

アルゴリズムと データ構造

コンピュータサイエンスコース
知能コンピューティングコース

第5回

ソートアルゴリズムの計算量の下界
第 p 要素選択アルゴリズム

塩浦昭義

情報科学研究科 准教授

shioura@dais.is.tohoku.ac.jp

<http://www.dais.is.tohoku.ac.jp/~shioura/teaching>

中間試験について

- 日時: 6月4日(木)8:50~10:20
- 過去にレポートを一度も出していない場合, 受験は不可
- 教科書, ノート等の持ち込みは一切不可
- 座席はこちらで指定
- 試験内容: 第6回目までの講義で教えたところ
 - アルゴリズムやデータ構造の挙動
 - 時間計算量の解析, および関連する証明問題
 - 用語の定義, など
- 50点満点, 24点以下は追試レポートもしくは単位不可

ソートアルゴリズムの計算量の下界

- ソートアルゴリズムの時間計算量

- バブルソート, クイックソートは $O(n^2)$

- マージソート, ヒープソートは $O(n \log n)$

2要素の
比較のみを
使った
ソートアルゴ
リズム

- バケットソート, 基数ソートは $O(n)$

(バケツの数, 桁数などを定数と見なしたとき)

- 数値情報を利用
- 数値の範囲が限定されている

- ソートのために最低限必要な計算時間は？

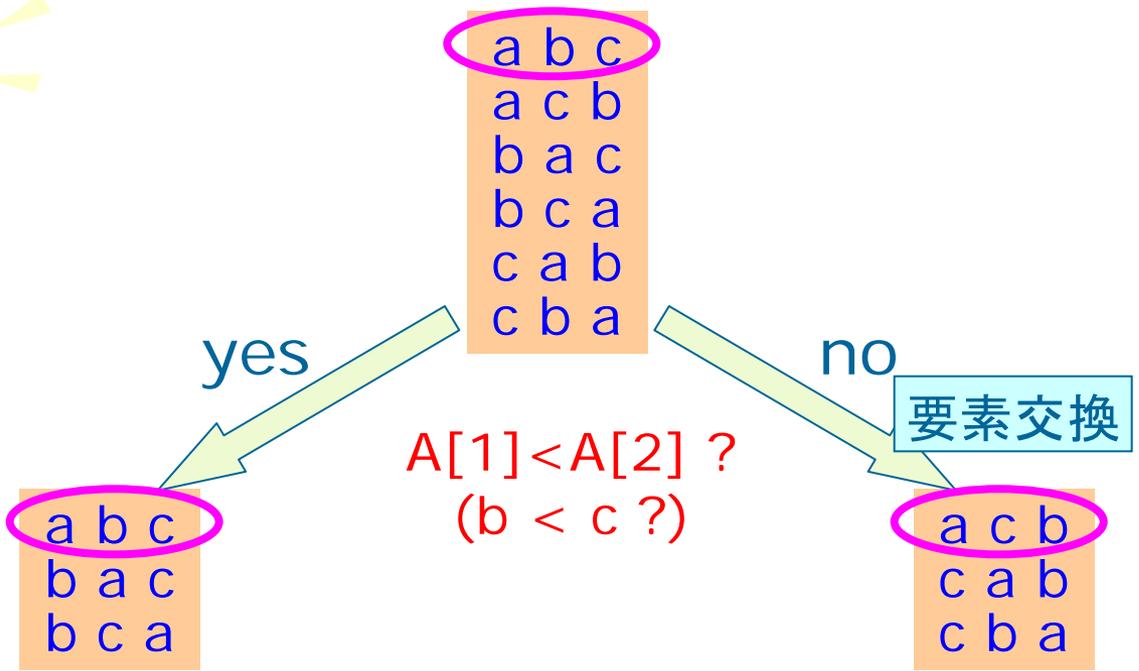
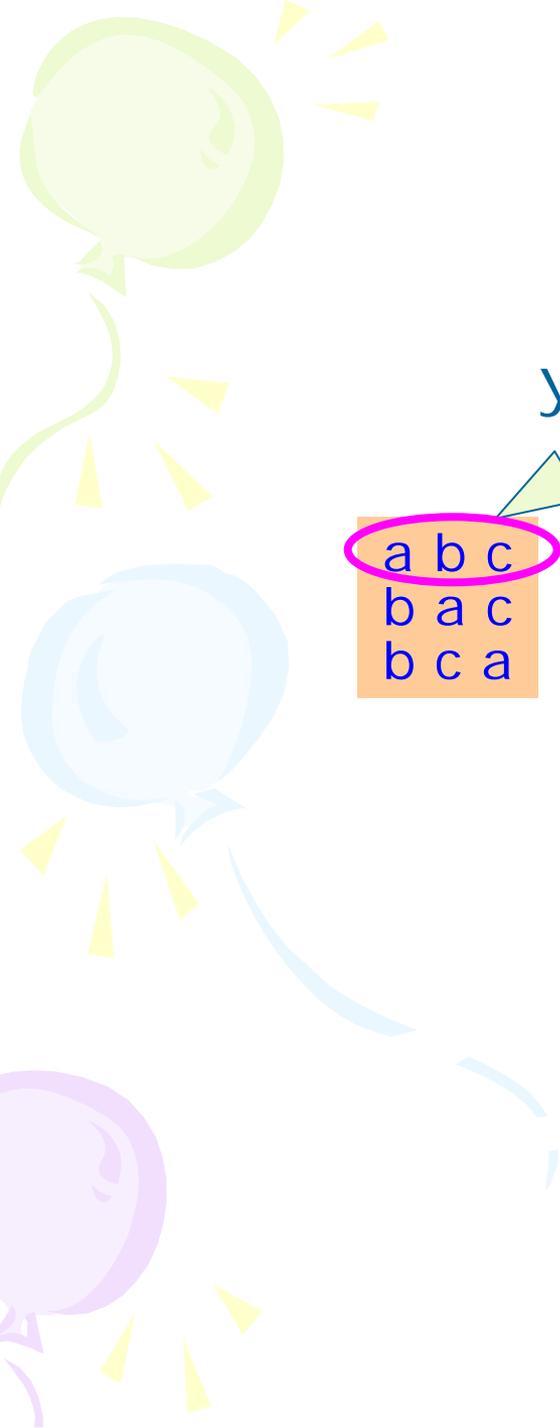
定理: 2要素の大小比較に基づくソートアルゴリズムは $\Omega(n \log n)$ の時間計算量を必要とする.

ソートアルゴリズムの決定木 (その1)

- 2要素の比較に基づくソートアルゴリズムの進行状況は決定木により表現可能
 - 比較を1回行なう度に状況がどのように変化するかを表す
- 例: $A[0] = a, A[1] = b, A[2] = c$ に対するバブルソート
 - a, b, c の値および大小関係は未知
 - 簡単のため, 全ての値は異なると仮定
 - 起こりうるソート結果は
 - $abc, acb, bac, bca, cab, cba$ の6通り
 - 比較を行なう度に, 起こりうるソート結果の数が減少

ソートアルゴリズムの決定木 (その2)

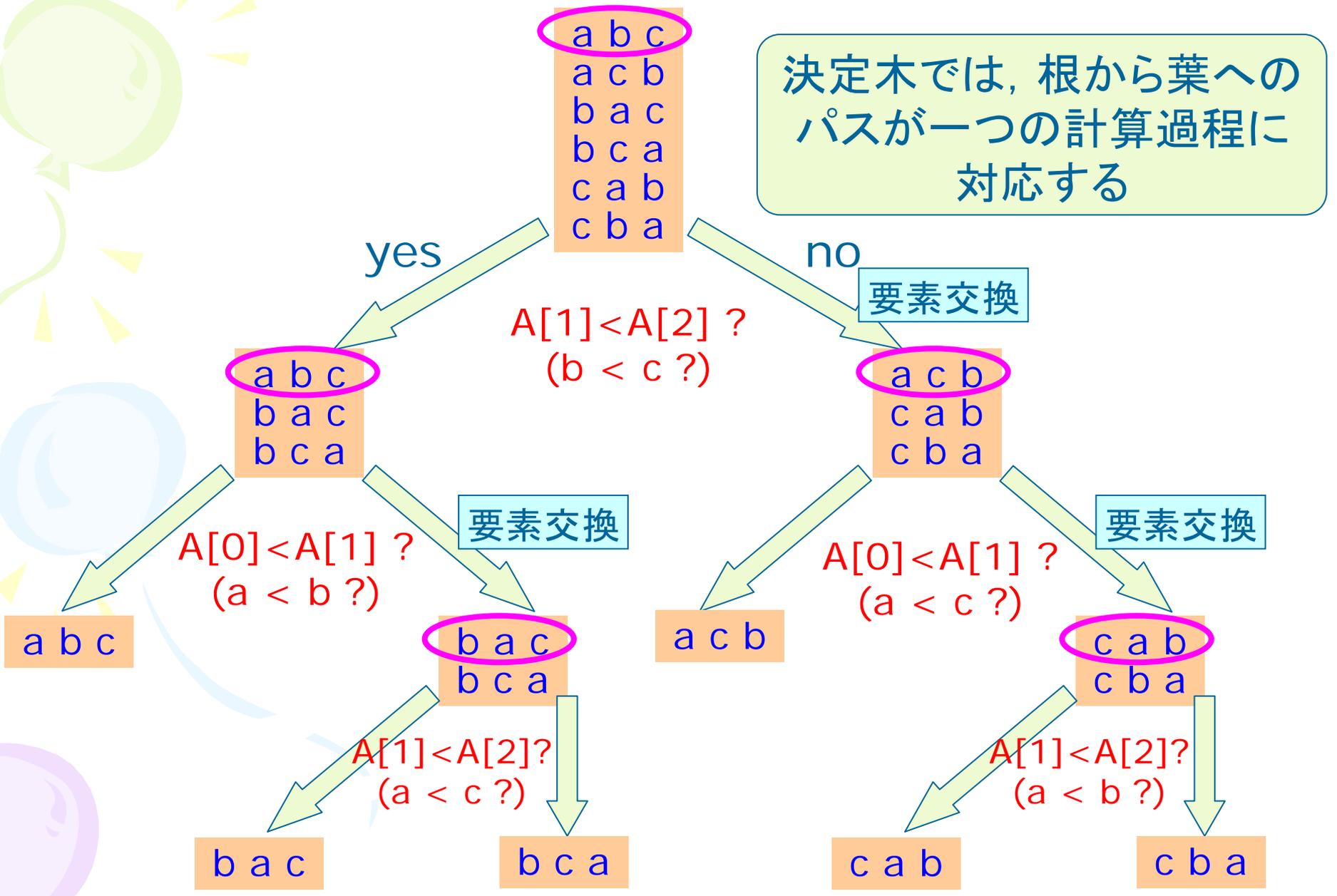
- バブルソートでは, 最初に $A[1]=b$ と $A[2]=c$ を比較
 - $b < c$ の場合,
 - 起こりうるソート結果は abc, cab, cba の3通り
 - バブルソートでは要素の入れ替えをしない → $A: abc$
 - $b > c$ の場合,
 - 起こりうるソート結果は acb, bac, bca の3通り
 - バブルソートでは $A[1]$ と $A[2]$ を入れ替え → $A: acb$
 - 以下, この場合を考える



ソートアルゴリズムの決定木 (その3)

- バブルソートでは, 次に $A[0]=a$ と $A[1]=c$ を比較
 - $a < c$ の場合,
 - 起こりうるソート結果は abc の1通り
 - バブルソートでは要素の入れ替えをしない → $A: abc$
 - バブルソートでは, この後(不要な) $A[1]$ と $A[2]$ の比較をして終了 → $A: abc$ をソート結果として出力
 - $a > c$ の場合,
 - 起こりうるソート結果は cab, cba の2通り
 - バブルソートでは $A[0]$ と $A[1]$ を入れ替え → $A: cab$
 - バブルソートでは, この後 $A[1]$ と $A[2]$ の比較を行なう.

決定木では、根から葉へのパスが一つの計算過程に対応する



比較回数の下界値

- 決定木を使って、アルゴリズムでの比較回数の下界値を解析する
 - 決定木の根から葉へのパスの長さ
 - \leq アルゴリズムでの比較の回数
 - よって、決定木の高さ $h \leq$ 最悪ケースでの比較の回数
 - 決定木の葉の一つ一つは n 要素の順列 (ソート列) に対応
 - $\rightarrow n!$ 個の葉が存在する
 - 高さ h の木の葉の数 $\leq 2^h \rightarrow n! \leq 2^h \rightarrow h \geq \log n!$
 - スターリングの近似公式より $\log n! \doteq n \log n - n$
(n が十分に大きいとき)
- ∴ 最悪ケースでの比較回数 $\geq h$
 - $\geq n \log n - n = \Omega(n \log n)$

第p要素の選択

- n 個の実数の中から p 番目に小さい(大きい)ものを選ぶ
 - $p=1 \rightarrow n$ 個の実数の中の**最小値**
 - $p=n \rightarrow n$ 個の実数の中の**最大値**
 - $p = \lceil n/2 \rceil$ または $\lfloor n/2 \rfloor \rightarrow n$ 個の実数の中の**中央値 (median)**
- 最小値(最大値)は簡単に $O(n)$ 時間で求めることが可能
 - 最小値の暫定値を $b = A[0]$ とする
 - 各 $i = 1, 2, \dots, n-1$ に対し $A[i]$ と b を比較,
 $A[i]$ の方が小さければ $b = A[i]$ とする
 - 最後に b を最小値として出力
- $1 < p < n$ のときの時間計算量は？

第p要素を求めるための 時間計算量

- n 個の実数の中から p 番目に小さいものを選ぶ
 - 簡単なやり方: n 個の実数を昇順にソートしたあとで, p 番目の要素を選ぶ → $O(n \log n)$ 時間
- 今回紹介する方法---ソートを使わないアルゴリズム
 - QUICKSELECT: 平均時間計算量 $O(n)$, 実用上高速
 - SELECT: 最悪時間計算量 $O(n)$, 実用上は少し劣る
 - (LAZYSELECT: 平均時間計算量 $O(n)$, 実用上高速)

QUICKSELECTの手順(その1)

- クイックソートと同様のアイデアを用いたアルゴリズム

p=5 の場合

- (0) $A[0], \dots, A[n-1]$ が全て同じ要素
→いずれかの要素を出力, 終了

3	3	3	3	3	3	3	3
---	---	---	---	---	---	---	---

- (1) $A[0], \dots, A[n-1]$ から
ひとつの値(軸要素)を選ぶ

3	2	0	5	8	3	4	1
---	---	---	---	---	---	---	---

- (2) 軸要素未満の要素と
それ以外に分割

k = 軸要素未満の要素数

2	0	1
---	---	---

$k=3$

3	5	8	3	4
---	---	---	---	---

$n-k=5$

QUICKSELECTの手順(その2)

(3) $p \leq k$ のとき,

軸要素未満の配列から第 $k-p$ 要素を再帰的に選択

$p > k$ のとき,

軸要素以上の配列から第 $p-k$ 要素を再帰的に選択

$p=5$ の場合

2	0	1
---	---	---

$k=3$

3	5	8	3	4
---	---	---	---	---

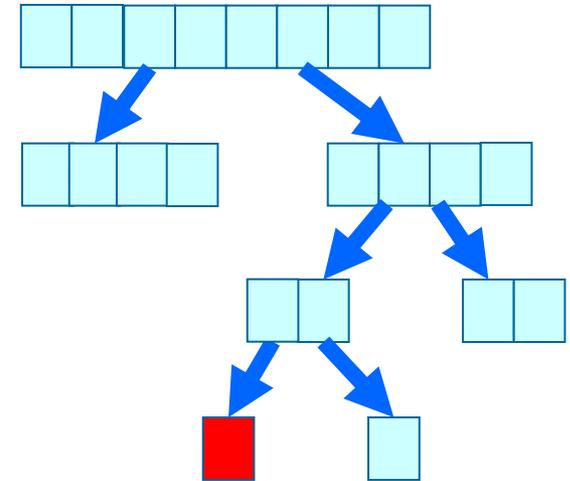
$n-k=5$

$p=5 > 3=k$ なので
右の配列から
第 $5-3=2$ 要素を
再帰的に選択

3

QUICKSELECTの時間計算量 (その1)

- 大きさ n の配列の分割は cn 時間で可能
- 分割後の配列の大きさは
軸要素の選び方に依存



- 運がよい場合: 毎回の分割で
配列の大きさが半分になる
→ $cn + c(n/2) + c(n/4) + \dots + c \cdot 2 + c \cdot 1$
 $\leq cn \times 2 = O(n)$ **線形時間**
- 普通の場合: 毎回の分割で配列の大きさが
 $(1 - \varepsilon)$ 倍以下になる (ε は定数, $0 < \varepsilon < 1$)
→ $cn + c(1 - \varepsilon)n + c(1 - \varepsilon)^2n + c(1 - \varepsilon)^3n$
 $+ \dots + c n / (1 - \varepsilon) + c \cdot 1$
 $\leq cn \times 1 / (1 - \varepsilon) = O(n)$ **線形時間**

QUICKSELECTの時間計算量 (その2)

- 大きさ n の配列の分割は cn 時間で可能

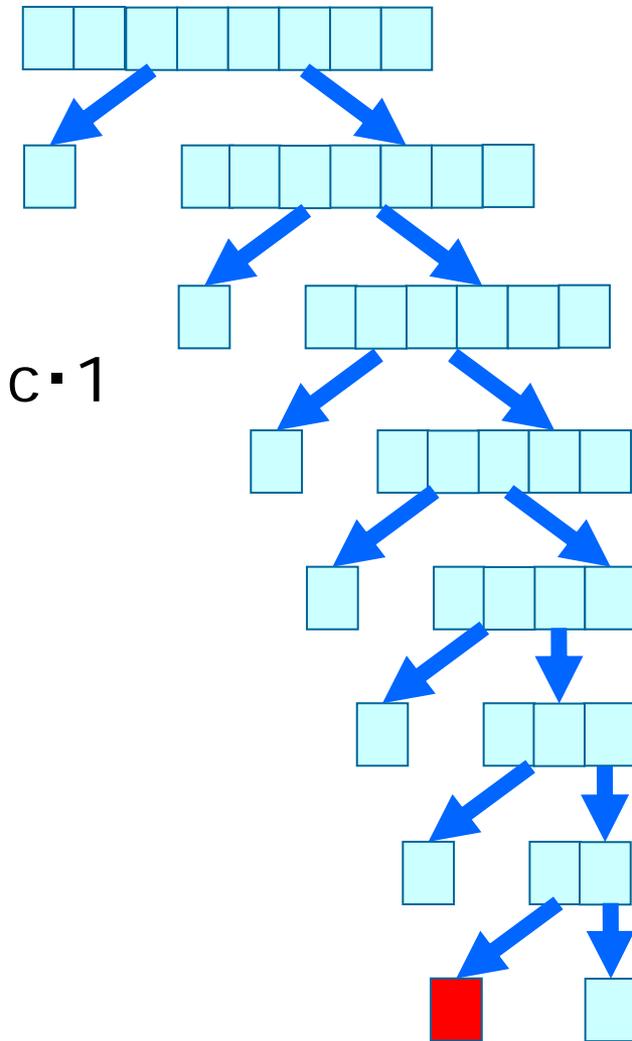
- **最悪の場合 (運が悪い場合):** 毎回の分割で配列の大きさが1ずつ減少

$$\begin{aligned} \rightarrow & cn + c(n-1) + c(n-2) + \dots + c \cdot 2 + c \cdot 1 \\ & \leq cn(n-1)/2 = O(n^2) \end{aligned}$$

- 軸の選び方をランダムにする

$$\rightarrow \text{平均的な時間計算量は } O(n)$$

解析はかなり面倒なので省略



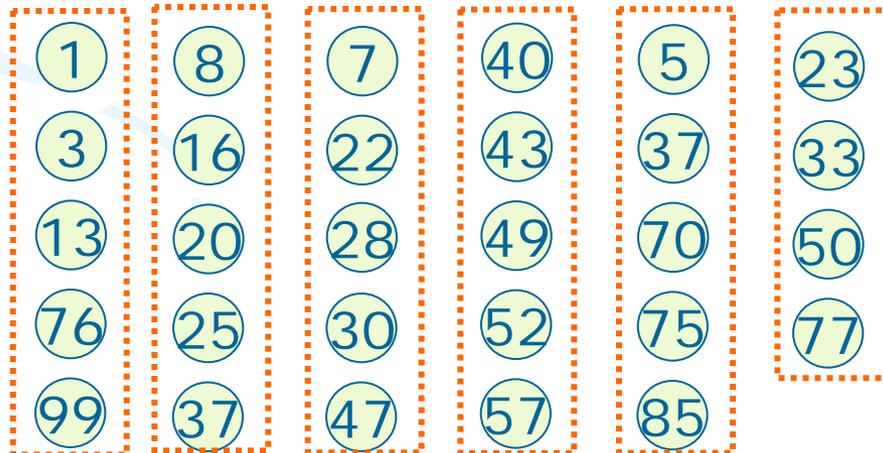
SELECTの概要

- QUICKSELECTで時間計算量が $O(n)$ になるケース
 - 毎回の分割で配列の大きさが $(1 - \varepsilon)$ 倍以下になる
(ε は定数, $0 < \varepsilon < 1$)
- どんな入力に対しても, このケースが起こるように軸要素の選び方を工夫する
 - アルゴリズムSELECT
($\varepsilon = 1/4$)

SELECTでの軸要素の選び方

- 説明を簡単にするため、全ての要素が異なると仮定
 - $A[i]$ と $A[k]$ が等しい場合は、
 - 配列の番号 i, k によって大小関係を決めればよい
($i < k$ ならば $A[i]$ が「小さい」、 $i > k$ ならば $A[k]$ が「小さい」)

手順1: n 個の要素を5個ずつのグループに分ける。
5個未満のグループが一つ出来たら、それは別にしておく。
→ ちょうど5個のグループの数は $\lfloor n/5 \rfloor$



$$n = 29$$

$$\lfloor n/5 \rfloor = 5$$

SELECTでの軸要素の選び方

手順2: $\lfloor n/5 \rfloor$ 個の各グループにおいて、中央値(3番目に小さい値)を計算し、取り出す。

手順3: 手順2で取り出した $\lfloor n/5 \rfloor$ 個の要素に対し、中央値 ($\lceil \lfloor n/5 \rfloor / 2 \rceil = \lfloor (n+5)/10 \rfloor$ 番目に小さい値)を計算し、軸要素 a とする

1	8	7	40	5	23
3	16	22	43	37	33
13	20	28	49	70	50
76	25	30	52	75	77
99	37	47	57	85	

$$n = 29$$

$$\lfloor n/5 \rfloor = 5$$

中央値の集合

13, 49, 70, 28, 20

この中の中央値は

28 ← 軸要素 a

各反復での要素数の減少率

グループ1, 2のそれぞれの中央値は軸要素 a 未満
 →グループ1, 2には a未満の要素が3つ以上存在
 →a 未満の要素数
 $\geq 3 \times 3 \text{グループ} - 1 = 8$

グループ4, 5のそれぞれの中央値は軸要素aより大きい
 →グループ4, 5には aより大きい要素が3つ以上存在
 →a より大きい要素の数
 $\geq 3 \times 3 \text{グループ} - 1 = 8$

グループ1	グループ2	グループ3	グループ4	グループ5	
1	8	7	40	5	23
3	16	22	43	37	33
13	20	28	49	70	50
76	25	30	52	75	77
99	37	47	57	85	

軸要素 a より大きい要素の割合は？
 軸要素 a より小さい要素の割合は？

各反復での要素数の減少率

軸要素 a は, 各グループの中央値の中
 の中央値, $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor (n+5)/10 \rfloor$
 番目に小さい要素

「グループの中央値 $< a$ 」となる
 グループ数は $\lfloor (n+5)/10 \rfloor - 1$
 →各グループには a 未満の要素
 が3つ以上存在

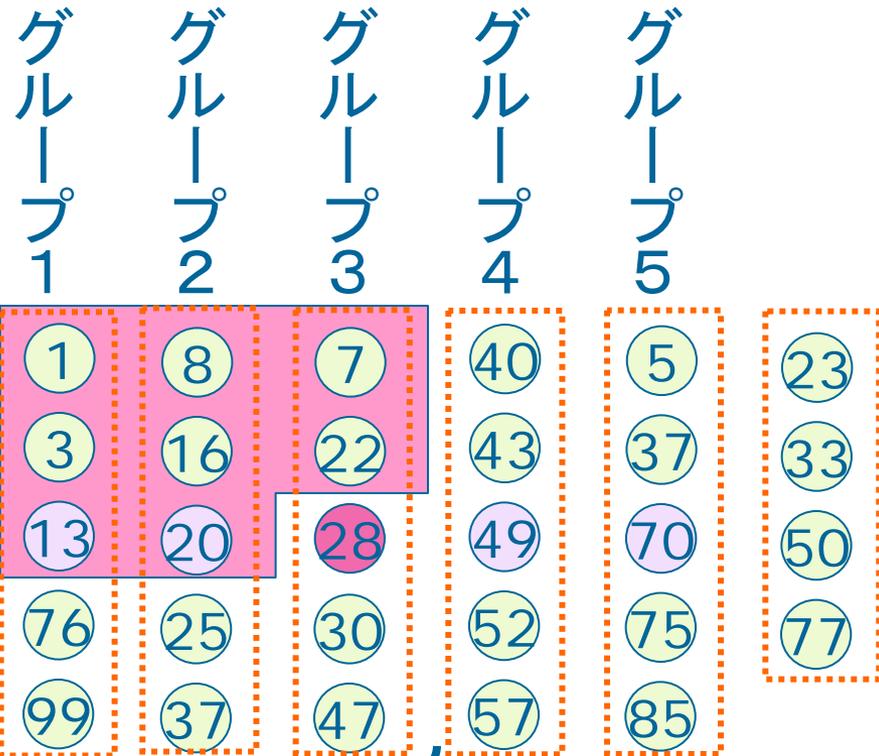
→ a 未満の要素数

$$\geq 3 \times \lfloor (n+5)/10 \rfloor - 1$$

$n \geq 50$ のとき, この値は $n/4$ 以上

$n \geq 50$, かつ見つけたい要素が
 軸要素以上 →

次の反復での要素数 $\leq 3/4n$



$$\lfloor (n+5)/10 \rfloor$$

各反復での時間計算量の解析

軸要素 a は, 各グループの中央値の中の中央値, $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ 番目に大きい要素

「グループの中央値 $> a$ 」となる

グループ数は $\lfloor n/10 \rfloor - 1$

→各グループには a より大きい要素が3つ以上存在

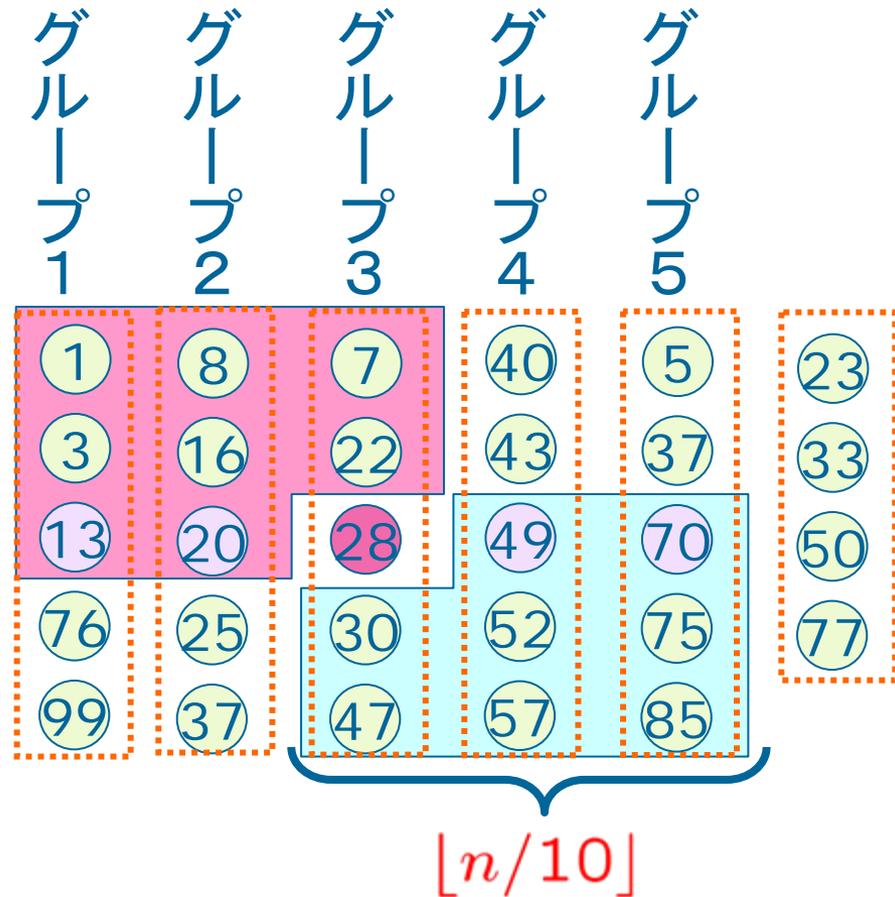
→ a より大きい要素の数

$$\geq 3 \times \lfloor n/10 \rfloor - 1$$

$n \geq 80$ のとき, この値は $n/4$ 以上

$n \geq 80$, かつ見つけたい要素が軸要素未満 →

次の反復での要素数 $\leq 3/4n$



SELECTの時間計算量の解析

- アルゴリズムSELECTでは,
 - $n \geq 80$ のとき, 一回の反復で要素数が n から $3/4n$ 以下に減少する
 - $n < 80$ のとき, 定数時間で第 p 要素を選択することが可能 (要素数 n が定数以下なので)
- 整数 k が $(3/4)^k \geq 80$, $(3/4)^{k+1} < 80$ を満たすとすると, 合計の時間計算量は
$$c\{n + 3/4n + (3/4)^2n + \dots + (3/4)^kn + 80\}$$
$$\leq c \times 4n + 80 \quad c = O(n) \text{ となる}$$

今日のレポート(締切:5/28)

- 次の要素集合に対して第9要素(9番目に小さい要素, $p=9$)を求めたい. QUICKSELECTおよびSELECTを適用したときのアルゴリズムの挙動を詳しく説明せよ. なお, QUICKSELECTでの軸の選び方は, クイックソートの(c)を使うこと.

50, 23, 67, 43, 91, 29, 45, 77, 69, 89,
31, 33, 73, 22, 60, 48, 41, 97, 84, 30

※来週(5/21)は休講です