アルゴリズムとデータ構造

第5回　高速なソート

第5回のキーワード

アルゴリズム関係

- クイックソート（quicksort）
- ピボット（軸）（pivot）
- 分割統治（divide and conquer）
- マージソート（merge sort）
- $O(n \log n)$

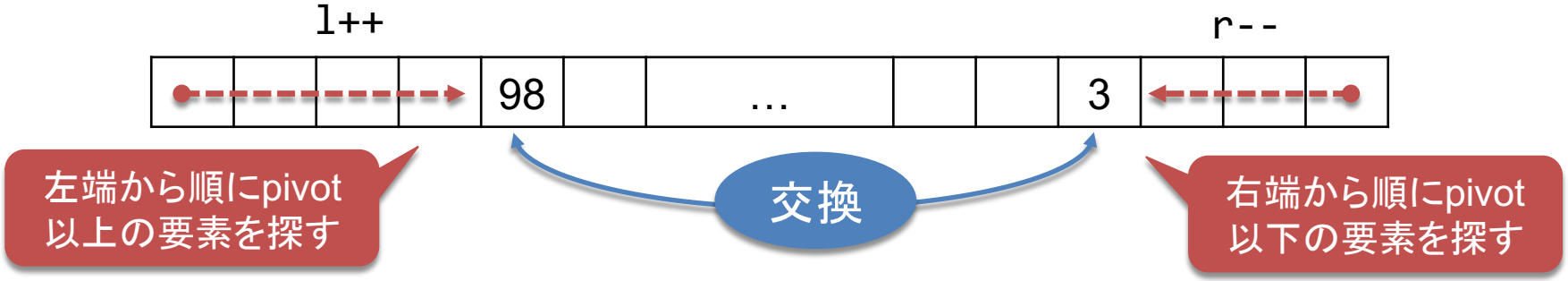
Java関係

- 無限ループ
  - while(true)
  - for(;;)

クイックソート

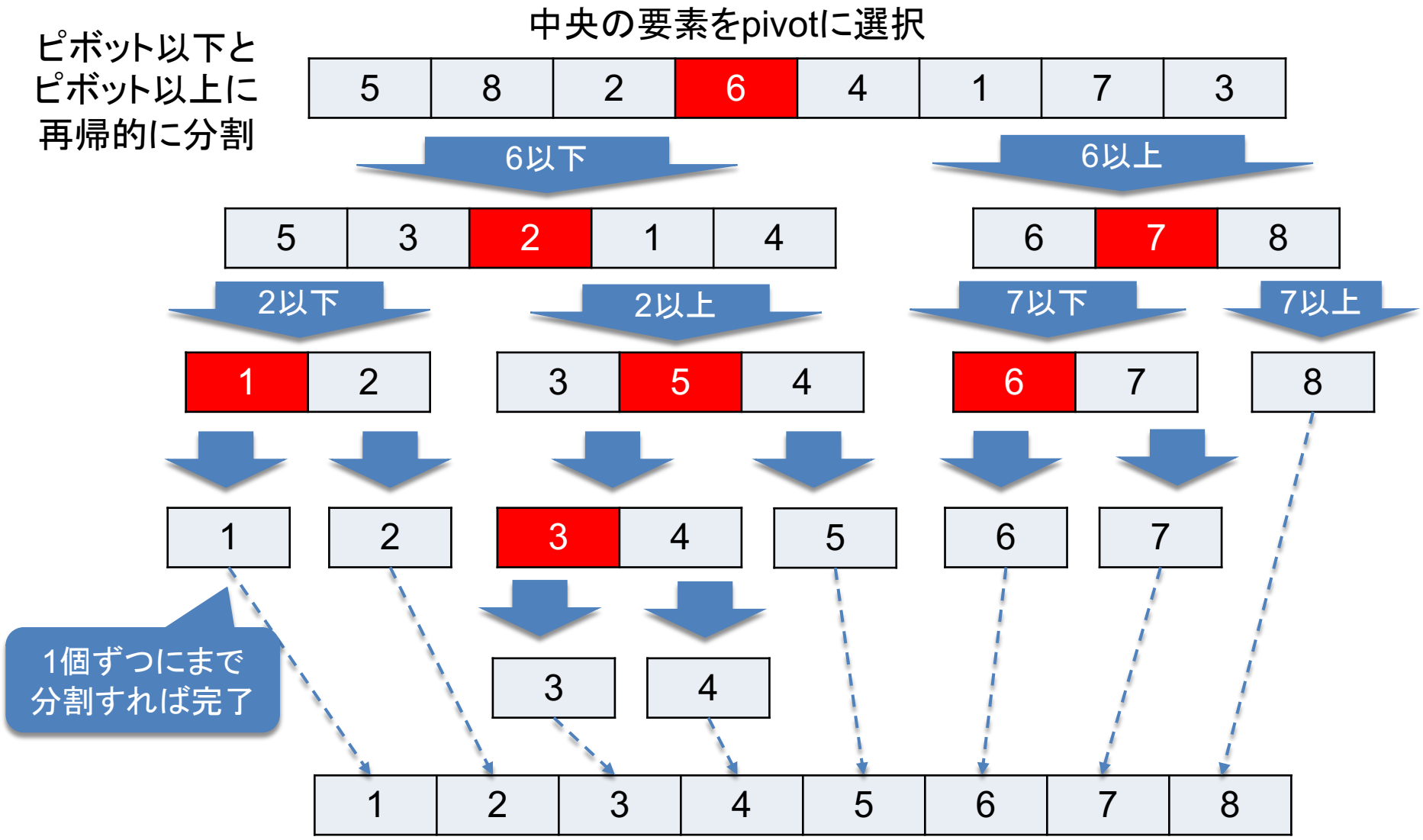
- アルゴリズム
  - まず配列の中から「ピボット」(pivot，軸)と呼ばれる要素を選ぶ
    - 配列の中央の要素を用いる方法，ランダムに選ぶ方法などがある
  - 配列の中のピボット以下の要素を左側，ピボット以上の要素を右側に寄せて，配列を左右2つの部分配列に分割する
  - さらにそれぞれの部分配列に対しても，その中でピボットを選び，同様の処理を再帰的に適用して分割していく
  - すべての部分配列の要素が1つになるまで分割を繰り返すと，配列全体のソートが完了する
- 代表的な分割方法

「以下」と「以上」なのは，pivotが最大・最小値でもオーバーランを防ぐため



クイックソートの例

ピボット以下と
ピボット以上に
再帰的に分割



確認問題

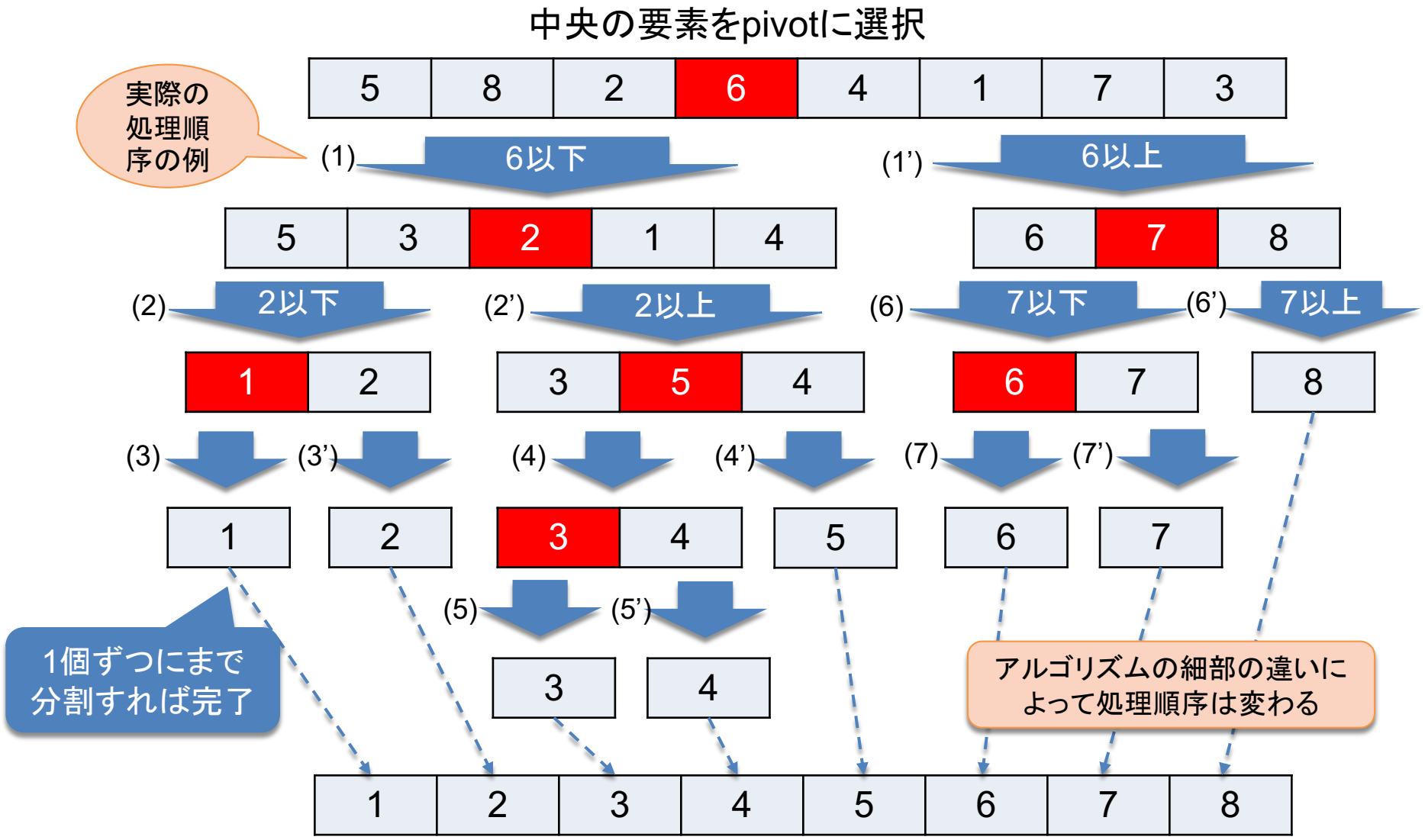
- クイックソートの理解
  - 配列aの内容をクイックソートで昇順に整列する際の変化の過程を図示し，値の比較と交換の回数を述べよ。
    - ビボットは中央の要素を用いよ。

a	4	1	3	5	2

  - 右のコードは部分配列a[left]～a[right]の中で，pivot以下の要素は前半，pivot以上の要素は後半に集める処理である。空欄を適切に埋めて完成させよ。

```
int l = left, r = right;
while (true) {
    while (a[l] < pivot) {

    }
    while (a[r] > pivot) {

    }
    if (l >= r) break;

    double t;



    l++; r--;
}
```

クイックソートの処理順序の例

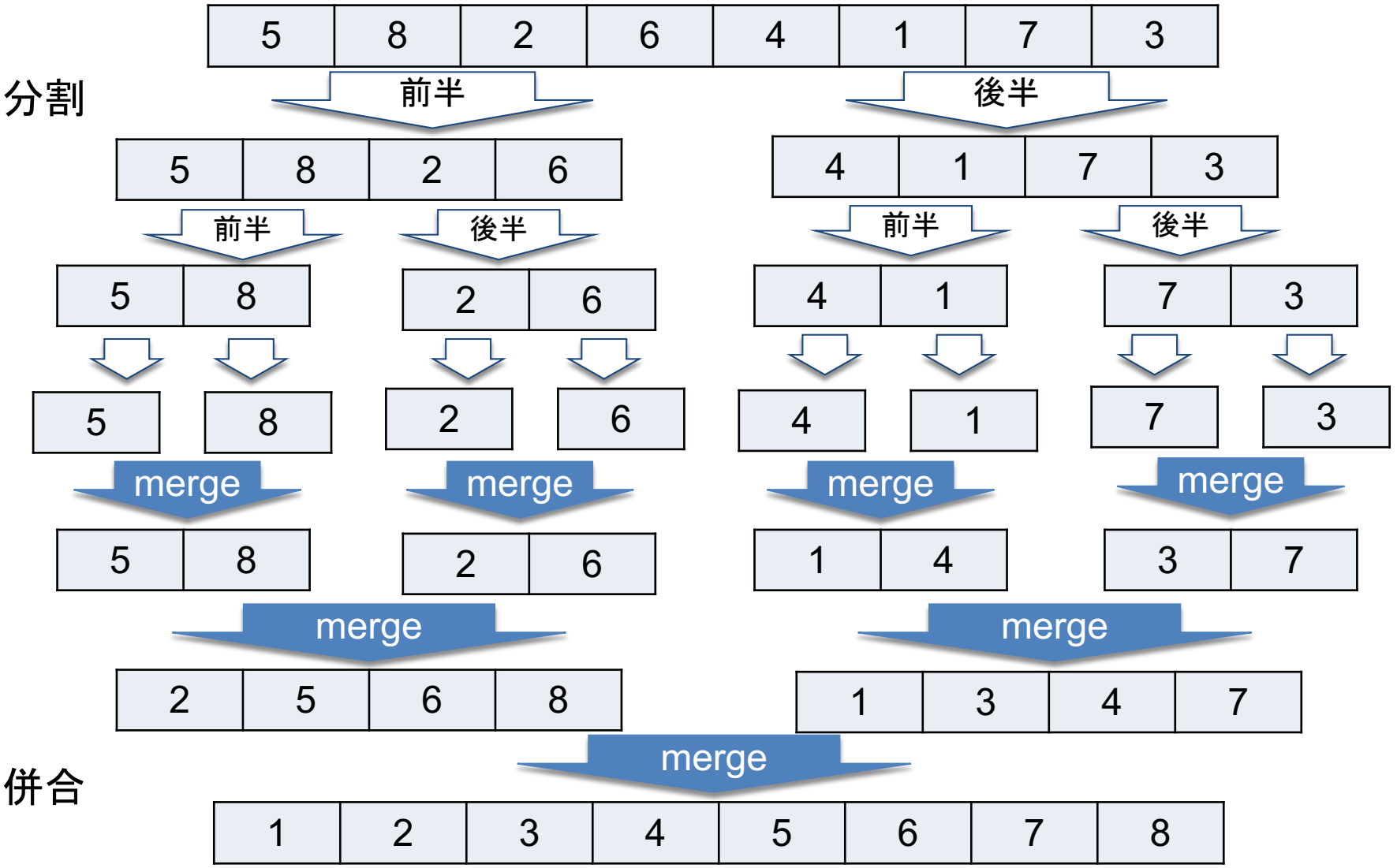


マージソート

- アルゴリズム
  - 配列を要素数が半分ずつになるように，前半部分と後半部分に再帰的に分割していく
  - 要素が1つになるまで分割されたものは，整列済みとみなせる
  - 整列済みになった前半部分と後半部分を，分割とは逆に再帰的にマージしていくと，最終的に配列全体のソートが完了する
  - ただし，前半部分と後半部分をマージして同じ領域に入れなおすには，前半部分を退避して空けておくことが必要になる

- 分割統治アルゴリズム
  - 再帰的に，対象をいくつかの部分に分割し，それらの部分にも同様な処理を適用し，結果を再結合していくアルゴリズム
  - クイックソート，マージソート，2分探索法など

マージソート(トップダウン型)の例



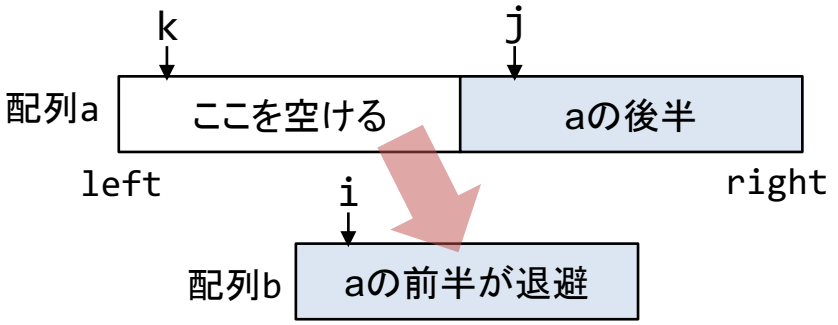
確認問題

- マージソートの理解
  - 配列aの内容をマージソートで昇順に整列する際の変化の過程を図示し，値の比較と交換の回数を述べよ。

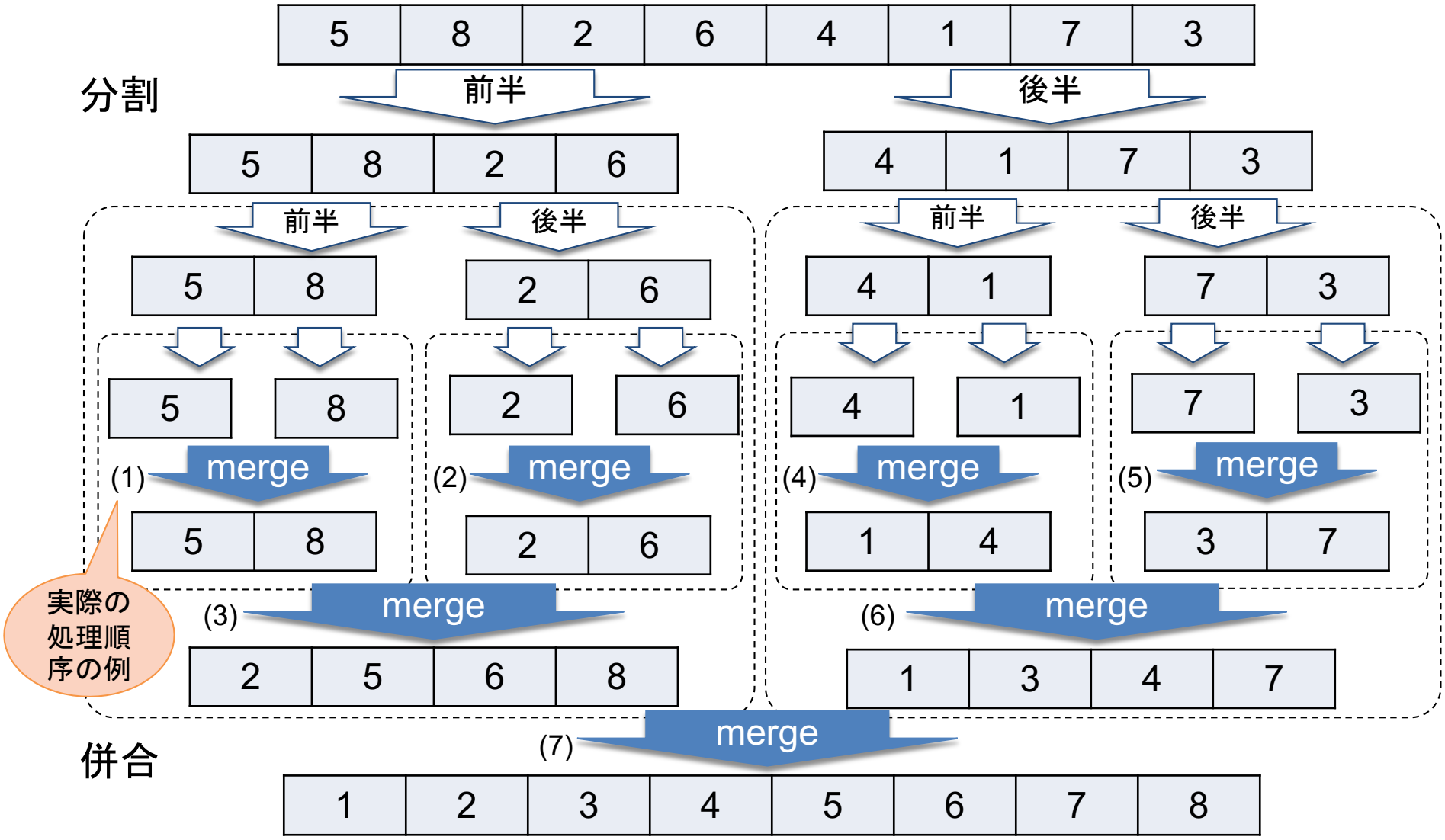
a	4	1	3	2

  - 右上のコードは，部分配列a[left]～a[right]の中の前半部分を配列bに退避した後，a[left]以降にマージしなおす処理である。空欄を適切に埋めて完成させよ。

```
int i=0, j=mid+1, k=left;
while (k <= right) {
    if (i >= b.length)
        a[k++] =
    else if (j > right)
        a[k++] =
    else if (b[i] <= a[j])
        a[k++] =
    else
        a[k++] =
}
```

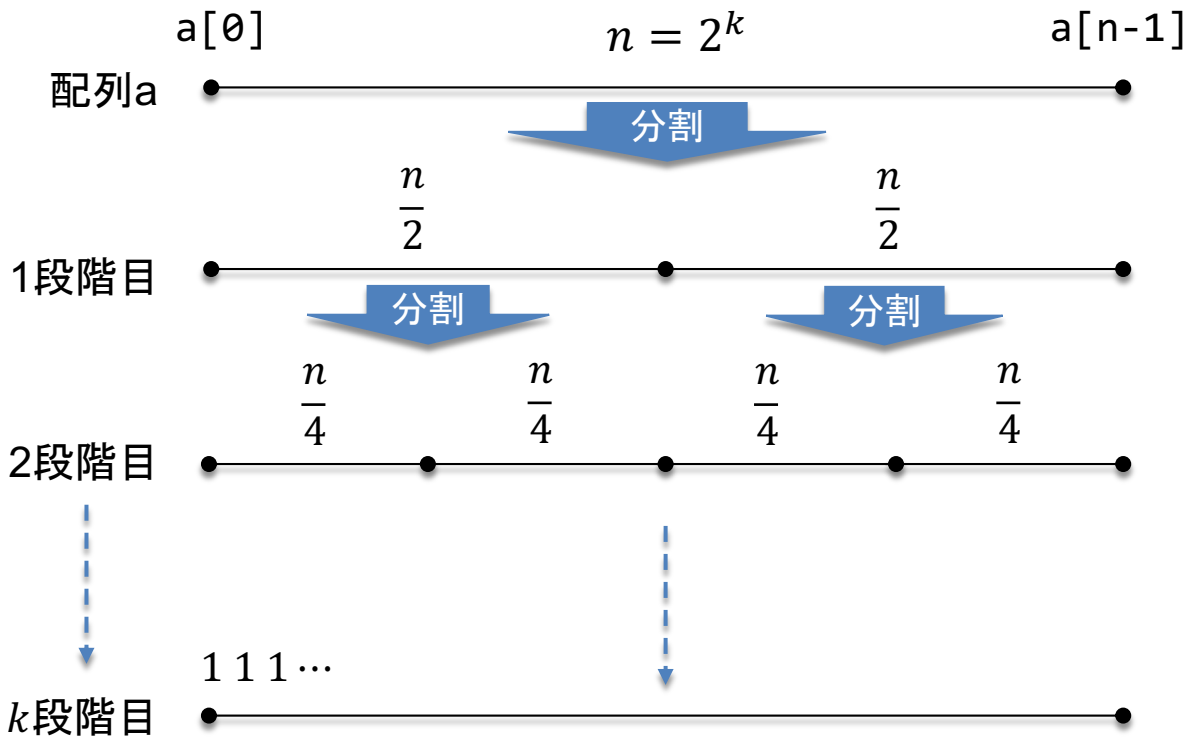


マージソートの処理順序の例



クイックソート/マージソートの計算量

理想的なクイックソートまたはマージソート



要素同士の比較回数

n回

n回

n回

nに比例する回数の処理をk段階

計算量はnkに比例

nの式で表すと？

k段階目に各1個になるから $\frac{n}{2^k} = 1$

ソートアルゴリズムの特徴

- ソートの安定性
  - 同じ優先度の（キーが同じ）データの順序が維持されること
  - マージソートや前回の単純なソートは（順序が）安定である
  - クイックソートやシェルソートは高速だが（順序が）安定でない
- 内部ソート vs 外部ソート
  - 対象の配列の中で処理が完結するものを「内部ソート」という
  - マージソートは追加の記憶領域が必要な「外部ソート」である
- 比較ソート vs その他のソート
  - 通常のソートは，要素同士を比較して交換していく（比較ソート）
  - 要素の値（範囲）が少ない場合，数え上げや分類でソートできる

補足：クイックソートの平均計算量

要素数がn個のときの平均比較回数をA_nと表すと，それを2分割する組み合わせから，以下のように考察される。

$$A_1 = 0$$

$$A_2 = 2 + A_1 + A_1 = 2 + 2A_1$$

$$A_3 = 3 + \frac{(A_2+A_1)+(A_1+A_2)}{2}$$

$$= 3 + \frac{2}{2}(A_1 + A_2)$$

$$A_4 = 4 + \frac{(A_3+A_1)+(A_2+A_2)+(A_1+A_2)}{3}$$

$$= 4 + \frac{2}{3}(A_1 + A_2 + A_3)$$

よって，A_nは以下のように表せる。

$$A_n = n + \frac{2}{n-1}\sum_{k=1}^{n-1} A_k$$

さらに，ここでnA_{n+1}と$(n-1)A_n$の式を縦に並べて引き算する。

$$nA_{n+1} - (n-1)A_n$$

$$= (n+1)n - n(n-1) + 2A_n$$

$$= 2n + 2A_n$$

この式を整理すると，漸化式を作れる。

$$nA_{n+1} = (n+1)A_n + 2n$$

$$\frac{A_{n+1}}{n+1} = \frac{A_n}{n} + \frac{2}{n+1}$$

A_1も考慮してA_n/nの一般項を求める。

$$\frac{A_n}{n} = 2\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right) - 2$$

ここで，nが十分に大きければ，

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \approx \log n + \gamma$$

となることが知られているので（$\gamma \approx 0.577$），平均計算量は以下のように表される。

$$O(A_n) \approx O(n \log n)$$

補足：ボトムアップ型のマージソート