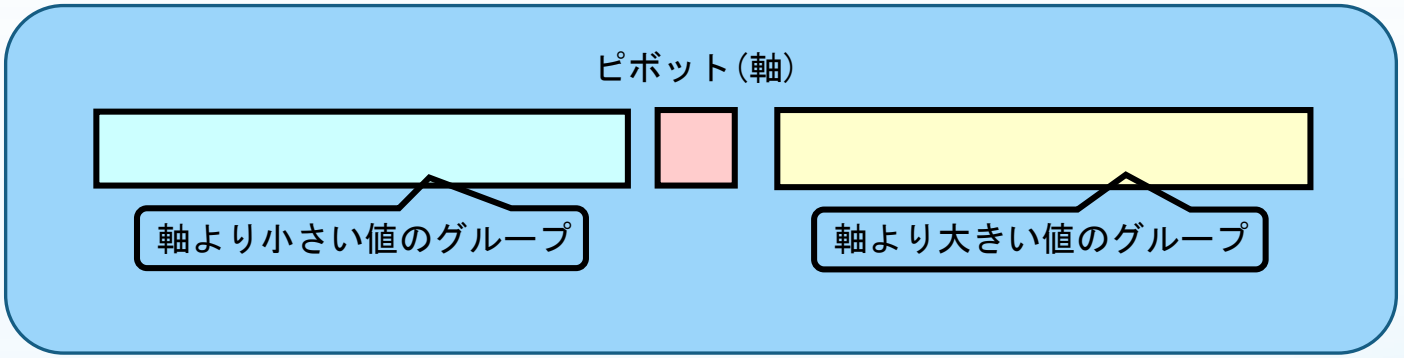
高速整列アルゴリズム

クィックソートとは

① クィックソート

❶ 適当にサンプリングして得られたデータNに着目し、
N以下のデータ系列とN以上のデータ系列に分割する。
❷ このデータNを軸(ピボット)という。
❸ データNを軸として、
その軸の値より小さい要素は軸より前方へ、
大きな要素は軸より後方へ振り分ける。
❹ 適当な長さの系列になるまで分割を繰り返し、
それぞれの系列をソートすれば
1つのソート済み系列となる。

② クィックソートの図



③ 分割の繰返しに再帰呼び出しを利用した
高速の整列法である。

④ ピボットの選び方には、
中央値や3要素(右・中央・左)の中間値の考え方がある。

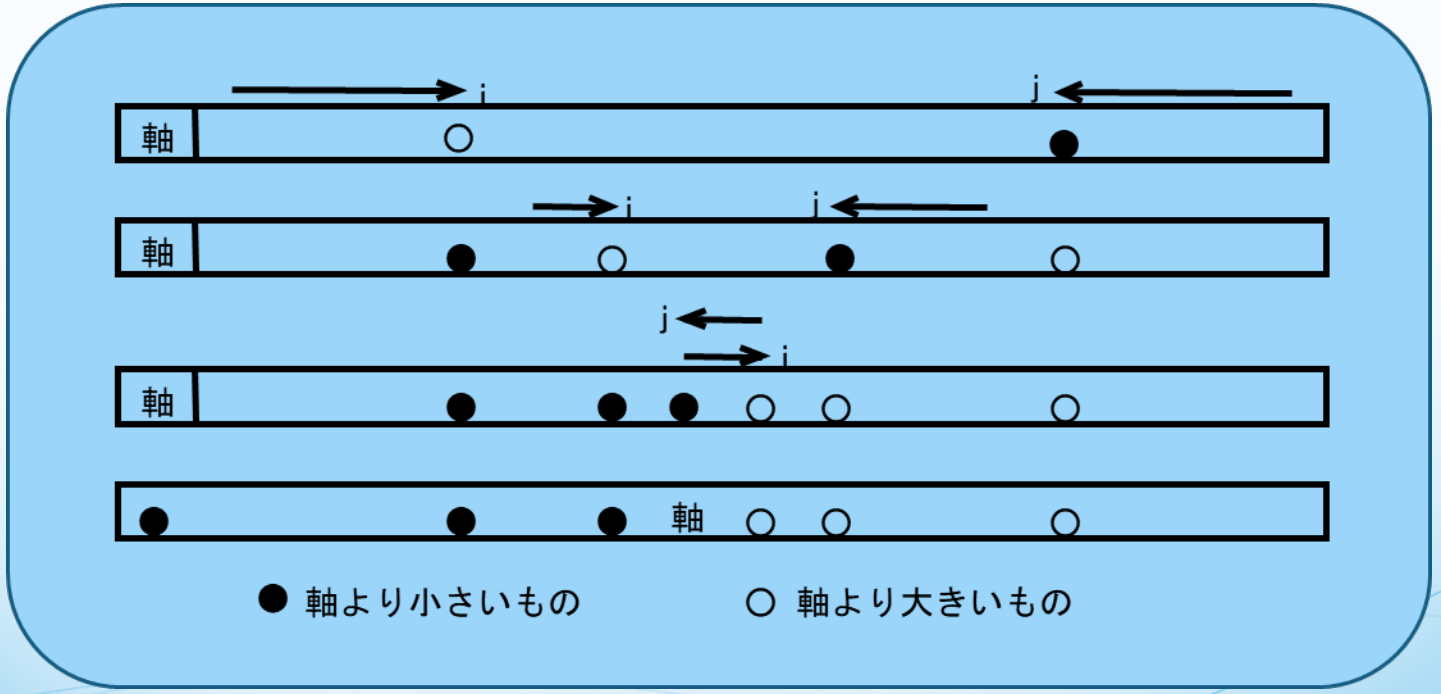
クィックソートの考え方

① クィックソートの考え方

❶ 大きさnの数列を考える。
❷ 数列の左側から走査していく変数をiとする。
❸ 右側から走査していく変数をjとする。
❹ 最も左側の要素(要素番号1)を基準値に設定する。

② クィックソートの考え方の図を次ページに示す。

③　クィックソートの考え方の図



④　図の説明

❶　図において、〇丸は基準値以上の値、
❷　●丸は基準値よりも小さい値、
❸　図の左右から同時に走査を繰り返して、
❹　左からの走査が〇丸に出会うと停止し、
❺　右から走査が●丸に出会うと停止し、
❻　〇丸と●丸の要素を交換する。

⑤　i≧jになるまでこの走査を繰り返すと、
左側に●丸、右側に〇丸が集まる。

クィックソートのアルゴリズム

① 左部分列と右部分列を作る走査は次のように行う。

❶ iを数列の左側＋1(要素番号2)、
　　jを数列の右端(要素番号n)に設定する。
❷ 数列を左から右に走査して、
　　要素の値が基準値以上になる位置iを見つけ、
　　その位置で走査を停止する。
❸ 数列を右から左に走査して、
　　要素の値が基準値よりも小さくなる位置jを見つけ、
　　その位置で走査を停止する。

❹ i≧jならば、
ループを抜け、❻に進む。
❺ i＜jならば、i項とj項を交換し、❷に戻る。
❻ 基準値とj項を交換する。

② この走査によって、
要素番号１から要素番号j－１までの左部分列(●丸)と
jからnまでの右部分列(〇丸)に分けることができる。

③ 基準値は〇丸の部分列に入る。

④ 次回の対象になる2部分列は次のようになる。

❶ 左部分列は　左端(要素番号１)からj－１
❷ 右部分列は　jから左端(要素番号n)

クィックソートの比較回数

① 平均比較回数は$n\log_2 n$

② 最大比較回数はn^2

クィックソートの具体例

① 問題

次に示す10個のデータを
クィックソートのアルゴリズを利用して整列する。

41 24 76 11 45 64 21 69 19 36

② 操作手順

❶ 基準値を41に、左側の初期値を24、
右側の初期値を36に設定する。

❷ 左側の走査は76で、
右側の走査は36で停止する。76と36を交換する。

41 24 36 11 45 64 21 69 19 76

❸ 左側の走査は45で、
右側の走査は19で停止する。45と19を交換する。

41 24 36 11 19 64 21 69 45 76

❹ 左側の走査は64で、
右側の走査は21で停止する。64と21を交換する。

1	2	3	4	5	6	7	8	9	10
41	24	36	11	19	21	64	69	45	76

❺ 左側の走査は64、右側の走査は21で停止する。

i＝7、j＝6となる。

❻　左側の41と21を入れ替えると、

21、24、36、11、19、41、64、69、45、76

③　41を基準値にして左部分列と右部分列に分かれる。

④　21～19の左部分列と
41～76の右部分列に対して、
同様の処理を繰り返す。

シェルソートとは

① シェルソートは、
挿入ソートを改良したものである。

② データ列の中からX間隔ずつ離れた
データを取り出して、その部分列のデータを
挿入ソートの考え方を利用して整列する。

③ このX間隔のギャップを、始めは大きく、
次第に小さくして最後に1にする。

④ ギャップの決め方は
種々あるが、よく用いられる方法にデータ数の
１／２、１／４　、…、１とする方式を用いる。

⑤ 例えば、データが８個ある場合、

❶ 最初の飛びは８／２＝４とし、
飛びの対のデータについて
挿入ソートを使用する。
❷ 次に、４／２＝２の飛びの対について
挿入ソートを使用する。
❸ 最後に２／２＝１の飛び、
通常の挿入ソートを行う。

⑥　挿入ソートは、
常に隣の要素と比較・交換するため、
実行速度が遅いという欠点があったが、
それを改善し、順番の違いを
早く修正することができる。

⑦　シェルソートの比較回数は、$n^{1.5}$に比例する。

シェーカソートとは

① シェーカソートは、
基本的には基本交換法(バブルソート)と同じである。

② バブルソートは常に左から右に比較していき、
右端に最大値(または最小値)がくるように整列する。

③ シェーカソートは左から右へ比較していき、
右端に最大値(または最小値)を固定した後に、
逆に右から左へ比較していき、
最小値(または最大値)を左端に
固定する方式である。

④ この操作を繰り返してデータを
昇順(または降順)に並べ替える方式である。

⑤ 左から右に比較していく場合、
データを交換した最後の位置を認識し、
その位置を起点にして
右から左への比較を開始する。

⑥ 右から左に比較する場合も、
データを交換した最後の位置を認識し、
その位置を起点として
左から右への比較を開始する。

⑦　左右からの走査が1回完了すると、
　　次の走査の範囲LからRの要素の数は減少する。

⑧　Lの左側、Rの右側は整列済みとなる。

⑨　この走査を繰り返して、L≧Rとなると、処理を終了する。

シェーカーソートの具体例

19　10　　26　　43　　21　　13　　比較回数左から5回
↑L　　Lは左から右に走査
10　19　　26　　43　　21　　13
10　19　　26　　43　　21　　13
10　19　　26　　43　　21　　13
10　19　　26　　21　　43　　13
10　19　　26　　21　　13　　43
↑R最後に交換した要素

10	19	26	21	13	43	比較回数右から4回
↑				↑R		Rは右から左に走査
10	19	26	13	21	43	
10	19	13	26	21	43	
10	13	19	26	21	43	
			↑L		↑R	

① 次の走査は、19(L)と21(R)の間で走査を繰り返し、
最後に交換した位置を記憶する。

② 走査の左の起点Lと右の起点RがL≧Rとなると、
走査を終了する。

ヒープソートとは

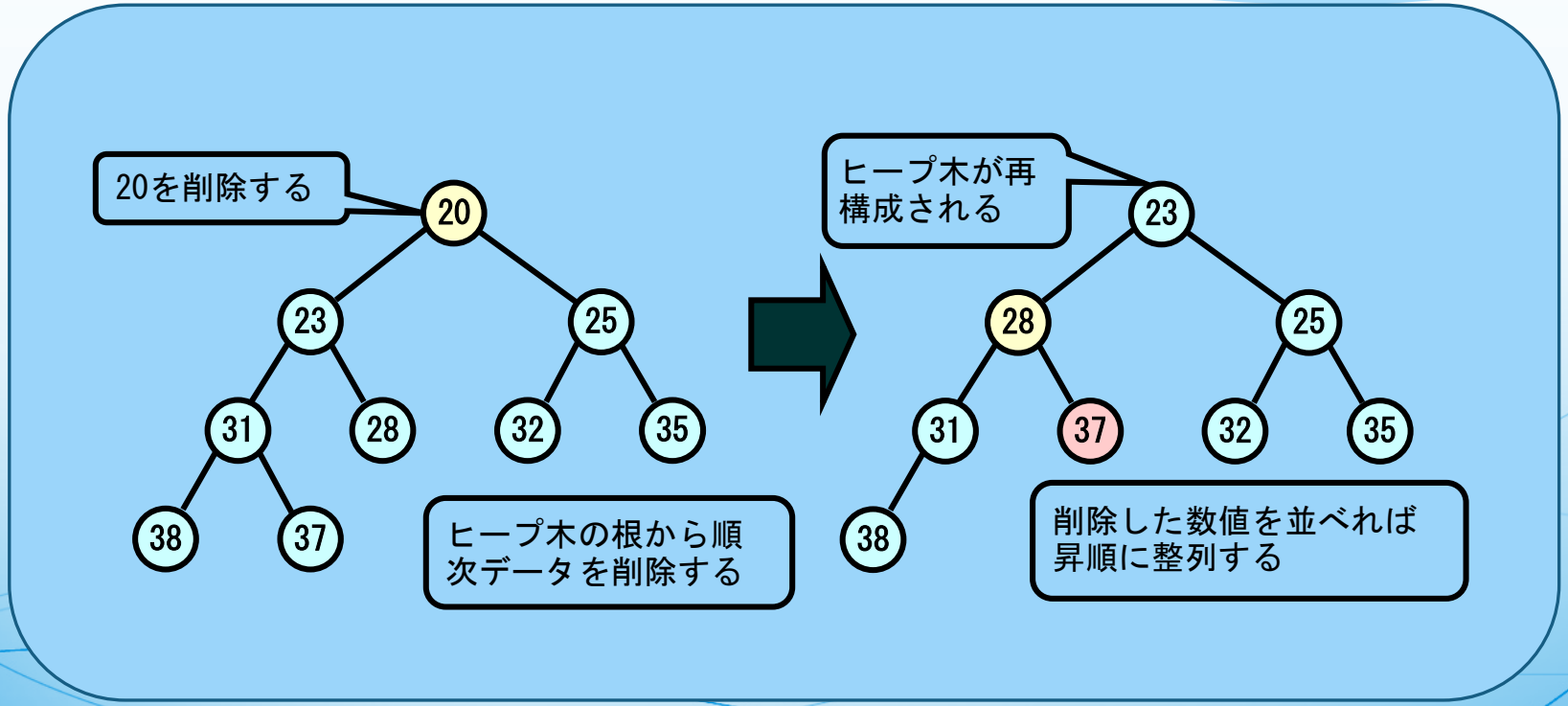
① ヒープソートは、
ソートをヒープ木の構造を利用して行う方法である。

② ヒープソートは、根の部分のデータを取り出した後、
ヒープを再構成する作業を繰り返せば、
ソート済みデータ系列が得られる。

③ データ構造のヒープ木の削除の考え方が使用される。

④ ヒープソートの比較回数は、
平均比較回数も最大比較回数も$n\log_2 n$となる。

ヒープ木の削除の図


20を削除する
20
23
25
31
28
32
35
38
37
ヒープ木の根から順
次データを削除する
ヒープ木が再
構成される
23
28
25
31
37
32
35
38
削除した数値を並べれば
昇順に整列する