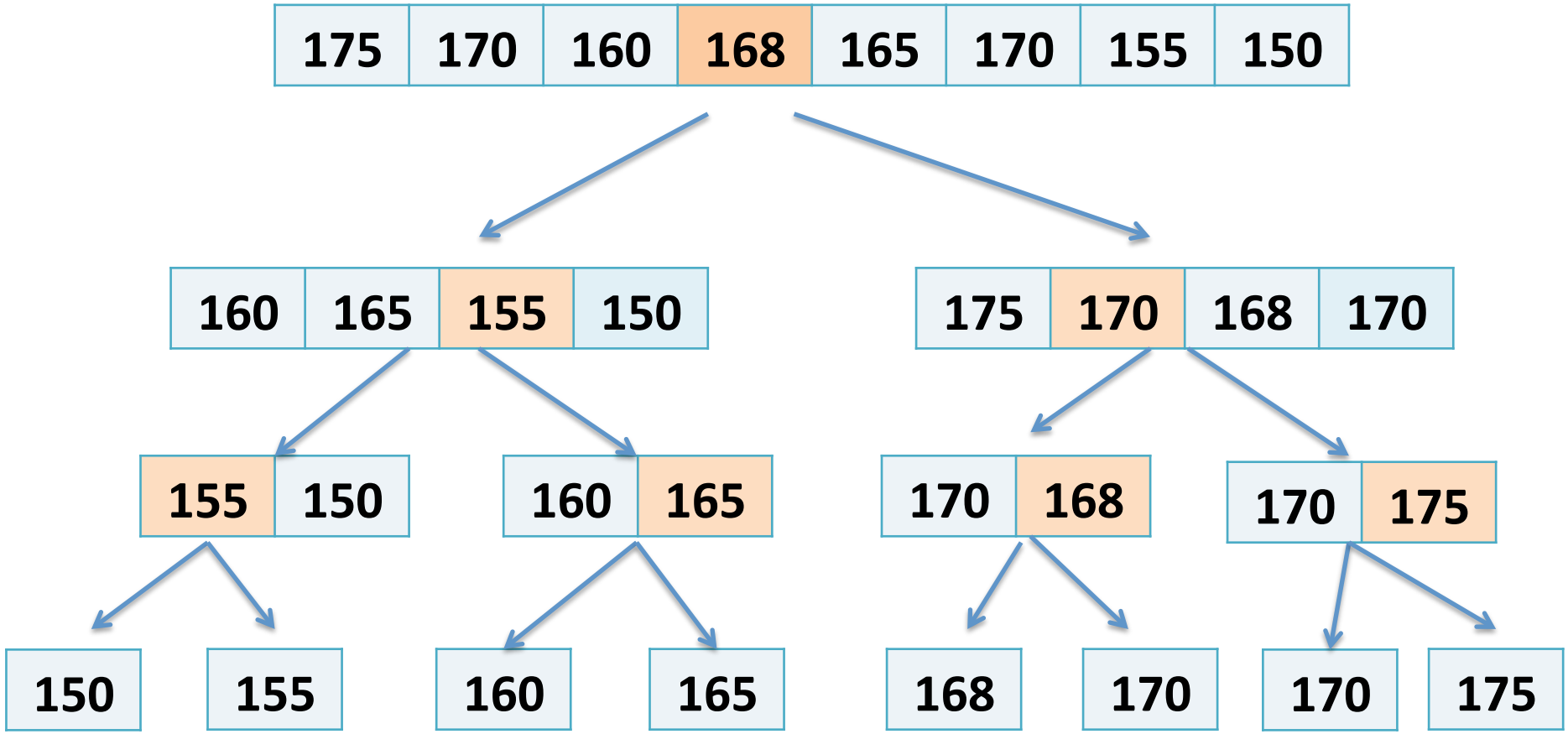
クイックソート(動作例)

- 例) 8人の身長を昇順に並べ替える例を考える



クイックソート(動作例)

- 例) 8人の身長を昇順に並べ替える例を考える



再掲：クイックソート(概要)

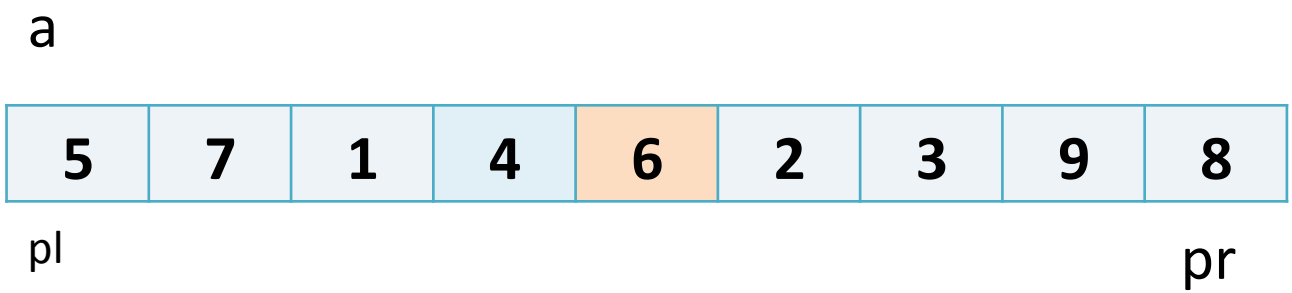
- 高速(Quick)なアルゴリズムとしてよく知られている
- アルゴリズムの基本方針
  1. 基準になるデータを決める
  2. 基準データよりも大きなグループ、小さなグループに分ける
  3. 1. 2. を繰り返す(再帰を使う)

再掲：クイックソート(概要)

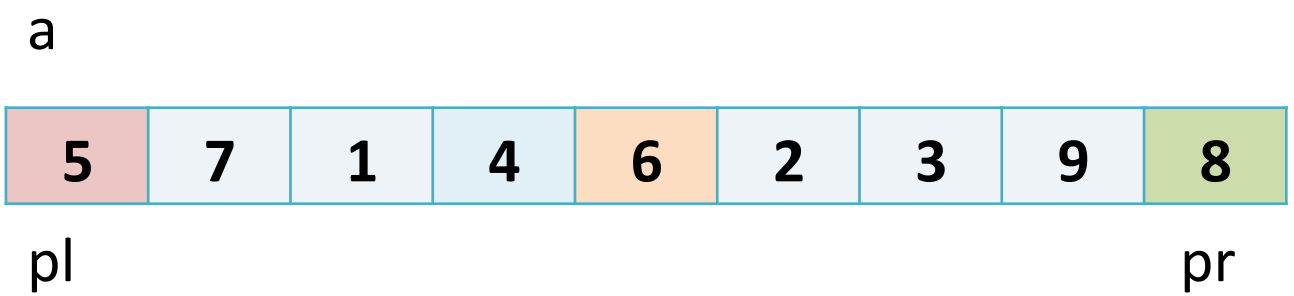
- 高速(Quick)なアルゴリズムとしてよく知られている
- アルゴリズムの基本方針
  1. 基準になるデータを決める
     → 配列の中央の値(a[n/2])で固定
  2. 基準データよりも大きなグループ、小さなグループに分ける
     → 配列を分割する必要あり
  3. 1. 2. を繰り返す(再帰を使う)

配列の分割

- 例）配列aを2つのグループに分割する例を考える
  - 基準点は配列の中央に設定
    x = a[n / 2];
    x: 基準点の値、n: 配列の要素数
  - 6以下の数を左側、6以上の数を右側に移動したい

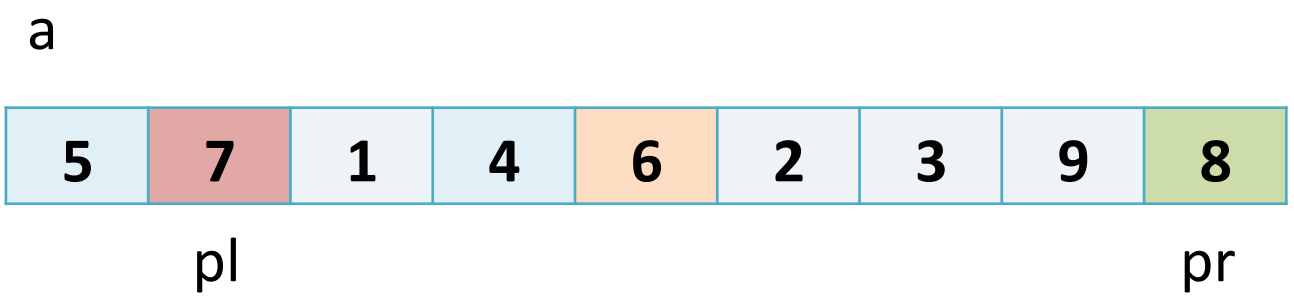


配列の分割(動作例)



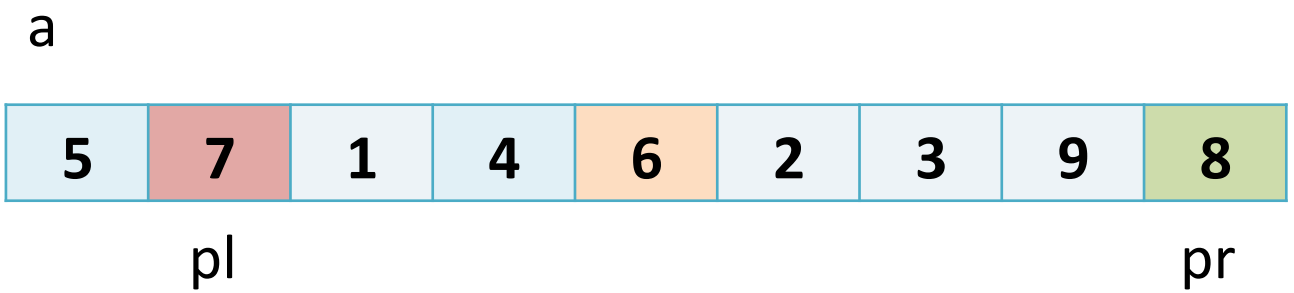
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）

配列の分割(動作例)



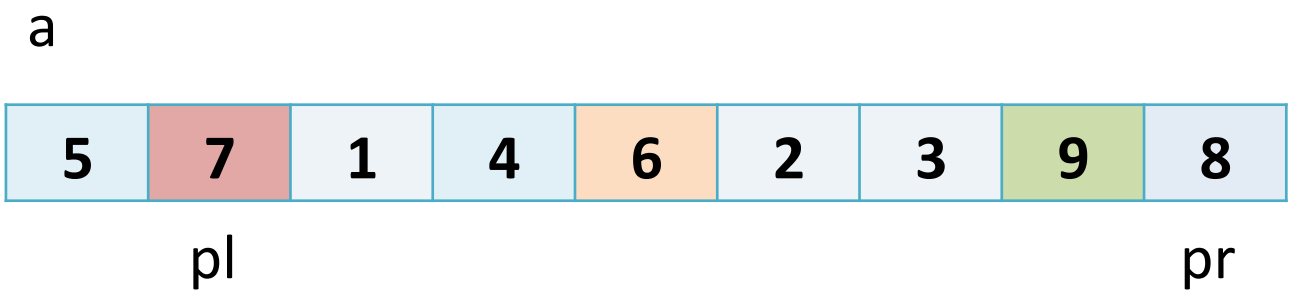
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 1

配列の分割(動作例)



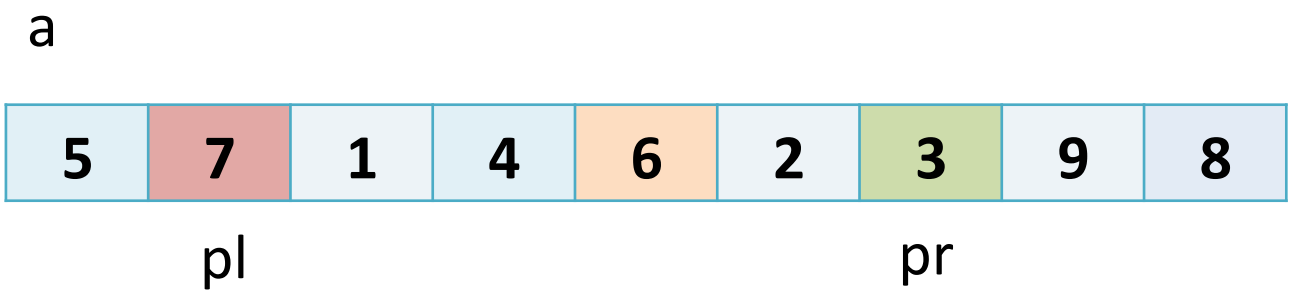
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 2
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）

配列の分割(動作例)



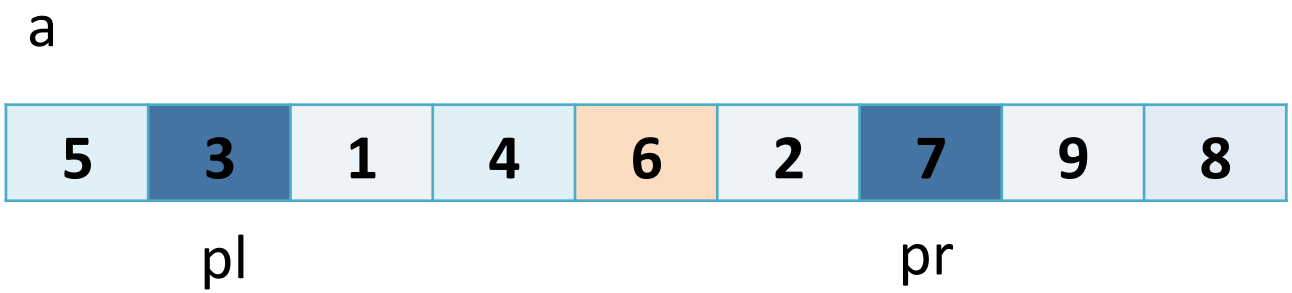
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 2
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）

配列の分割(動作例)



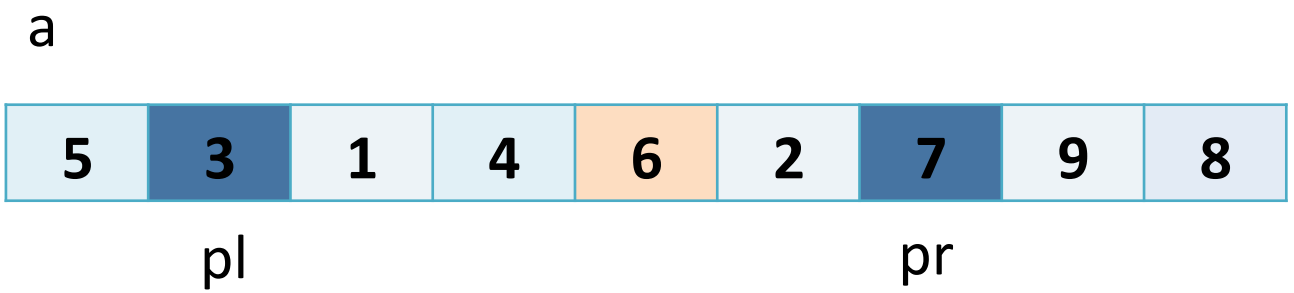
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 2
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
   → 3を発見したのでpr == 6

配列の分割(動作例)



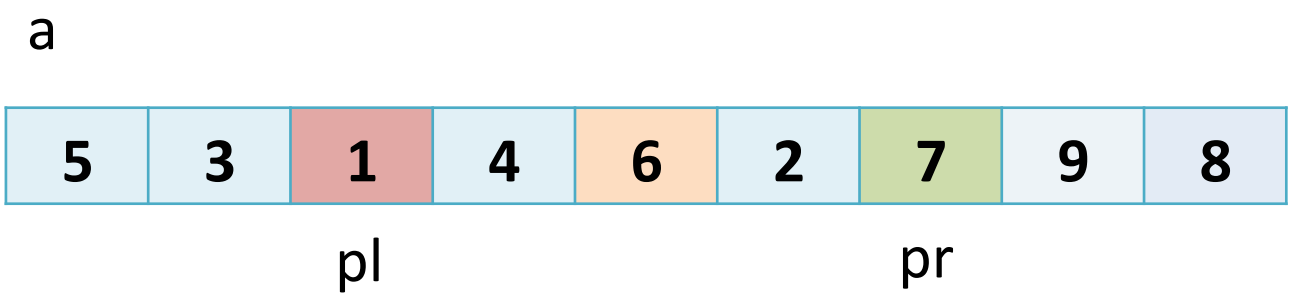
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 2
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
   → 3を発見したのでpr ==6
3. a[pl]とa[pr]の値を入れ替える

配列の分割(動作例)



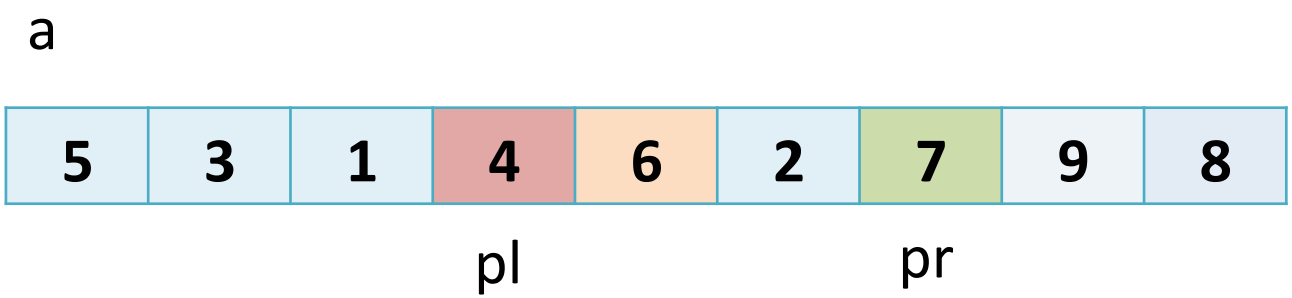
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 2
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
   → 3を発見したのでpr ==6
3. a[pl]とa[pr]の値を入れ替える → pl > prになるまで繰り返す

配列の分割(動作例)



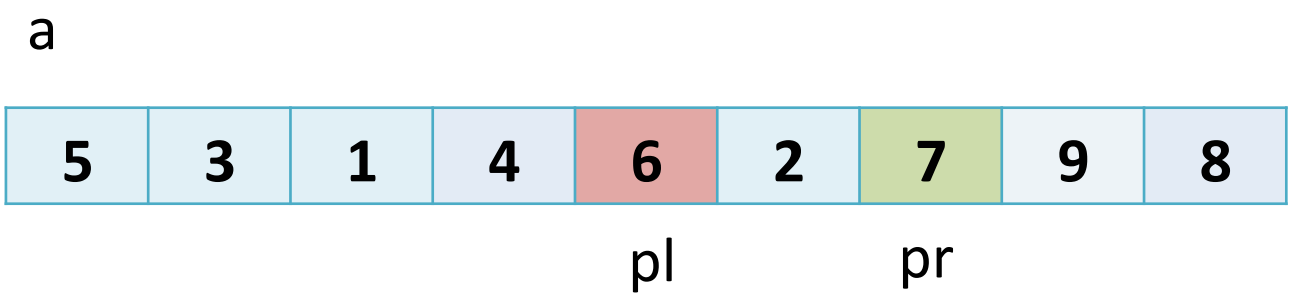
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
3. a[pl]とa[pr]の値を入れ替える　→ pl > prになるまで繰り返す

配列の分割(動作例)



0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
3. a[pl]とa[pr]の値を入れ替える　→ pl == prになるまで繰り返す

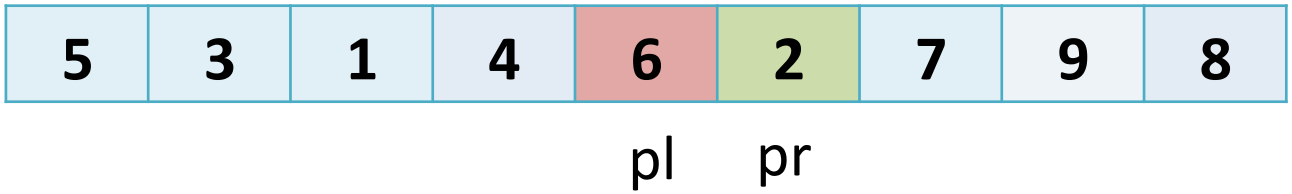
配列の分割(動作例)



0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 6を発見
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
3. a[pl]とa[pr]の値を入れ替える　→ pl == prになるまで繰り返す

配列の分割(動作例)

a



0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 6を発見したのでpl == 24
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
   →　2を発見
3. a[pl]とa[pr]の値を入れ替える　→ pl <= = prになるまで繰り返す

配列の分割(動作例)

a

0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 2
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
   → 3を発見したのでpr ==6
3. a[pl]とa[pr]の値を入れ替える → pl <= = prになるまで繰り返す

配列の分割(動作例)

a

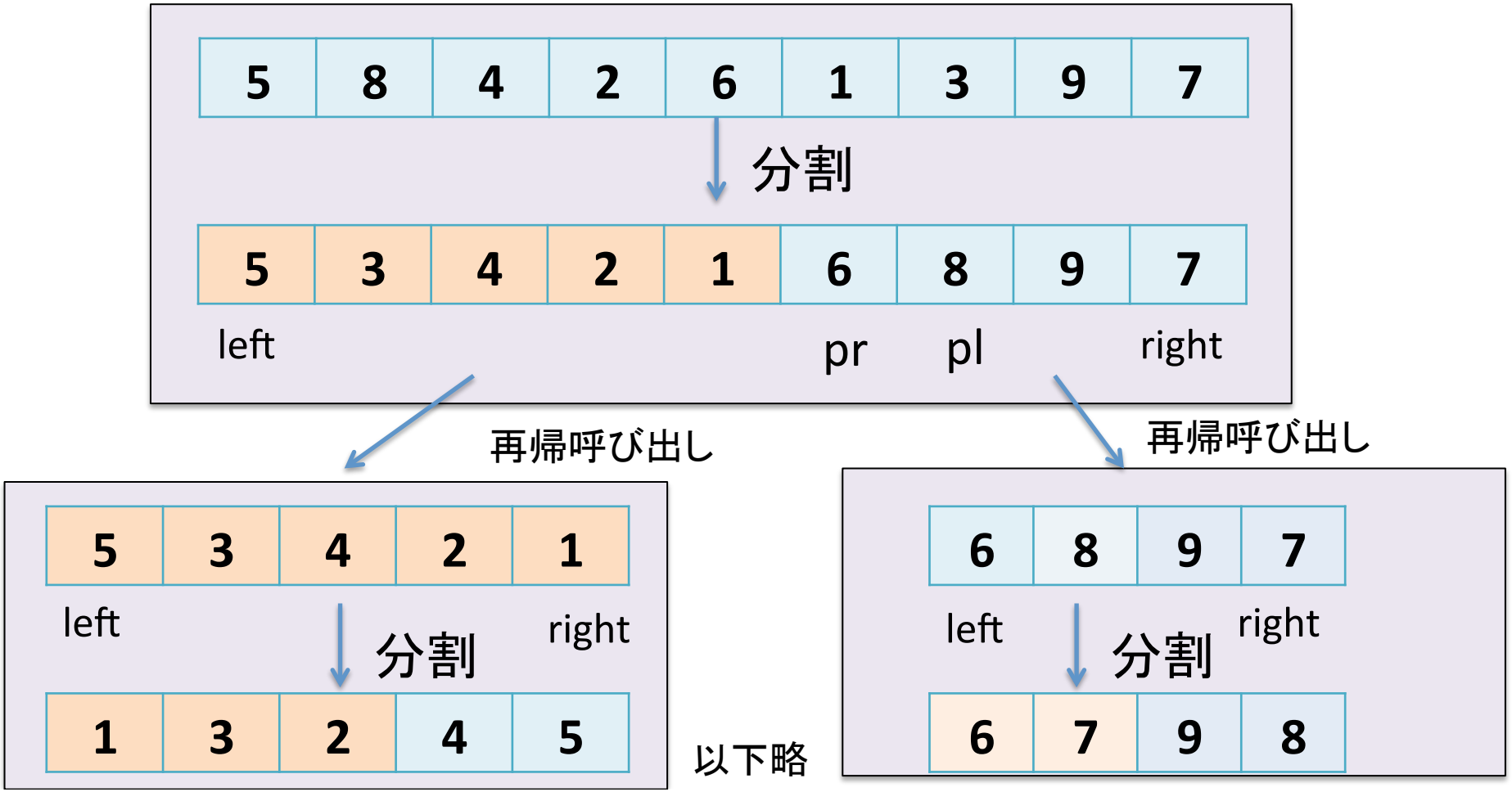
0. plを左端、prを右端にセット
1. 基準点(=6)以上の数が見つかるまでplを右に移動していく
   （pl++していく）
   → 7を発見したのでpl == 2
2. 基準点(=6)以下の数が見つかるまでprを左に移動していく
   （pr--していく）
   → 3を発見したのでpr ==6
3. a[pl]とa[pr]の値を入れ替える → pl < prになるまで繰り返す

配列の分割(まとめ)

- 配列の分割方法
  1. a[pl] >= xが見つかるまでpl++
  2. a[pr] <= xが見つかるまでpr—
  3. 見つかればa[pl]とa[pr]の値を交換
  4. 1. 2. 3. をpl > prになるまで繰り返す

クイックソートの実装

- 要素数が1になるまで分割を繰り返す



演習1

- 次項のプログラムをquick.cというファイル名で保存しクイックソートのアルゴリズムを理解しなさい

- 補足 : 22/23行目if文の条件式
  - left < pr : 分割左部の要素数が1でないか判定
  - pl < right : 分割右部の要素数が1でないか判定

```c
#include <stdio.h>

#define DATASIZE 10
#define swap(type, x, y) do { type t = x; x = y; y = t; } while (0)

/* クイックソート */
void quick(int a[], int left, int right) {
  int pl = left;
  int pr = right;
  int x = a[(pl+pr)/2];
  // 1. 配列の分割
  do {
    while (a[pl] < x) pl++;
    while (a[pr] > x) pr--;
    if (pl <= pr) {
      swap(int, a[pl], a[pr]);
      pl++;
      pr--;
    }
  } while (pl <= pr);
  // 2. 再帰呼び出し
  if (left < pr) quick(a, left, pr);
  if (pl < right) quick(a, pl, right);
}

int main() {
  int x[DATASIZE] = {9, 5, 8, 4, 2, 6, 1, 3, 9, 7};
  quick(x, 0, DATASIZE-1);
  for(int i=0; i < DATASIZE; i++) {
    printf("x[%d]: %d\n", i, x[i]);
  }
  return 0;
}
```

演習2

- 第12週演習5を参考に、ファイルからデータを読み込みクイックソートするプログラム(quick_file.c)を作成せよ
  - データファイルは以下のコマンドで入手可能
    cp ~ishigaki/sample4binary .
  - ヒント:
    quick.cのmain()関数を書き換えることで作成する

参考：第12週演習5
(ファイルからの読み出し例)

```
int i, data_size = 0;
int a[50000];    /* 大きめのサイズの配列を用意しておく．  */
FILE *fp;

fp = fopen("sample4binary.txt", "r");
if(fp == NULL) {
  printf("ファイルを開くことが出来ませんでした．¥n");
  return 0;
}

/*  ファイルが終わりでない　「かつ」
  配列を飛び出さないうちは，読み込みを続ける  */
while ( ! feof(fp) && data_size < 50000) {
  fscanf(fp, "%d", &(a[data_size]));
  data_size++;
}
fclose(fp);
data_size = data_size-1;
/* なお、上のwhileループでは，EOFの行を余分に
読み込んでいるので，実際のデータ数は一つ少ない．  */
```

演習3

- 先週までに作成したバブルソートのプログラムについても演習2と同様にファイル読み込みできるように書き換え、bubble_file.cというファイル名で保存せよ

演習4

- 先週までに作成した単純挿入ソートのプログラムについても演習2と同様にファイル読み込みできるように書き換え、insert_file.cというファイル名で保存せよ

演習5

- 演習4までに作成したxxx_file.cを以下のコマンドで名前付きコンパイルせよ

  $ cc quick_file.c –o quick

  $ cc bubble_file.c –o bubble

  $ cc insert_file.c –o insert

演習6

- timeコマンドで各アルゴリズムの実行時間を測定し、クイックソートが速いことを確認せよ

  クイックソートの計測コマンド
  $ time ./quick
  単純挿入ソートの計測コマンド
  $ time ./insert
  バブルソートの計測コマンド
  $ time ./bubble

時間がある人向け1

- jotコマンドを使うと乱数を生成できます。
  $ jot -r 45000 1000 9999 > sample4binary.txt

  上のコマンドでは1000から9999までのランダムな値を45000個表示し、sample4binary.txtに書き出しています。

  1万個/10万個/100万個の乱数をファイルに書き出し、それぞれのアルゴリズムで並べ替えてみましょう。
  (クイックソートの速さを体感してください)

時間がある人向け2

- 横軸にデータの個数、縦軸に計算にかかる時間をプロットしたグラフを作成してみましょう
- クイックソートの平均計算時間がO(nlogn)になっていることを確認しましょう