

第13回 クイックソート・その他

アルゴリズムとデータ構造ならびに同演習

柏原 昭博

akihiro.kashihara@inf.uec.ac.jp

講義内容

- 第11回目 ヒープ
- 第12回目 ソート（整列）
- 第13回目 クイックソート・その他のソート
- 第14回目 文字列処理（基本）
- 第15回目 文字列処理（配列処理・文字探索）

ソート法の分類

	ソート法	計算量
基本形	基本 交換 法（バブル・ソート）	$O(n^2)$
	基本 選択 法（直接選択法）	
	基本 挿入 法	
改良型	クイック・ソート （改良交換法）	$O(n \log_2 n)$
	ヒープ・ソート （改良選択法）	
	シェル・ソート（改良挿入法）	$O(n^{1.2})$

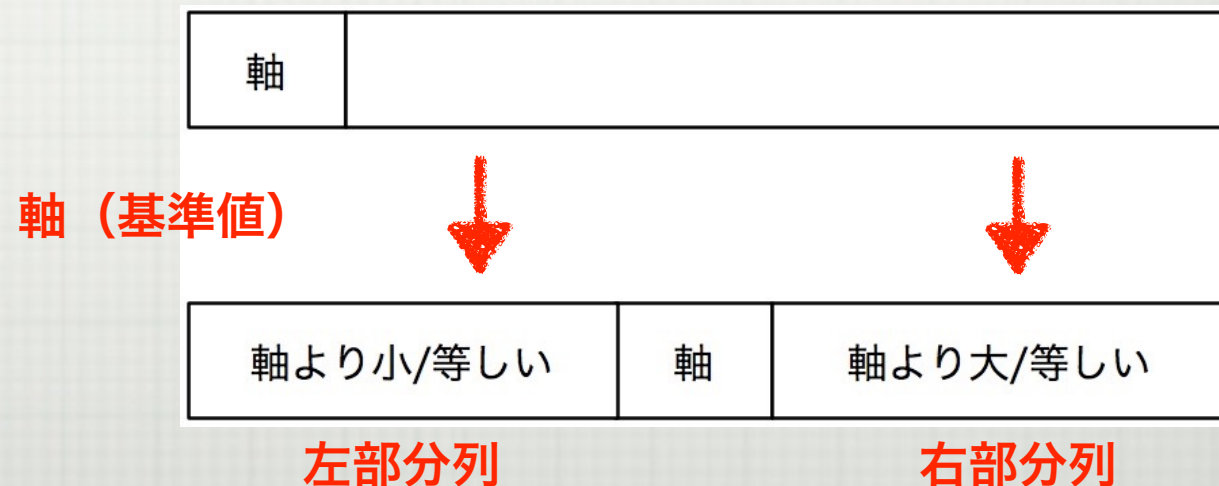
マージソート

クイックソート

📌 基本的な考え方

📌 **軸（基準値）** を決め、その他の要素を軸以下（左部分列）、軸以上（右部分列）に分ける

📌 分けた各部分列に対して、再帰的にそれぞれ軸を決めて2つに分ける



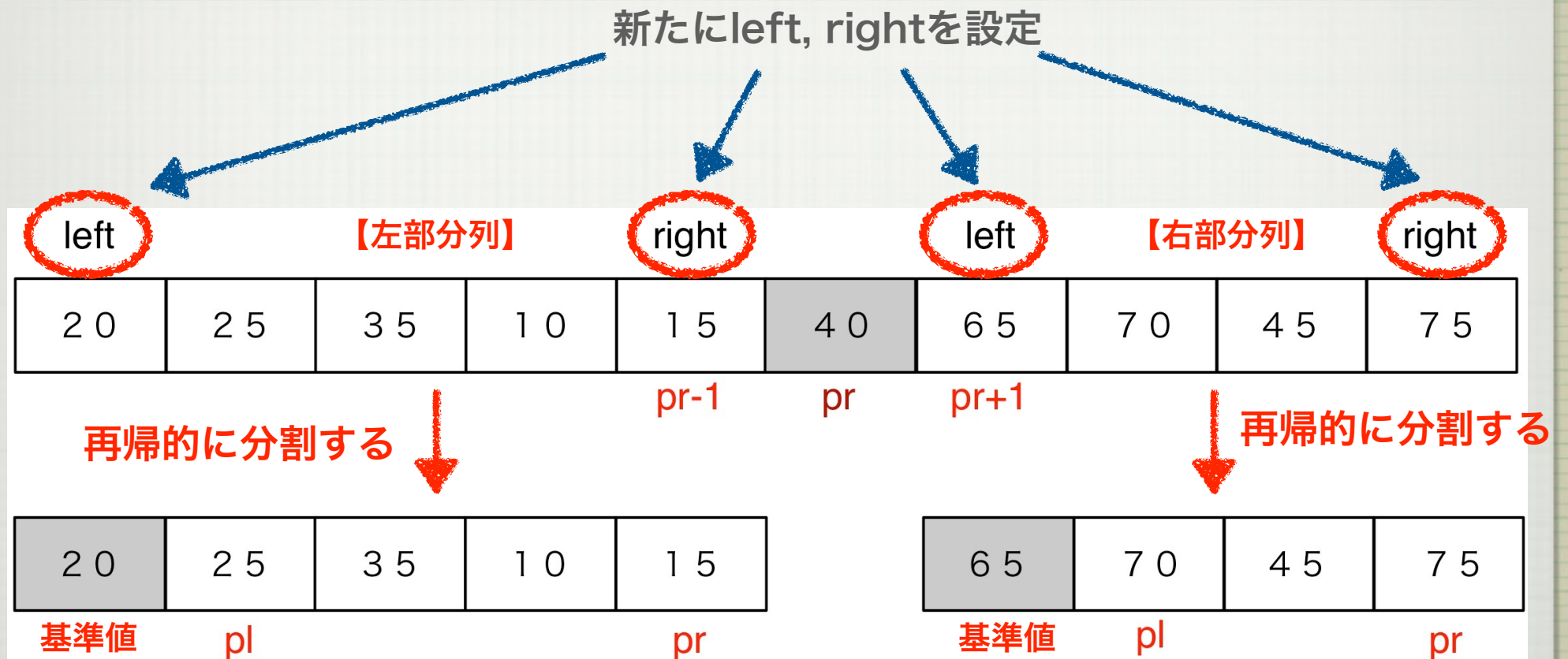
クイックソートの方法：配列の分割

- 軸を決める
 - 配列の先頭要素を軸とする
- 軸以外の要素を**2分割**（軸より小さい部分列， 軸より大きい部分列）
- 部分列ごとに軸を決めて， **再帰的に**部分列を2分割
- 部分列の要素が1つになるまで繰り返す

配列の2分割

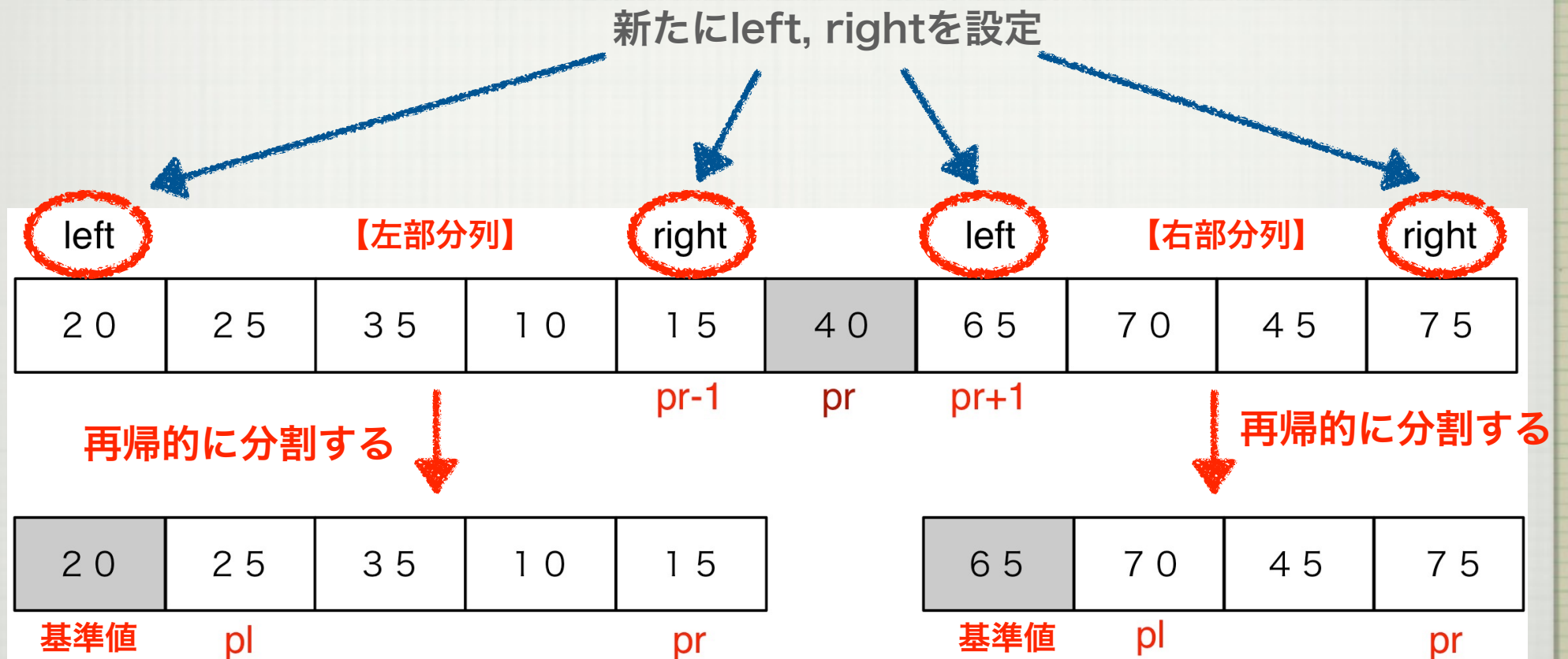


再帰的分割



1. $pl > pr$ となるまで, 要素を探索・交換する.
2. $pl \geq pr$ で, 基準値と $a[pr]$ を交換する.

再帰的分割



再帰的分割の終了 : $\text{left} \geq \text{right}$

left					right				
20	25	35	10	15	40	65	70	45	75

pr-1 pr pr+1

基準値

20	25	35	10	15
pl		pr		

基準値

65	70	45	75
pl		pr	

20	15	10	35	25
pr		pl		

pl > prで終了

65	45	70	75
pr		pl	

基準値とa[pr=2]を交換

基準値とa[pr=1]を交換

10	15	20	35	25	40	45	65	70	75
----	----	----	----	----	----	----	----	----	----

基準値

基準値

基準値

10	15
----	----

35	25
----	----

70	75
----	----

pl pr
基準値とa[0]を交換

pl > prで終了
pl pr
基準値とa[1]を交換

pl > prで終了
pl pr
基準値とa[0]を交換

10	15	20	25	35	40	45	65	70	75
----	----	----	----	----	----	----	----	----	----

演習13-1

- sample13-1.c に、軸をもとに配列を左部分列、右部分列に分割する関数を示す。このプログラムを動作させて、配列の分割方法を理解しなさい。

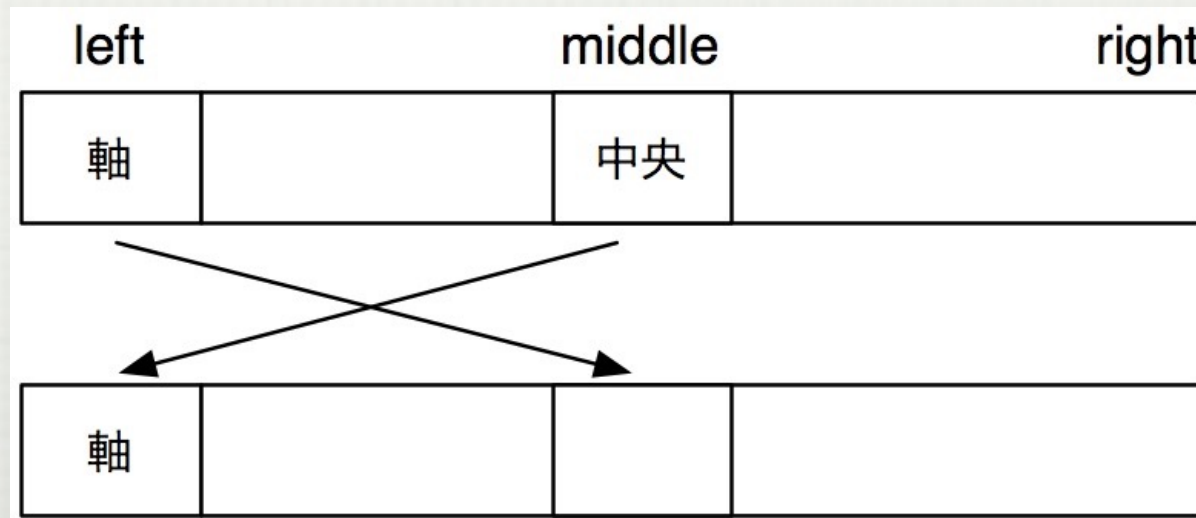
sample13-1.c
(抜粋)

```
void partition (int a[], int left, int right)
{
    int i,tmp;
    int pl, pr;
    int pivot;

    if (left<right) {
        pivot=a[left];//軸の設定
        pl=left; pr=right+1;//左側ポインタ・右側ポインタ初期化
        do {
            pl++; pr--;
            while (a[pl]<pivot) pl++;//左側ポインタの移動
            while (a[pr]>pivot) pr--;//右側ポインタの移動
            if (pl<pr) //値の交換
                tmp=a[pl]; a[pl]=a[pr]; a[pr]=tmp;
        } while (pl<pr);
        a[left]=a[pr]; a[pr]=pivot;//軸と要素の交換
    }
}
```

演習13-2

- sample13-1.c もとに，配列の中央の要素を軸として左部分列，右部分列に分割するプログラム（sample13-2.c）を作成しなさい。
ヒント：あらかじめ配列の最初の要素と中央の要素を交換しておく




```
void partition (int a[], int left, int right)
```

```
{
```

```
    int i,middle,tmp;
```

```
    int pl, pr;
```

```
    int pivot;
```

```
    if (left<right) {
```

```
        middle=(left+right)/2; //中央の要素
```

```
        tmp=a[middle]; a[middle]=a[left]; a[left]=tmp;
```

```
        //先頭要素と中央要素の交換
```

```
        pivot=a[left]; //軸の設定
```

```
        pl=left; pr=right+1; //左側ポインタ・右側ポインタの初期化
```

```
        do {
```

```
            pl++; pr--;
```

```
            while (a[pl]<pivot) pl++; //左側ポインタの移動
```

```
            while (a[pr]>pivot) pr--; //右側ポインタの移動
```

```
            if (pl<pr) { //値の交換
```

```
                tmp=a[pl]; a[pl]=a[pr]; a[pr]=tmp;
```

```
            }
```

```
        } while (pl<pr);
```

```
        a[left]=a[pr]; a[pr]=pivot; //軸と要素の交換
```

```
    }
```

```
}
```

sample13-2.c

(抜粋)

課題13-1

- sample13-2.c もとに，配列の中央の要素を軸として再帰的に配列を左部分列，右部分列に分割し，昇順に整列するプログラム（ex13-1.c）を作成しなさい.

軸も交換対象とするクイックソート：2分割

- これまで軸を左端に置き，交換対象から外してきたが，軸も含めて交換を行い，再帰的に配列を2分割（左部分列，右部分列）する．なお，ここでは軸を配列の中央の要素とする．



軸も交換対象とする
クイックソート：
再帰的分割

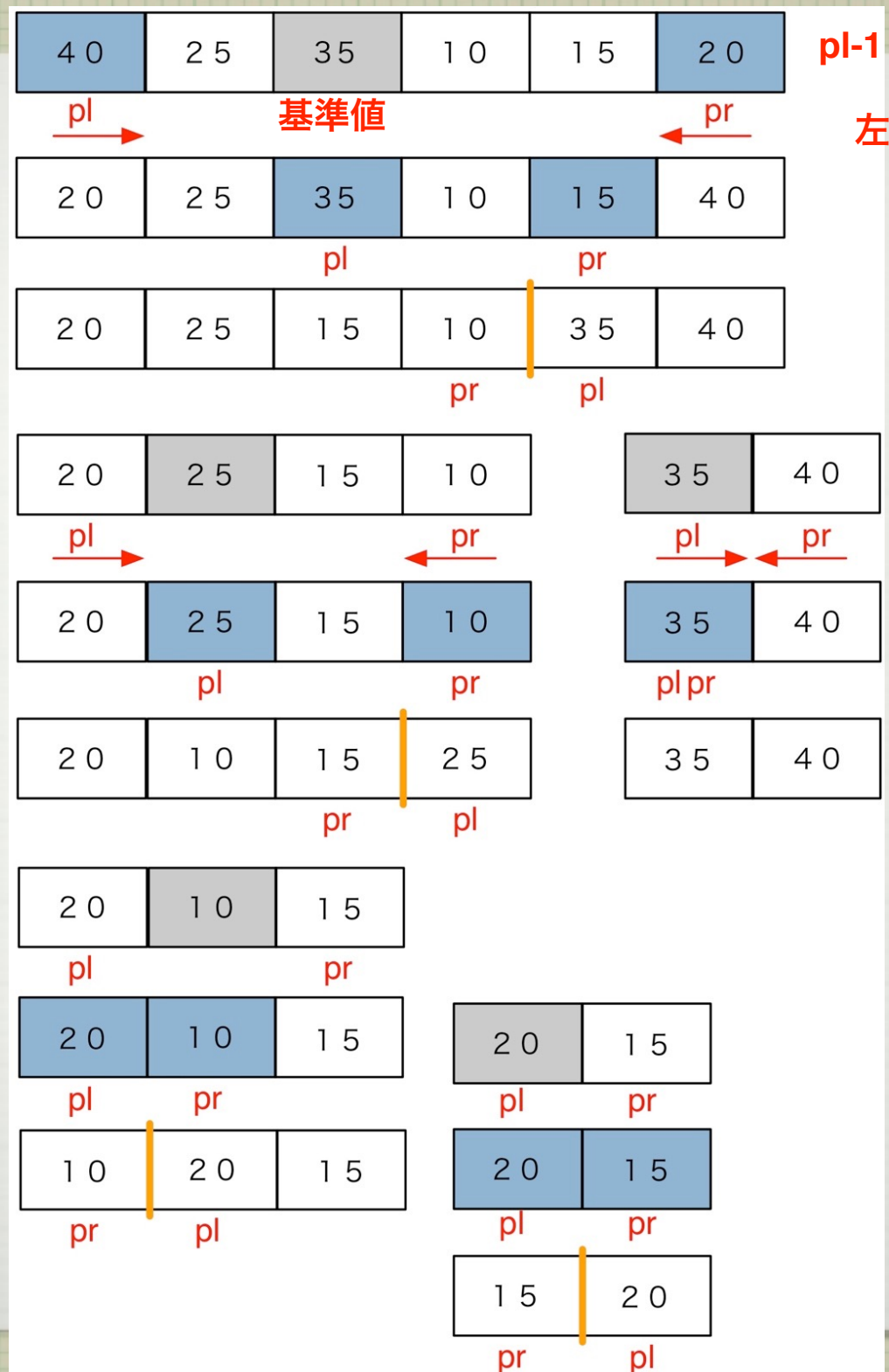
- 左部分列: $a[\text{left}] \sim a[\text{pr}]$
- 右部分列: $a[\text{pr}+1] \sim a[\text{right}]$

に対して再帰的に2分割

部分列ごとに

$\text{left} \geq \text{right}$ で終了

left



課題13-2

- 軸も交換対象として、再帰的に配列を左部分列、右部分列に分割し、要素を整列するプログラム (ex13-2.c) を作成しなさい。なお、軸は中央の要素とする。

クイックソートの計算量

□ 平均計算量

□ 要素数: $n \rightarrow$ 一回の比較回数: $n-1$

□ 比較回数 $Q_n = (n-1) + Q_a + Q_b$ (なお, $a+b=n-1$)

分割 (a,b) の組み合わせ: $(0, n-1), (1, n-2), \dots, (n-1, 0)$

等確率で起こるとすると,

$$Q_n = (n-1) + 1/n \sum_{k=0, n-1} Q_k \quad (n \geq 2)$$

$$Q_0 = 0, Q_1 = 0$$

この漸化式を解くと, $Q_n \doteq 2n \log n \rightarrow O(n \log n)$

□ 最悪計算量


軸が常に最小値 (あるいは最大値) となる場合 (= 基本選択法と同等)

比較回数: $(n-1) + (n-2) + \dots + 2 + 1 = n(n-1)/2 \rightarrow O(n^2)$

最悪のケース

軸が常に要素列の最小値（あるいは最大値）の場合
：一方の部分列が空になる

1 0	1 5	2 0	2 5	3 5
-----	-----	-----	-----	-----



軸が要素列の中央値の場合に最善

2 0	1 5	1 0	2 5	3 5
-----	-----	-----	-----	-----

つまり、同じ大きさの部分列に 2 分割：
分割回数は $\log_2 n$

クイックソートの改善

- 軸の選び方
 - 左端の要素を選択：部分列に偏りが生じる可能性あり
 - 中央値の計算：計算量が増す
 - **いくつかのサンプルから中央値を選ぶ**
- 再帰の除去
 - 再帰呼び出しせずに繰り返し型でプログラムを作成する
- 挿入法の併用
 - 要素数がある値より小さい場合に用いる

課題13-3

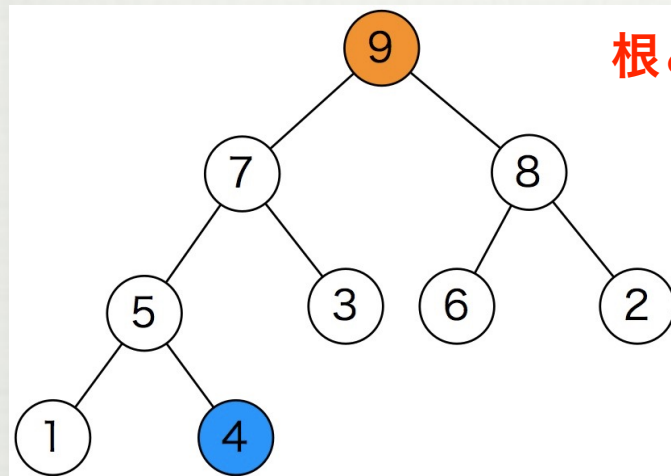
- 配列の先頭要素, 中央要素, 末尾要素の3つのうち中央値を持つ要素を軸として左部分列, 右部分列に分割するクイックソートのプログラムを作成しなさい. (ex13-3.c)

ヒープソート

- 11回目で、完全2分木からヒープを構成する方法、ヒープから根を取り除いて再構成する方法を学んだ。これを応用して要素を整列する方法がヒープソートである。
- 根を削除する代わりに根とヒープの最後の要素を交換し、ヒープを再構成することで整列を行うことができる。

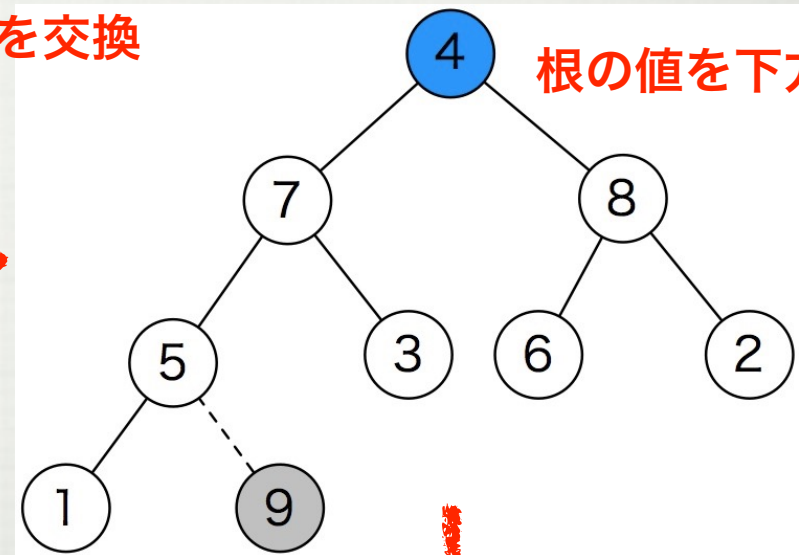
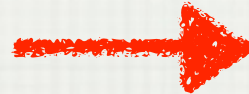
手順

- 根（最大要素）と最後の要素を交換して，ヒープを再構成
- これを繰り返せば，要素を整列（ソート）できる



ヒープの最後の要素を根へ移動

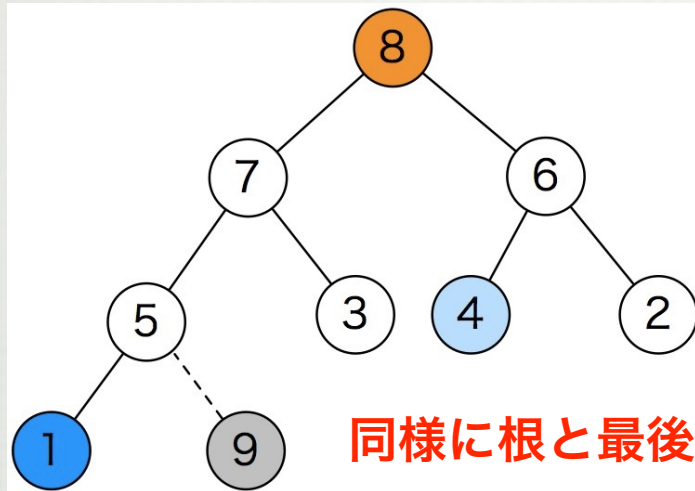
根と最後の要素を交換



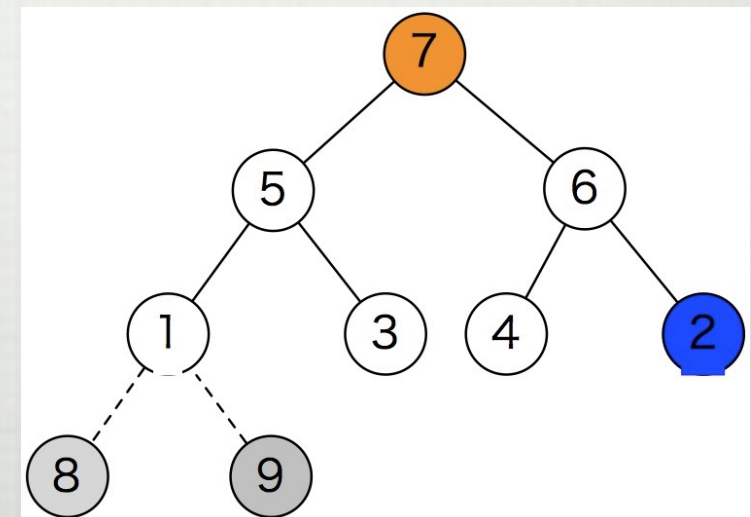
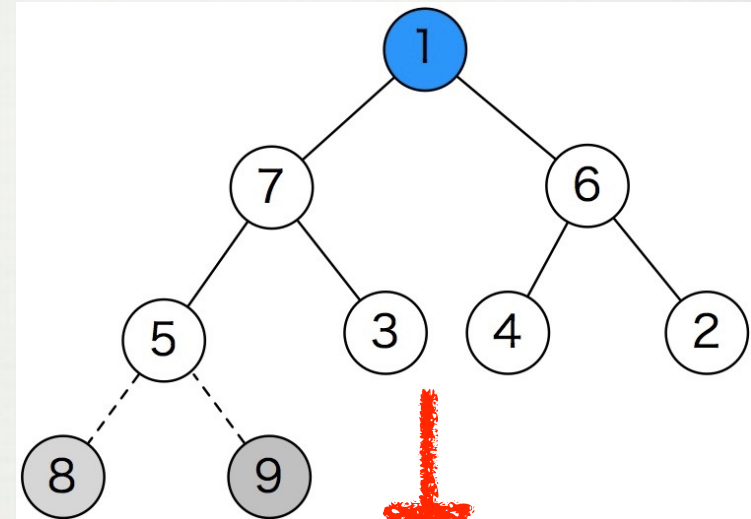
根の値を下方移動



ヒープから要素の整列



同様に根と最後の要素の交換



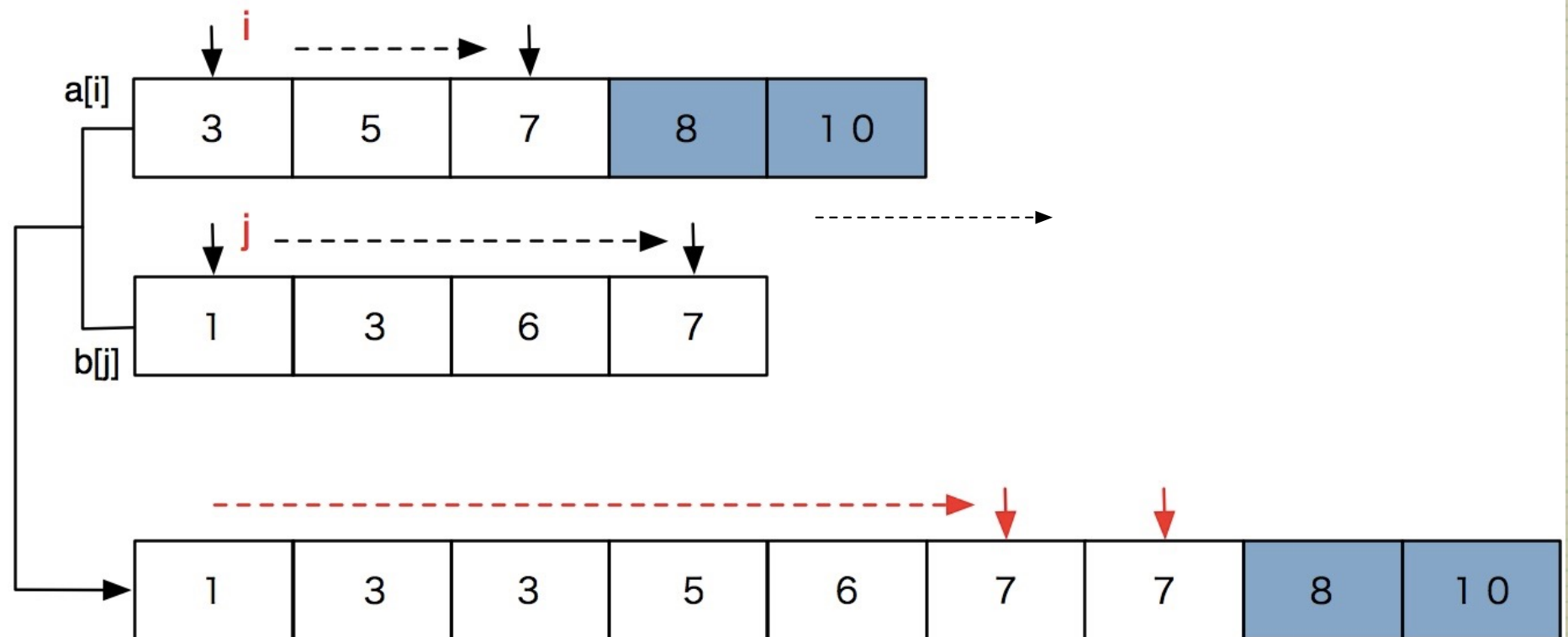
最後の要素が根となるまで繰り返す

演習13-3

- sample11-5.cを参考にして、完全2分木からヒープを構成し、ヒープソートを実行するプログラムを作成しなさい。 (sample13-3.c)

マージソート

□ ソート済み配列のマージ（併合）



$(a[i] \leq b[j]) ? a[i] : b[j]$

演習13-4

- sample13-4.cに、ソート済みの二つの配列をマージする関数を示す。配列を色々変えてみてプログラムを動作させ、マージソートの動作を理解しなさい。

```
#include <stdio.h>
```

```
void merge(int *, int, int *, int, int *);
```

```
int main (void)
```

```
{
```

```
    int a[5]={3,5,7,8,10}; //整列済み配列aの設定
```

```
    int an=sizeof(a)/sizeof(a[0]); //aの要素数
```

```
    int b[4]={1,3,6,15}; //整列済み配列bの設定
```

```
    int bn=sizeof(b)/sizeof(b[0]); //bの要素数
```

```
    int i;
```

```
    int c[an+bn]; //マージ後の配列
```

```
    merge(a, an, b, bn, c); //マージ
```

```
}
```

```
void merge (int a[], int an, int b[], int bn, int c[])
```

```
{
```

```
    int pa = 0; int pb = 0; int pc = 0;
```

```
    while (pa < an && pb < bn) //aとbのマージ
```

```
        c[pc++] = (a[pa] <= b[pb]) ? a[pa++] : b[pb++];
```

//aの要素がbよりも小さいか等しいときにaの要素をcへ出力

```
    while (pa < an) // 比較後, 配列aに要素が余っている場合はcへ出力
```

```
        c[pc++] = a[pa++];
```

```
    while (pb < bn) //比較後, 配列bに要素が余っている場合はcへ出力
```

```
        c[pc++] = b[pb++];
```

```
}
```

sample13-4.c

(抜粋)

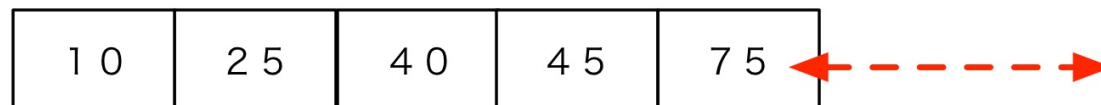
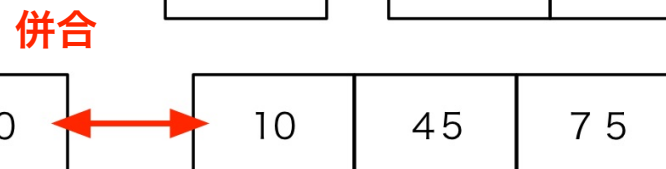
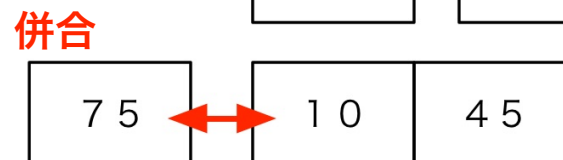
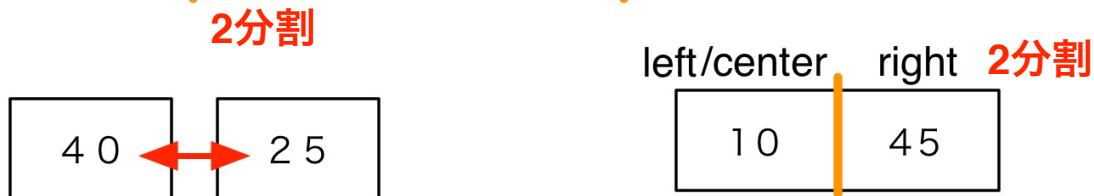
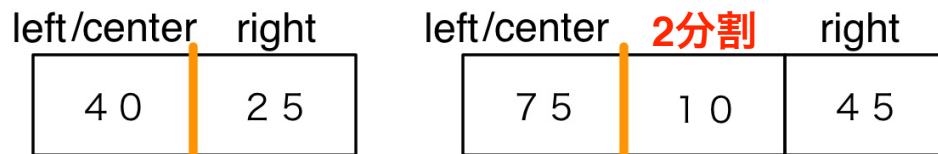
配列2分割によるマージソート

- 一つの配列を中央から再帰的に2分割し、それぞれの部分列をソートする

練習問題

- 次の配列を二分割しながら昇順にマージソートしてみよう

4 0	2 5	7 5	1 0	4 5	6 5	2 0	7 0	1 5	3 5
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----



分割回数：1 + 4 + 4
併合回数：4 + 4 + 1

マージソート

```
merge_sort (a[], left, right) {
```

```
    left < rightの間
```

```
        center: left とrightの中央値
```

```
        merge_sort (a, left, center);
```

```
        merge_sort (a, center+1, right);
```

```
        Merge (a, left, mid, right)
```

```
            …a[left~mid]とa[mid+1~right]を併合
```

計算量：Mergeには $O(n)$ ※ n を2のべき乗とする.

Aを2分割してMergeにかかる計算量： $O(n)/2 + O(n)/2 = O(n)$

さらに2分割してかかる計算量： $O(n)/4 + \dots + O(n)/4 = O(n)$

配列要素が1となるまでの2分割回数は, $\log n$ 回

全体の計算量は, おおよそ $O(n) \times \log n$. よって **$O(n \log n)$**

課題13-4

- 一つの配列を中央から二つ分割し、それぞれをソートした後にマージするプログラムを作りなさい。なお、各部分のソートにもマージソートを用いること。（ex13-4.c）
ヒント：配列の分割を再帰的に定義すると良い。

```
void merge_sort (int a[], int left, int right) {  
    left < rightの間  
        merge_sort (a, left, center);  
        merge_sort (a, center+1, right);  
        a[left~center]とa[center+1~right)とを比較・併合  
}
```

レポート提出

- 課題13-1～13-4
- 提出期限：7月22日 0:00（7月21日まで）
- 提出先：Webclass