

第6章 アルゴリズム

アルゴリズムとは？

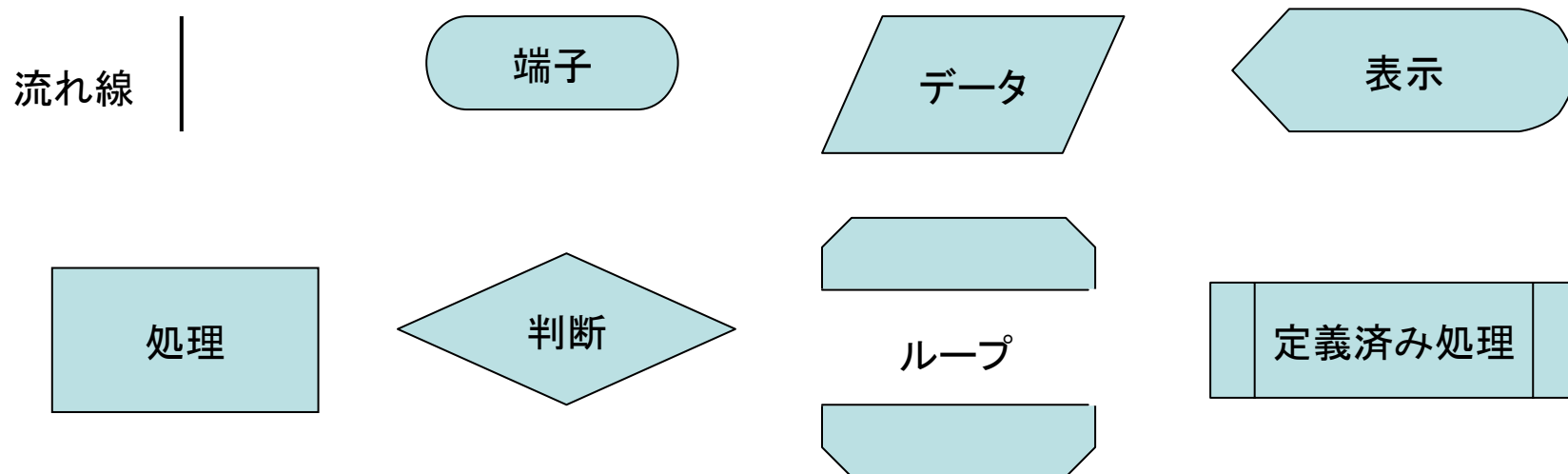
- ある処理を(効果的に)行う手順を示したもの
 - 高速化やメモリ使用量削減のための試み
 - これまた過去の様々な試みの集大成でもある
- 特に覚えてもらいたいもの
 - オーダによる計算量／メモリ使用量の見積もり
 - それぞれのアルゴリズム構築の理念

節概要

- フローチャート
- オーダ記法
- ソート
 - バブルソート
 - クイックソート(分割統治法)
 - マージソート
- データ探索アルゴリズム
 - 線形探索
 - 2分探索

フローチャート

- アルゴリズムの表し方の1つ
 - 擬似言語を用いて表すこともある
 - もちろん、プログラミング言語による記述もOK
- JIS X0121と規格化もされている



擬似言語でアルゴリズムを表した例

Algorithm 2: The behavior description of the D flip-flop as the pipeline register in a PSU processor.

Input: The data input D , the reset signal $reset$, the unifying signal U , and the clock signal clk .

Output: The data output out .

PSU_DFF(D , out , $reset$, U , clk)

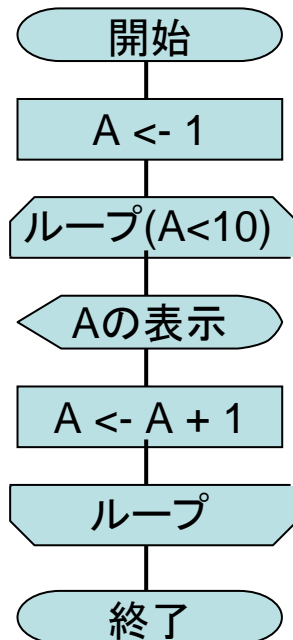
- (1) *define local register* Q
- (2) $out \leftarrow (U \& D) \mid (\overline{U} \& Q)$ /* Multiplexor */
- (3) $psu_clk \leftarrow U \& clk$
- (4) *always at positive edge of* psu_clk
- (5) **if** ($reset$) **then** $Q \leftarrow 0$
- (6) **else** $Q \leftarrow D$

[Lines (2), (3) and (4–6) work in parallel.]

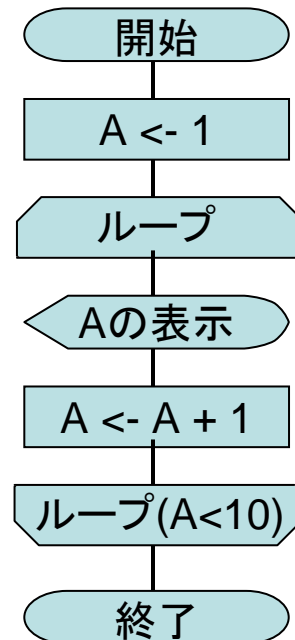
フローチャートの例

- ループの表記例
- cf. ループには前判定と後判定がある
 - 後判定は必ず1回はループ内を実行

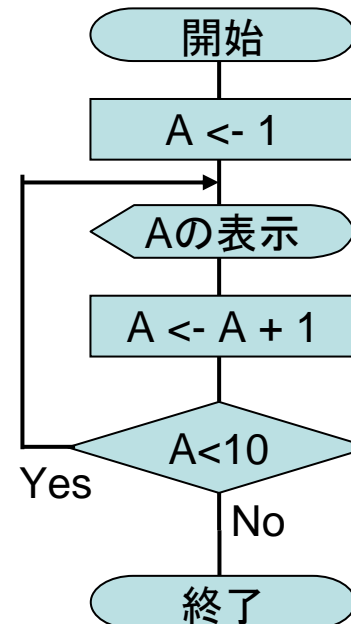
前判定ループ
の表記



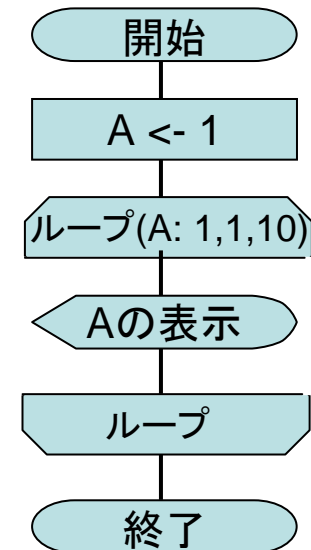
後判定ループ
の表記



後判定ループ
の表記その2



前判定ループ
の表記その2

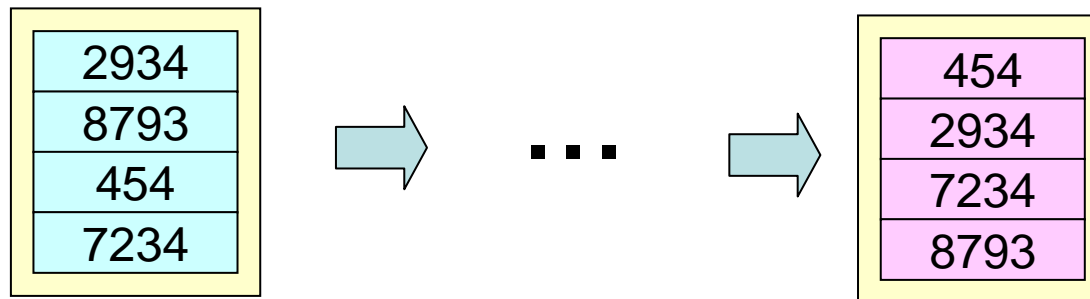


オーダー表記

- アルゴリズムの優劣はどう表す？
 - 計算量(=処理速度)で表す
 - メモリ使用量で表す
 - プログラミングのしやすさで表す
- 計算量やメモリ使用量を表すためにオーダー表記というものがある
 - データ数 N (N : 非常に大きい)に対しての計算量
 - 表記には支配的な項のみを残す
 - N^2 の項があるならば N の項は無視
 - 定数は N が非常に大きい場合は無視できるので書かない
 - ただし、同じオーダーのものを比較する場合は残す場合もある
 - 例: バブルソートが $O(N^2)$ 、クイックソートが $O(N \log_2 N)$

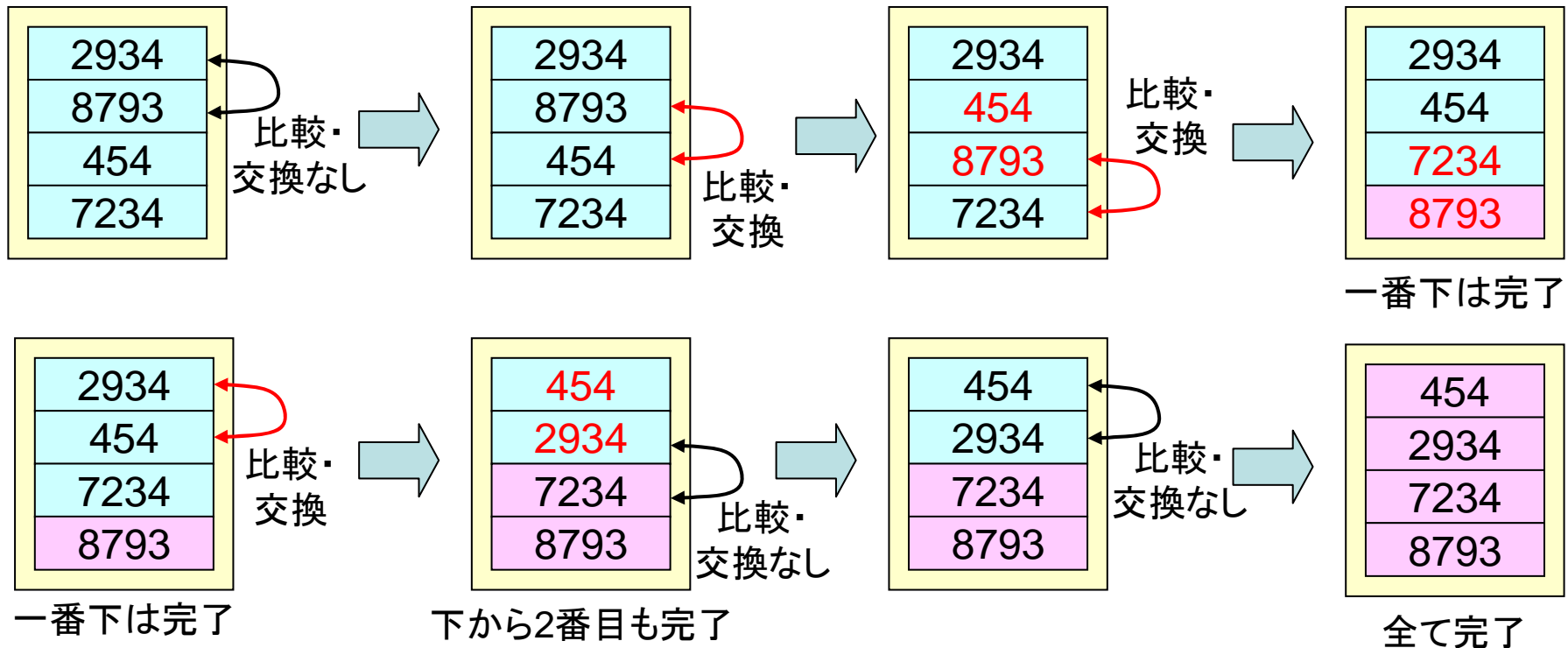
ソートとは？

- ばらばらな順で並んでいるデータを並び替えること
 - 一般的には、数値データの並び替えを対象とする
 - 文字データを数値データと考えれば、文字データの並び替えにもすぐに応用可能



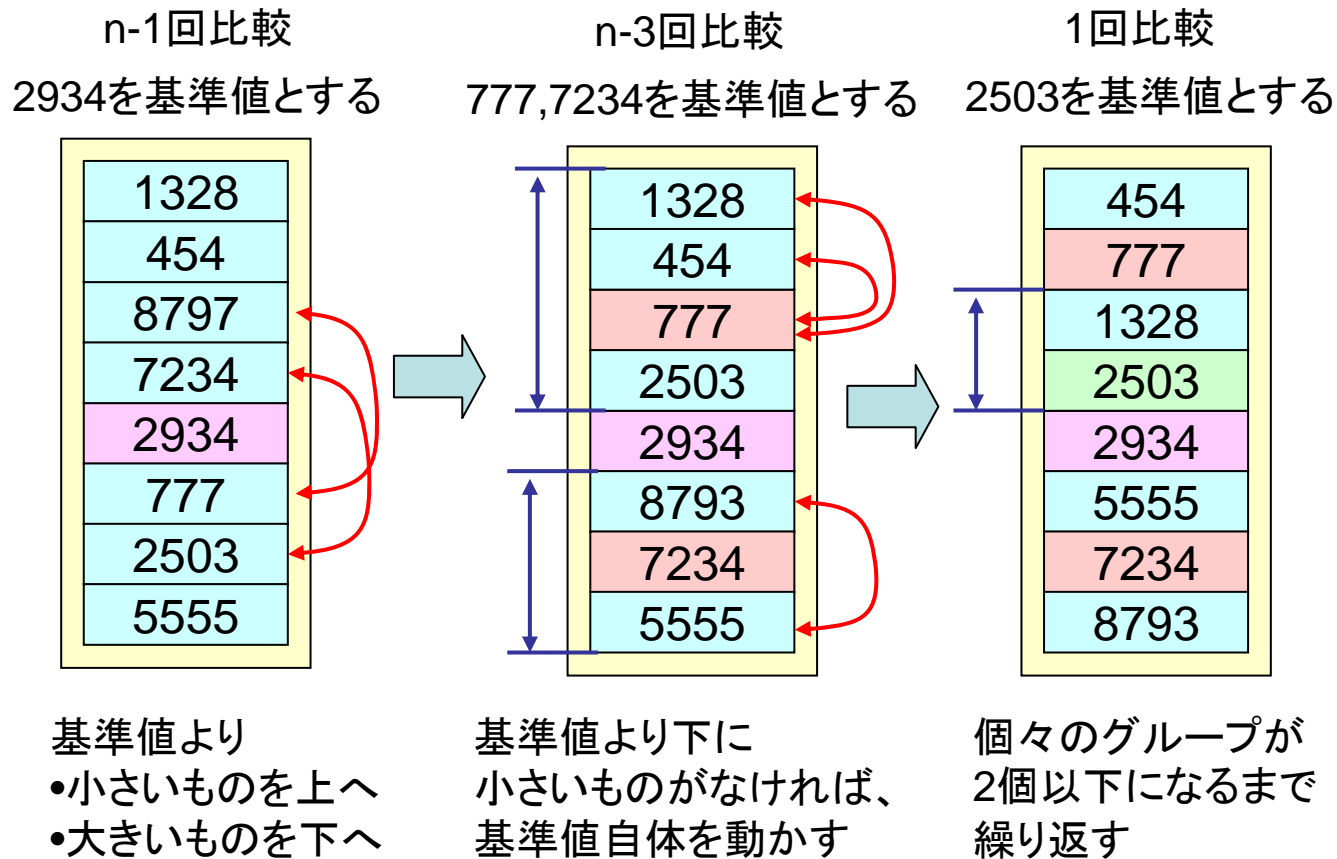
バブルソート

- 前から順番に比較、大きい方を後ろへ交換を繰り返す
 - 下までの比較中に交換が発生しない場合は、整列しているものとして打ち切り可能



クイックソート

- 基準値をもとにデータを比較／入れ替え

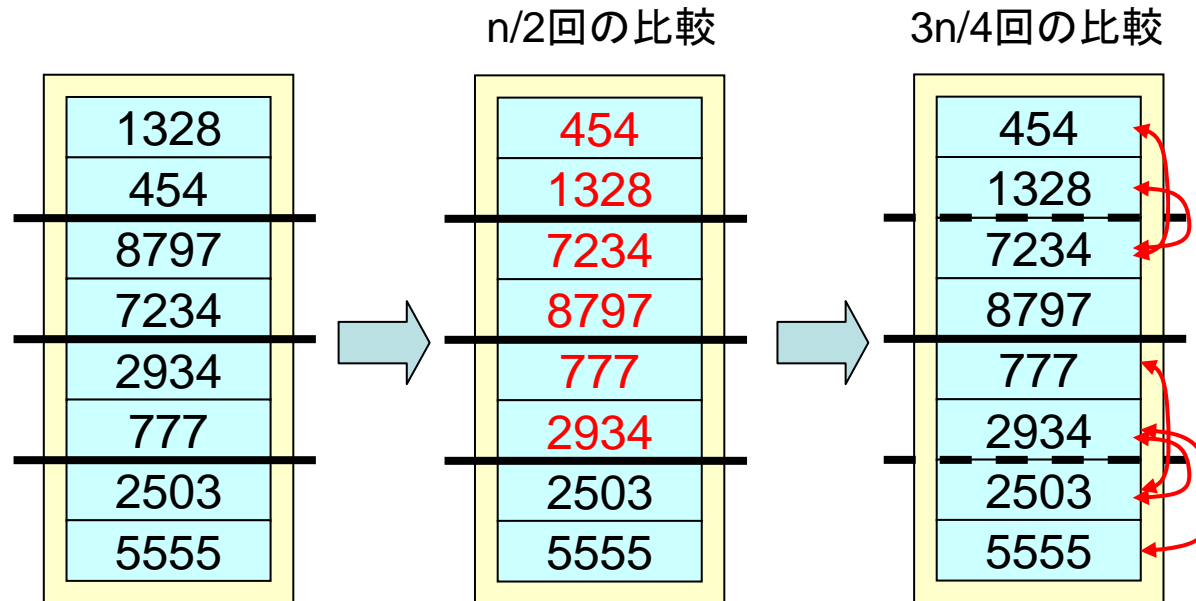


クイックソート関係のtips

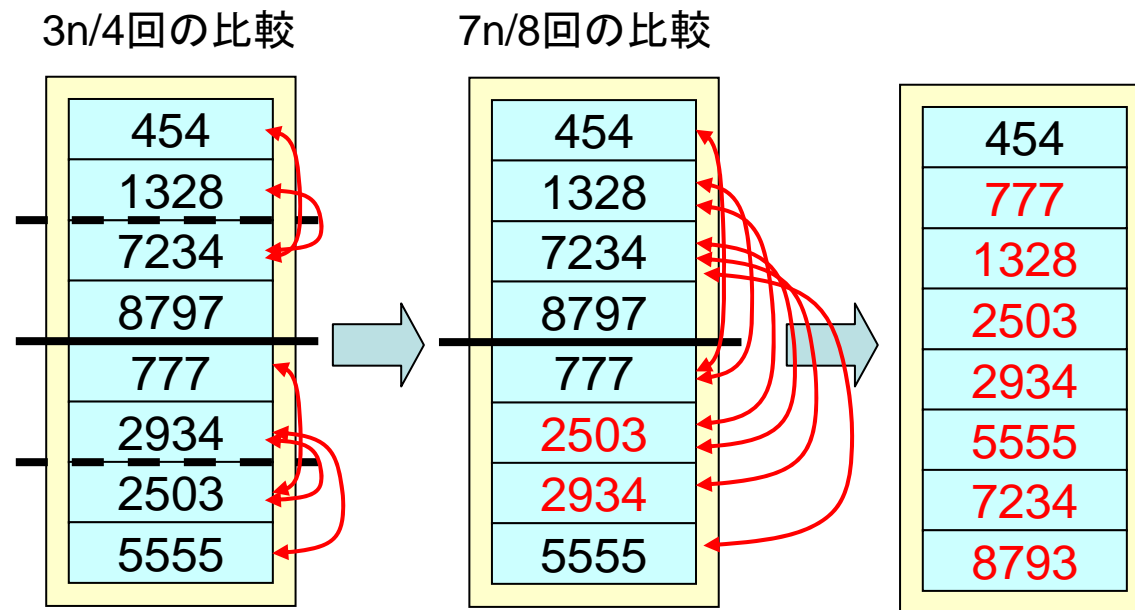
- **分割統治法**により比較回数を減らしたり、総ステップ数を減らしたりしている
 - 分割統治法:問題を小さな問題に分割してゆくことで、処理の複雑度を低減させていく手法
 - 基本的に、再帰的な作業に分割して、再帰的な関数呼び出しで実現
- 基準値は中央値が望ましいため、いくつかの値から中央値を取るという方法も使う
- 作業用の空間を別に準備することで、基準値に関する入れ替えを減らすことも可能

マージソート

- 各グループ内の値の数が2個以下になるまで分割
- 各グループ内で値のソート
- 隣接するグループ内の値を先頭同士から比較して並べ替え



マージソート



その他のソート手法

- 挿入法

- 未ソートのデータに対して、すでにソート済みのデータの中から挿入すべき箇所を特定し、そこに挿入するという作業を続けてゆく方法
- 挿入は、1つずつ後ろから前に移動させる形を取る
- 例: 2 5 1 4 3 -> 2 1 5 4 3 -> 1 2 5 4 3

- 選択法

- 未ソートのデータの中から最大値／最小値を求めた後、それを対応する場所に格納してゆく方法
- 例: 1 4 3 2 5 -> 1 3 2 4 5

その他のソート手法

- シェルソート
 - 挿入法の改良
 - 適当な間隔 h ごとのデータを別グループとし、そのグループごとのデータに対して挿入法を適用し、 h をせばめてゆくという方法
 - 事前に小さい値を前に持つてくることで、データの移動を削減
 - 例: 3 6 8 7 1 4 2 9 5 -> 2 1 4 3 6 5 7 9 8 -> 1 2 3 4 5 6 7 8 9
- ヒープソート
 - 最大値／最小値を求めるのに適したデータ構造であるヒープを利用して改良した選択法
 - 最初に未ソートのデータをヒープに登録し、最大値／最小値を取り出して選択法を行う

各ソートアルゴリズムのオーダ

$O(N^2)$ 隣接交換法(バブルソート)

$O(N^2)$ 挿入法

$O(N^2)$ 選択法

$O(N \log_2 N)$ クイックソート

$O(N \log_2 N)$ マージソート

$O(N \log_2 N)$ ヒープソート

$O(N^{3/2})$ シェルソート

演習

- 以下の値にクイックソートを適用する様子を示さない
 - 3, 2, 4, 5, 1
 - 過程は1回の比較／入れ替えごとに示してください

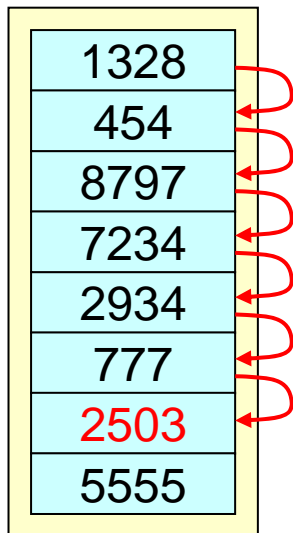
検索アルゴリズム

- 配列を使うもの
 - 線形探索
 - 2分探索
- 木構造を使うもの
 - 特に(バランス)2分木
- (グラフ構造を使うもの)

線形探索

- 配列の先頭から順番に探索
- オーダは $O(N)$

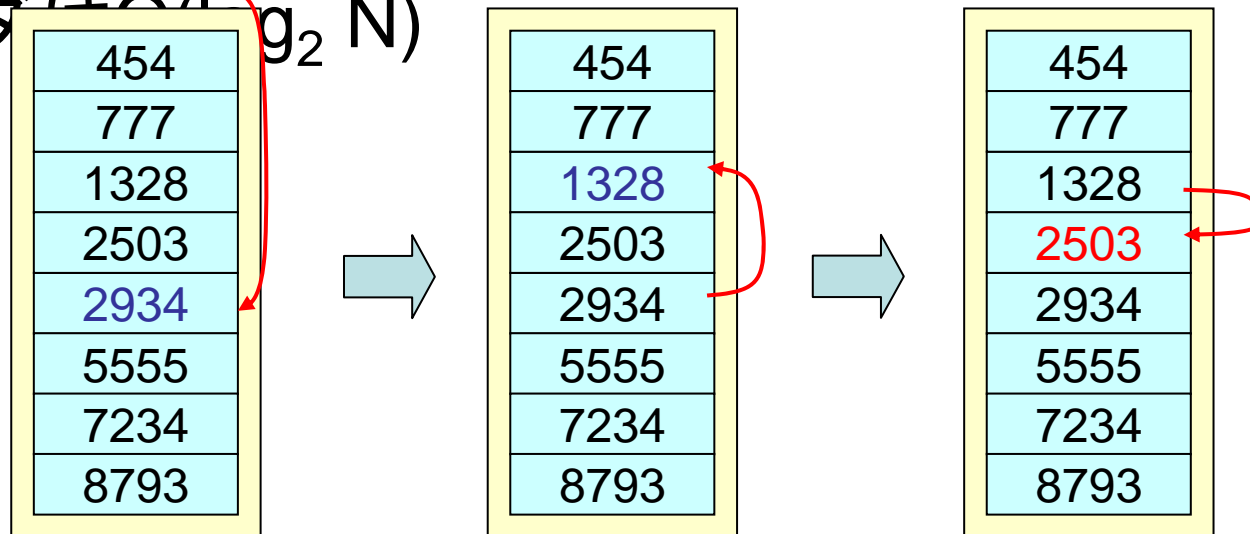
2503を探索



2分探索

- 分割統治法による探索
- データはあらかじめソートしておく
- 配列の中間点より探索を開始
 - 求める値が小さければ、前半分の未探索部の探索
 - 求める値が大きければ、後半分の未探索部の探索
- オーダーは $O(\log_2 N)$

2503を探索



6章のまとめ

- アルゴリズムはフローチャート等を利用して記述できる
- アルゴリズムの優劣の表し方として、オーダ記法がある
- ソートや探索アルゴリズムは複数あり、それぞれにアルゴリズム設計理念がある

中間試験について

- 問題数は8問
 - スライド内演習と同程度が6問
 - スライド内演習 + α の難易度が2問
- 資料持ち込みは可能です
 - ただし、検索は自力でやって下さい
 - 外部の検索機能は使ってはいけません
 - 自分で索引を作るのはOK

