

アルゴリズムと データ構造

第5回

§ 4.5 ソートアルゴリズムの計算量の下界

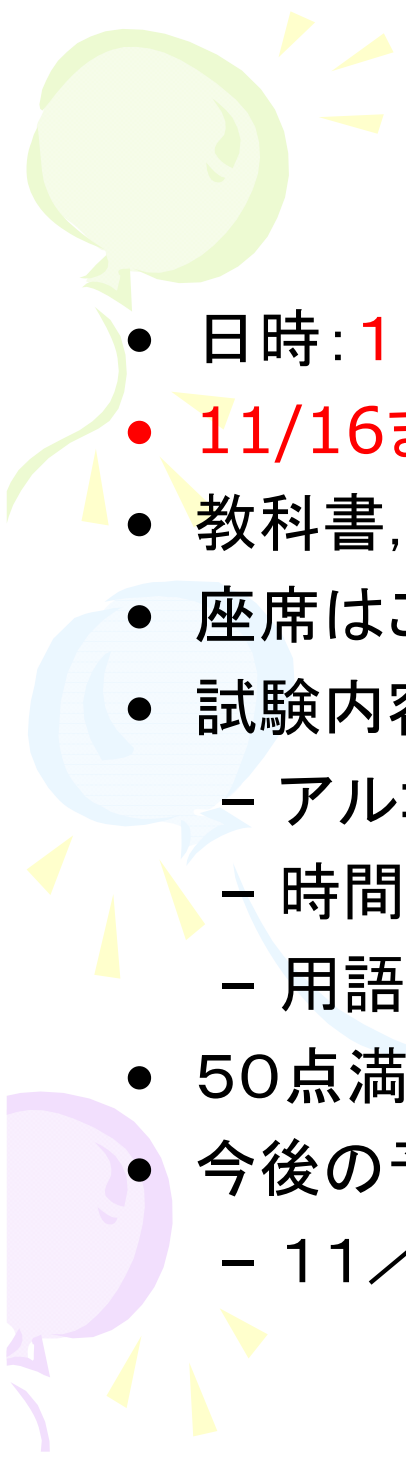
§ 4.6 第p要素選択アルゴリズム

塩浦昭義

情報科学研究科 准教授

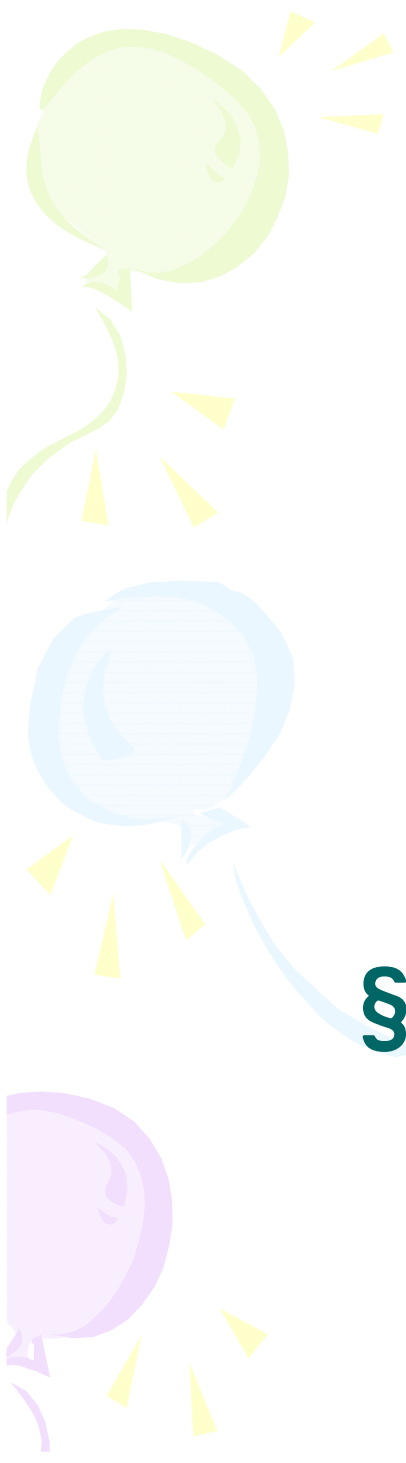
shioura@dais.is.tohoku.ac.jp

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



中間試験について

- 日時: 11月30日(水)13:00~14:30 (予定)
- 11/16までにレポートを一度も出していない場合, 受験不可
- 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: 11/9(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, 29点以下は追試レポートもしくは単位不可
- 今後の予定
 - 11/9: 第6回, 11/16: 第7回, 11/23: 祝日



§ 4.5 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
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
進行状況を**決定木**により表現
 - 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: $A[1] = a, A[2] = b, A[3] = c$ 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
 - ソート結果の候補は
 $abc, acb, bac, bca, cab, cba$ の6通り
 - 比較を行なう度に, ソート結果の候補数が減少

バブルソートの動き(その1)

Behavior of Bubble Sort

1回目の反復

$A[i]$ = 配列の i 番目の要素 ($i = 1, 2, \dots, n$)

■ $A[7]$ と $A[8]$ の大小を比較

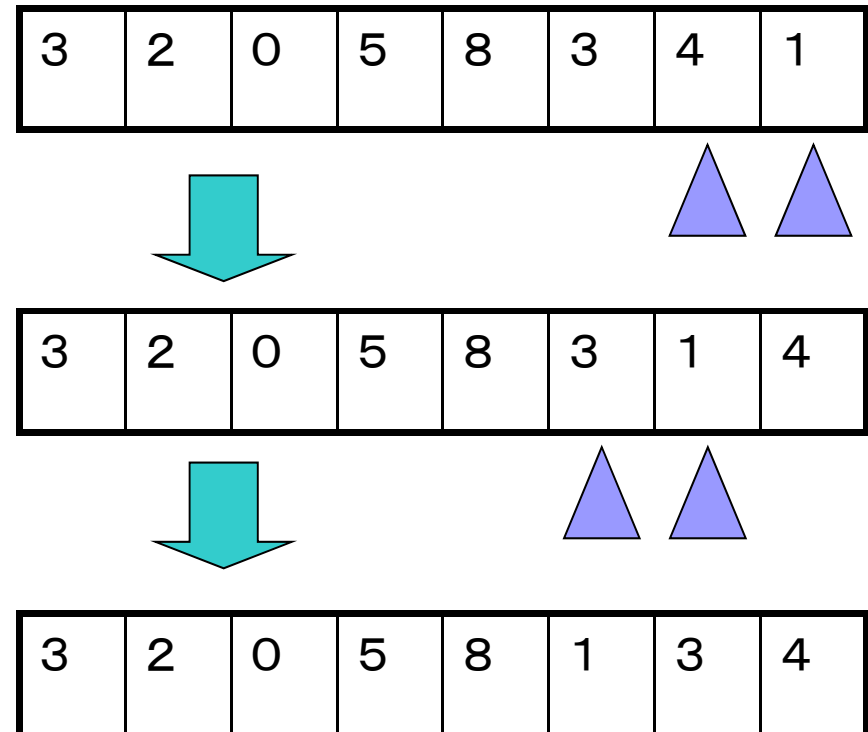
$A[7] > A[8]$

⇒ 2つの要素を入れ替え

■ $A[6]$ と $A[7]$ の大小を比較

$A[6] > A[7]$

⇒ 2つの要素を入れ替え



バブルソートの動き(その2)

■以下、同様に繰り返す

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



入れ替え

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



入れ替え

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



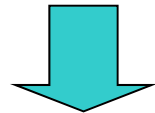
そのまま



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



入れ替え



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



入れ替え



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

A[1], ..., A[8]の
中で最小

1回目の
反復終了

バブルソートの動き(その3)

2回目の反復

$A[2], \dots, A[8]$ に対して1回目の反復と同じ作業を行う

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

全体で2番目に小さい

$A[2], \dots, A[8]$ の中で最小

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

3回目の反復

$A[3], \dots, A[8]$ に対して1回目の反復と同じ作業を行う

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

全体で3番目に小さい

$A[3], \dots, A[8]$ の中で最小

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

バブルソートの動き(その4)

4回目の反復

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



5回目の反復

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



6回目の反復

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



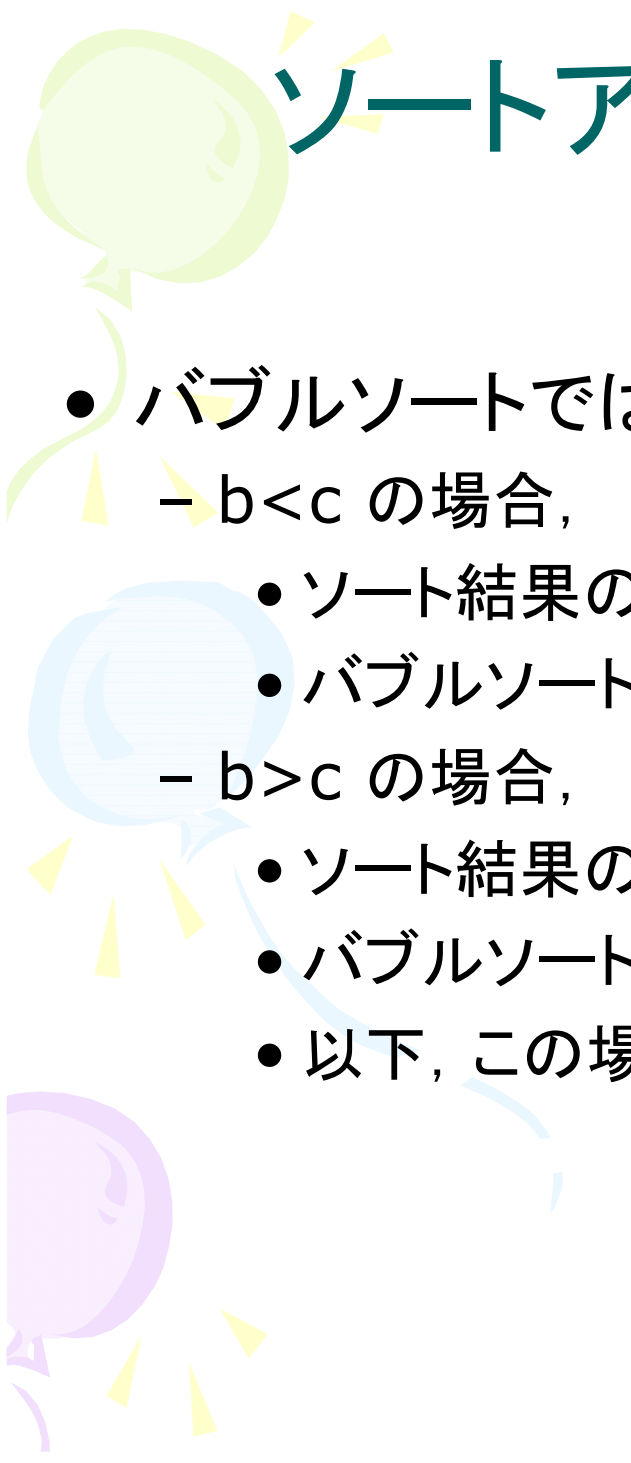
7回目の反復

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



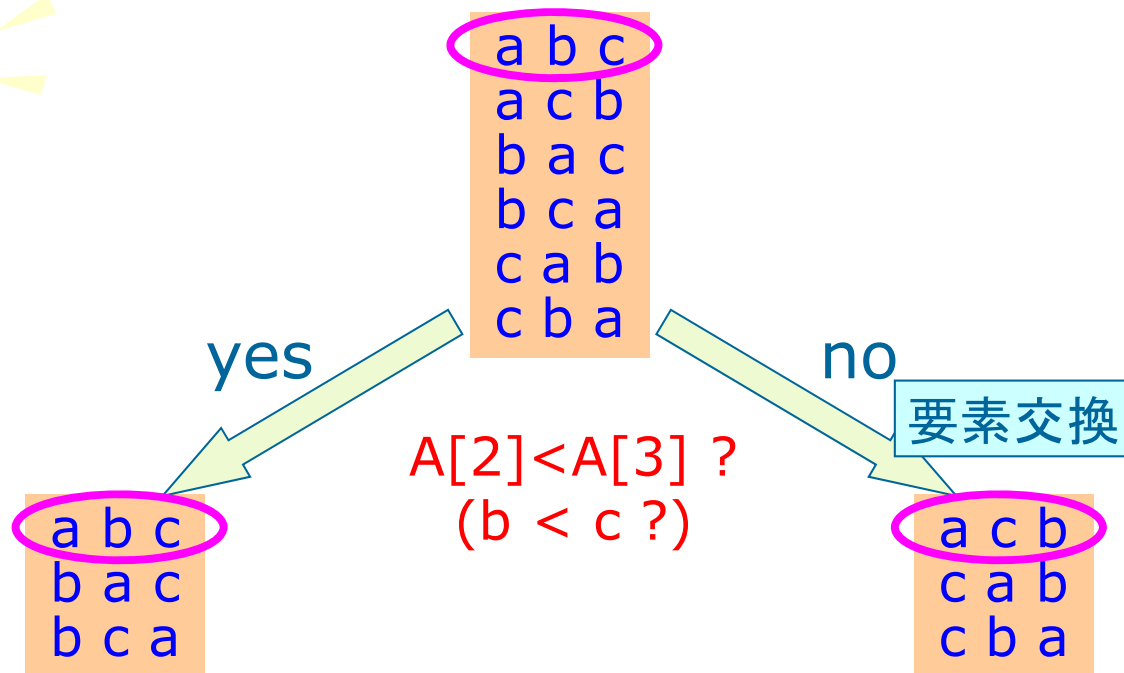
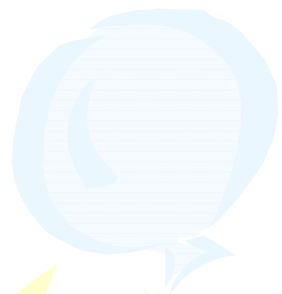
ソート完了！

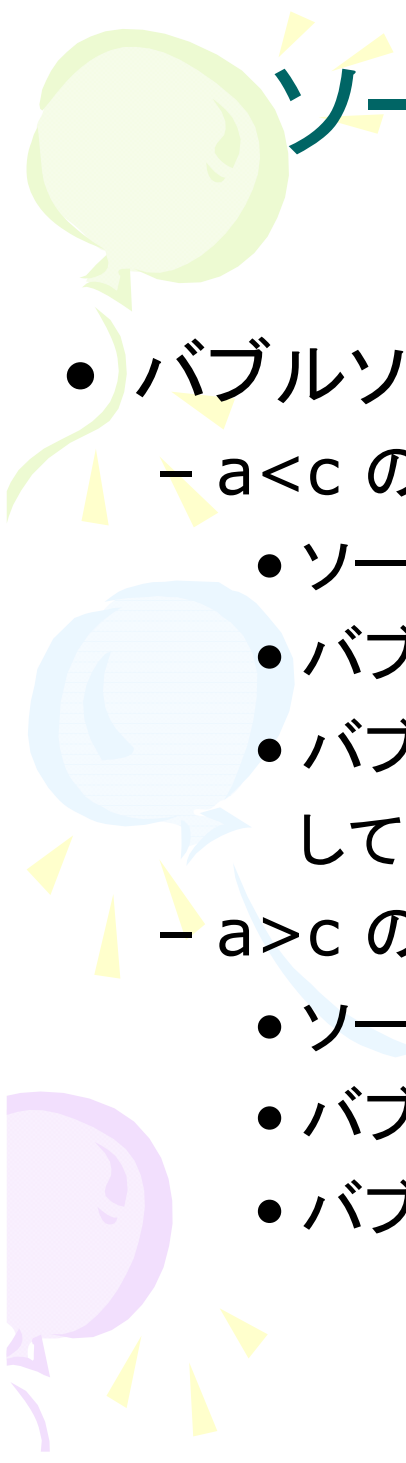
0	1	2	3	3	4	5	8
---	---	---	---	---	---	---	---



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 $A[2]=b$ と $A[3]=c$ を比較
 - $b < c$ の場合,
 - ソート結果の候補は abc, bac, bca 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,
 - ソート結果の候補は acb, cab, cba の3通り
 - バブルソートでは $A[2]$ と $A[3]$ を入れ替え → $A: acb$
 - 以下, この場合を考える

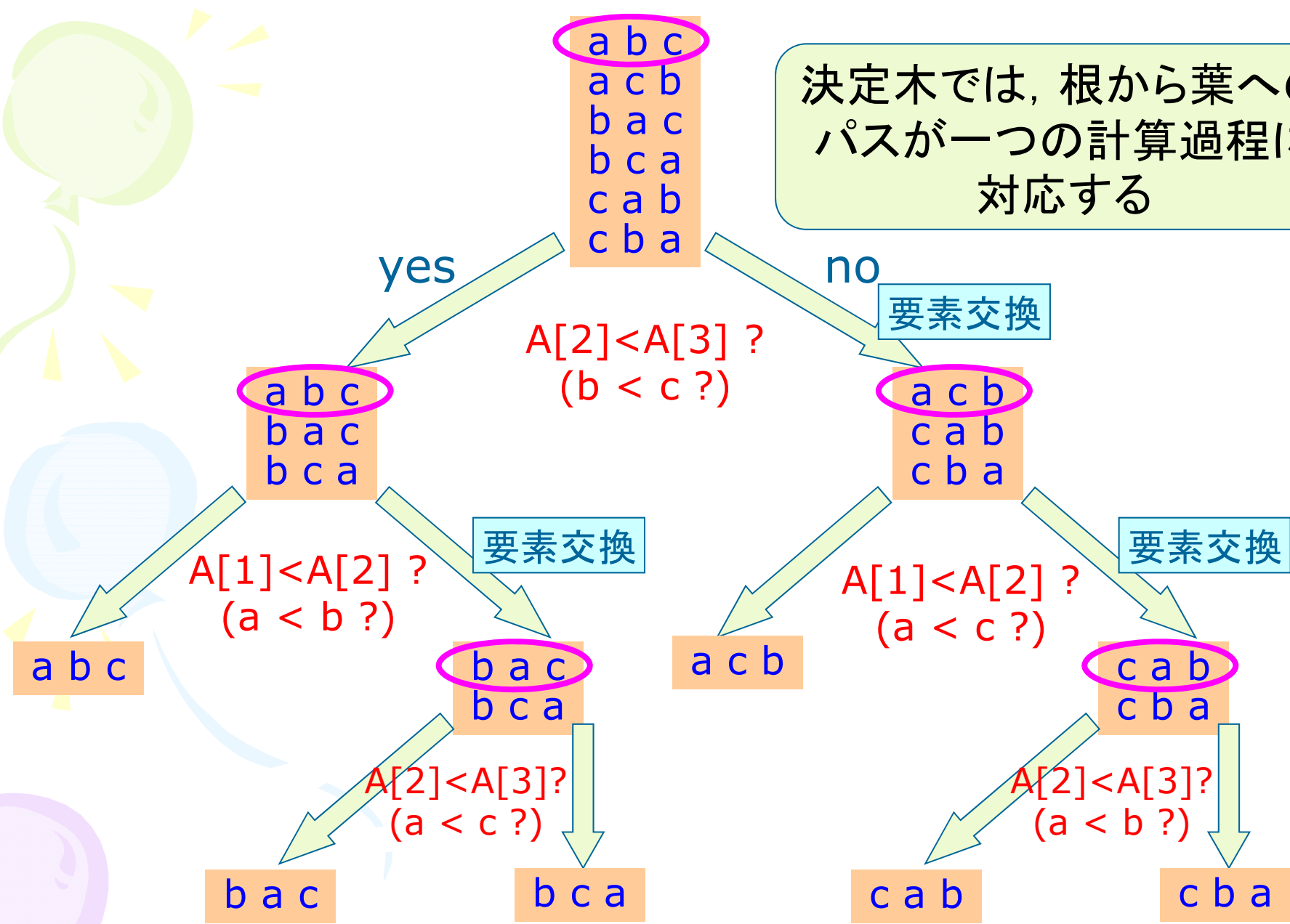




ソートアルゴリズムの決定木 (その3)

- バブルソートでは, 次に $A[1]=a$ と $A[2]=c$ を比較
 - $a < c$ の場合,
 - ソート結果の候補は acb の1通り
 - バブルソートでは要素の入れ替えをしない → $A: acb$
 - バブルソートでは, この後(不要な) $A[2]$ と $A[3]$ の比較をして終了 → $A: acb$ をソート結果として出力
 - $a > c$ の場合,
 - ソート結果の候補は cab, cba の2通り
 - バブルソートでは $A[1]$ と $A[2]$ を入れ替え → $A: cab$
 - バブルソートでは, この後 $A[2]$ と $A[3]$ 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



アルゴリズムは、ソート結果の候補が1つに決まるまで比較を繰り返す

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調べる
- アルゴリズムでの比較の回数の最大値
 $\geq$ 根から葉へのパスの長さの最大値 = 決定木の高さ h

- 決定木の葉は 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 の二分木の葉の数 $\leq 2^h$

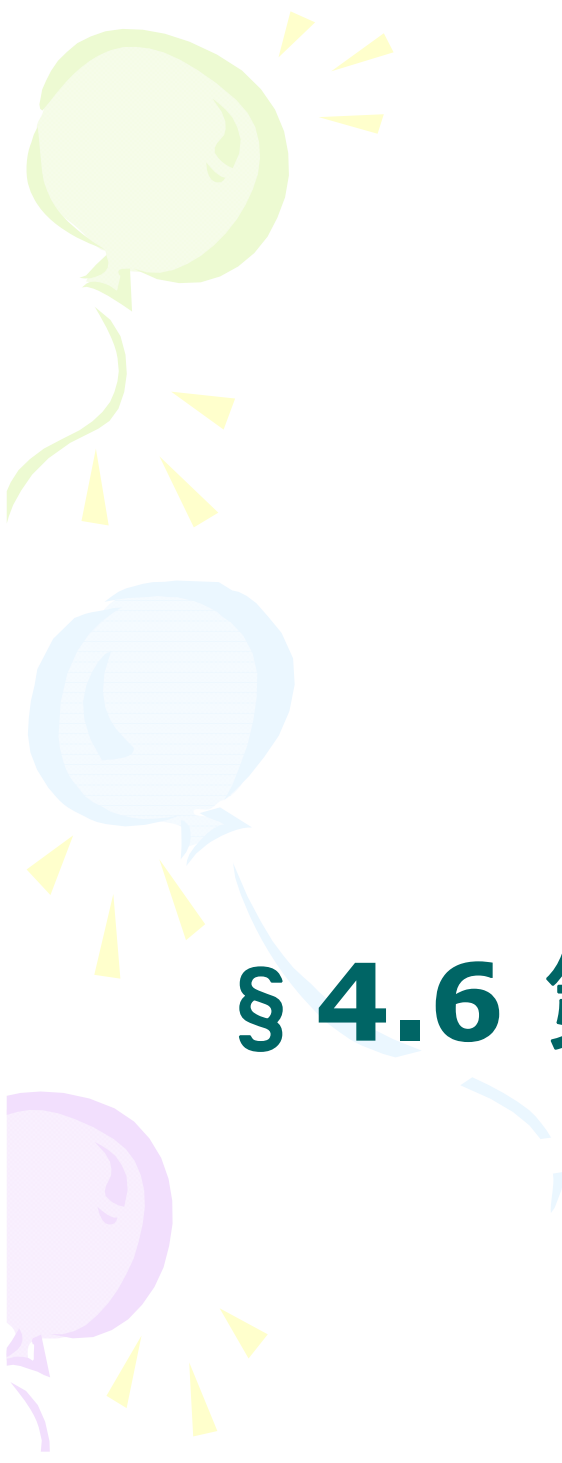
$\rightarrow n! \leq 2^h \rightarrow h \geq \log(n!)$

 – 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)

この値を
知りたい!

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.



§ 4.6 第P要素選択アルゴリズム

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

中央値
(5番目に小さい値)

73 25 69 52 91 37 76 40 88

最大値

2番目に
大きい値

第p要素の選択の時間計算量

- 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 $b = A[1]$ とする
 - 各 $i = 2, 3, \dots, n$ 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
- p番目 ($1 < p < n$) に小さい値を求める際の時間計算量は？
 - 簡単なやり方: n 個の実数を昇順にソート
 - 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) $A[1], \dots, A[n]$ 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) $A[1], \dots, A[n]$ 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 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

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

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

$k=3$

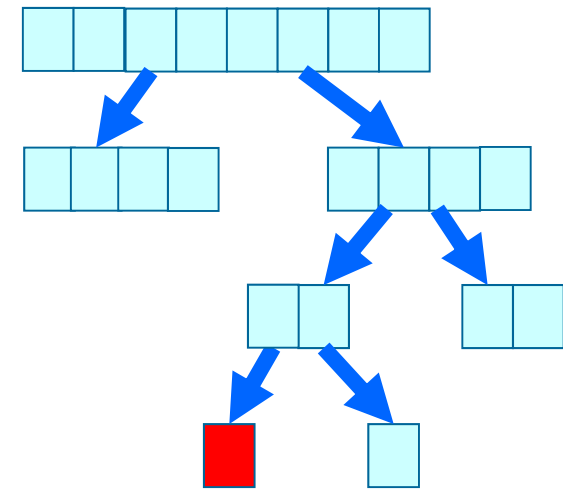
3	5	8	3	4
---	---	---	---	---

$n-k=5$

3

QUICKSELECTの時間計算量 (その1)

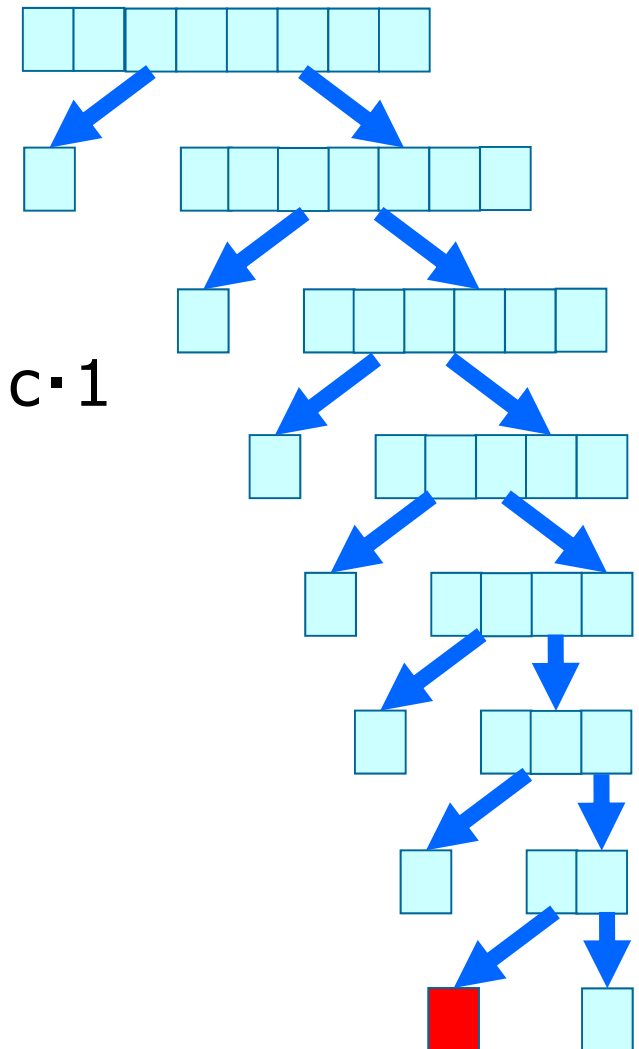
- 大きさ n の配列の分割は cn 時間で可能
- 分割後の配列の大きさは
軸要素の選び方に依存

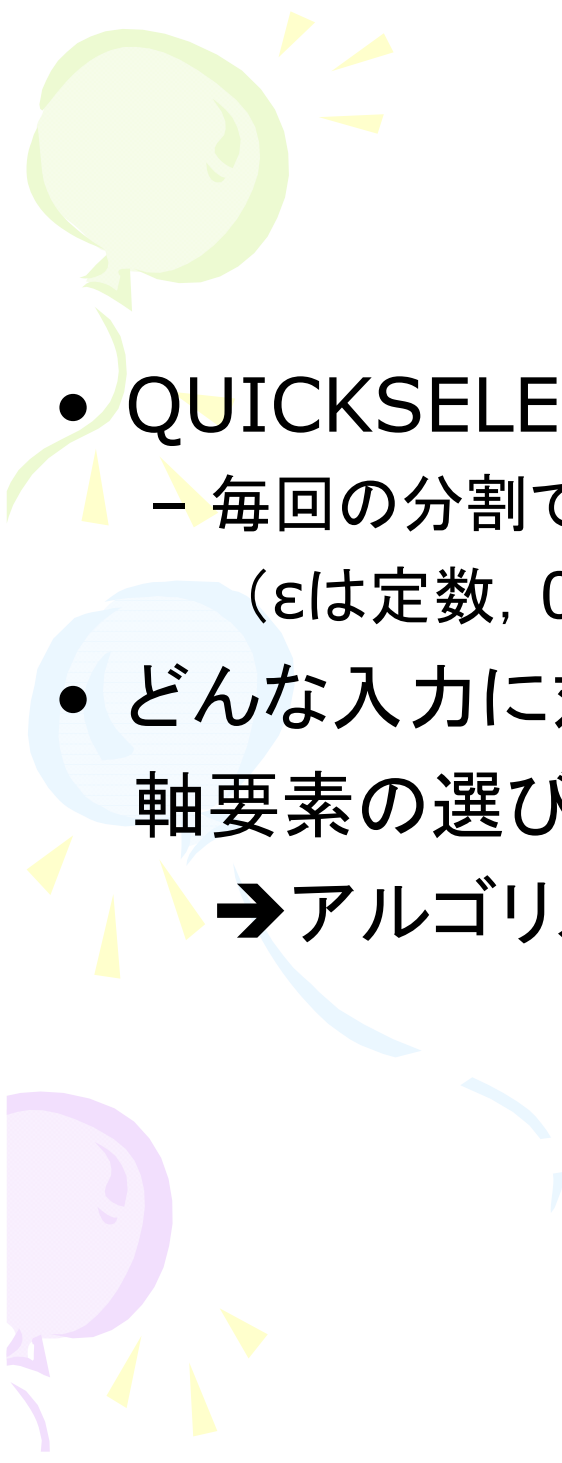


- 運がよい場合: 毎回の分割で
配列の大きさが半分になる
→ $cn + c(n/2) + c(n/4) + \dots + c \cdot 2 + c \cdot 1$
 $\leq cn \times 2 = O(n)$ 線形時間
- 普通の場合: 毎回の分割で配列の大きさが
($1-\epsilon$)倍以下になる(ϵ は定数, $0 < \epsilon < 1$)
→ $cn + c(1-\epsilon)n + c(1-\epsilon)^2n + c(1-\epsilon)^3n$
 $+ \dots + c n/(1-\epsilon) + c \cdot 1$
 $\leq cn \times 1/(1-\epsilon) = O(n)$ 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
→ $cn + c(n-1) + c(n-2) + \dots + c \cdot 2 + c \cdot 1$
 $\leq cn(n-1)/2 = O(n^2)$
- 軸の選び方をランダムにする
→ 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2つの配列の大きさが $(1-\varepsilon)$ 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 $\lfloor n/5 \rfloor$

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	

$$n = 29$$
$$\lfloor n/5 \rfloor = 5$$

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

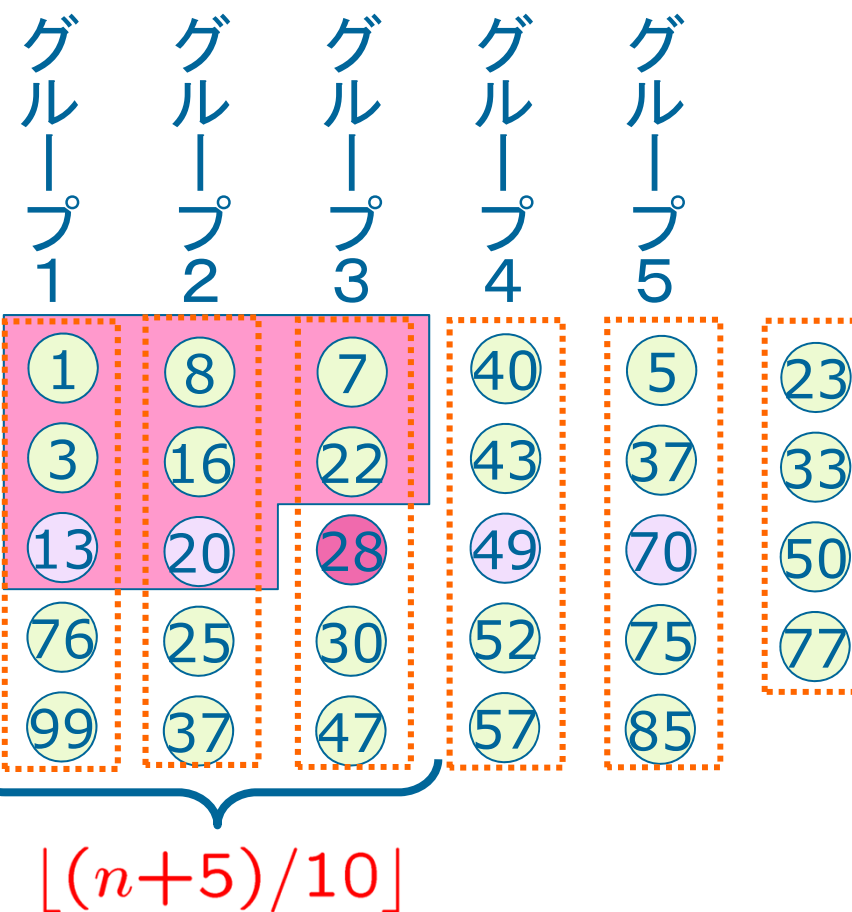
→ a 未満の要素数

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

$n \geq 50$ 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 $\leq 3/4n$



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

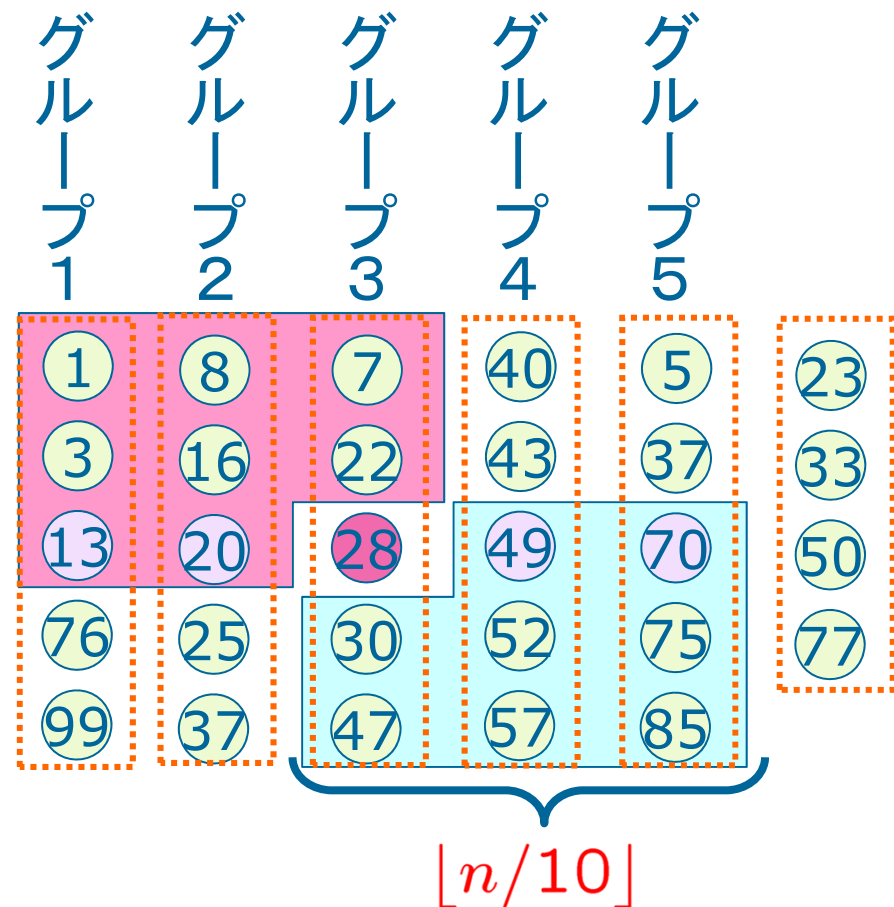
→ a より大きい要素の数

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

$n \geq 80$ 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)
- 反復回数を k とおく
 - $(3/4)^{k-1} \geq 80, (3/4)^k < 80$ を満たす
 - 合計の時間計算量は
$$c\{n + 3/4n + (3/4)^2n + \dots + (3/4)^{k-2}n + 80\}$$
$$\leq c \times 4n + 80 \quad c = O(n)$$



今日のレポート

問1: $A[1]=a$, $A[2]=b$, $A[3]=c$, $A[4]=b$ という4つの要素からなる配列に、バブルソートを適用する.

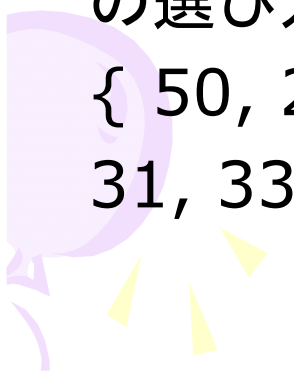
(1)バブルソートで行われる比較の回数 k を求めよ.

(2)バブルソートで行われる k 回の比較において, 配列のどの2つの要素が比較されるか, 説明せよ.

(3)バブルソートの決定木を書け.

問2: 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 }





レポートの解答例: QUICKSELECTを用いた場合

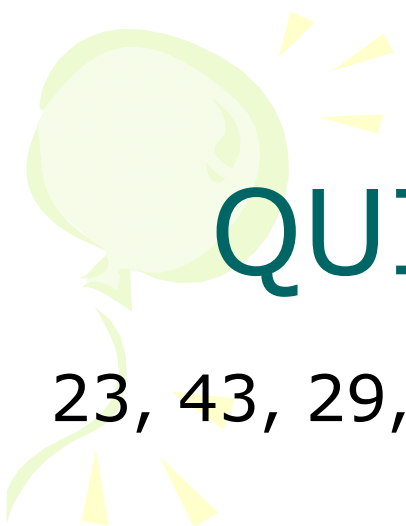
50, 23, 67, 43, 91, 29, 45, 77, 69, 89,
31, 33, 73, 22, 60, 48, 41, 97, 84, 30 (20個の整数)
から9番目に小さい値を求める

- 軸要素は1番目の50と2番目の23のうち, 大きい方→50
- 軸要素により分割

軸要素未満: 23, 43, 29, 45, 31, 33, 22, 48, 41, 30 (10個)

軸要素以上: 50, 67, 91, 77, 69, 89, 73, 60, 97, 84 (10個)

- 軸要素未満の整数の数10個 $\geq$ 9なので, 軸要素未満の整数の中で9番目に小さい値を求めればよい
- 



レポートの解答例: QUICKSELECTを用いた場合

23, 43, 29, 45, 31, 33, 22, 48, 41, 30 (10個)

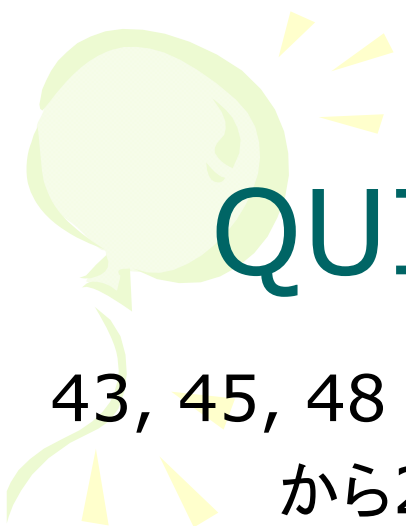
から9番目に小さい値を求める

- 軸要素は1番目の23と2番目の43のうち, 大きい方➡43
- 軸要素により分割

軸要素未満: 23, 29, 31, 33, 22, 41, 30 (7個)

軸要素以上: 43, 45, 48 (3個)

- 軸要素未満の整数の数7個<9なので, 軸要素以上の整数の中で
 $9 - 7 = 2$ 番目に小さい値を求めればよい
- 



レポートの解答例: QUICKSELECTを用いた場合

43, 45, 48 (3個)

から2番目に小さい値を求める

- 軸要素は1番目の43と2番目の45のうち, 大きい方➡45
- 軸要素により分割

軸要素未満: 43(1個)

軸要素以上: 45, 48 (2個)

- 軸要素未満の整数の数1個<2なので, 軸要素以上の整数の中で
 $2 - 1 = 1$ 番目に小さい値を求めればよい





レポートの解答例: QUICKSELECTを用いた場合

45, 48 (2個)から1番目に小さい値を求める

- 軸要素は1番目の45と2番目の48のうち, 大きい方→48
- 軸要素により分割

軸要素未満: 45(1個)

軸要素以上: 48(1個)

- 軸要素未満の整数の数1個 ≥ 1 なので, 軸要素以下の整数の中で1番目に小さい値を求めればよい

45 (1個)から1番目に小さい値を求める

- 要素数は1→45を出力 (終了)
- 



レポートの解答例: SELECTを用いた場合

50, 23, 67, 43, 91, 29, 45, 77, 69, 89,
31, 33, 73, 22, 60, 48, 41, 97, 84, 30 (20個の整数)
から9番目に小さい値を求める

- 5個ずつのグループに分割(4グループ)

50, 23, 67, 43, 91 29, 45, 77, 69, 89,

31, 33, 73, 22, 60 48, 41, 97, 84, 30

- それぞれのグループから中央値(3番目に小さい値)を選ぶ

50 69 33 48

- 上記の4つの整数から, 中央値(2番目に小さい値)を選ぶ→48
 - これを軸要素として分割を行なう
- 



レポートの解答例: SELECTを用いた場合

50, 23, 67, 43, 91, 29, 45, 77, 69, 89,
31, 33, 73, 22, 60, 48, 41, 97, 84, 30 (20個の整数)
から9番目に小さい値を求める

- 軸要素48により分割

軸要素未満: 23, 43, 29, 45, 31, 33, 22, 41, 30 (9個)

軸要素以上: 50, 67, 91, 77, 69, 89, 73, 60, 48, 97, 84 (11個)

- 軸要素未満の整数の数9個 $\geq$ 9なので, 軸要素未満の整数の中で9番目に小さい値を求めればよい
- 



レポートの解答例: SELECTを用いた場合

23, 43, 29, 45, 31, 33, 22, 41, 30 (9個の整数)
から9番目に小さい値を求める

- 5個ずつのグループに分割(1グループとあまり4個)

23, 43, 29, 45, 31 33, 22, 41, 30

- それぞれのグループから中央値(3番目に小さい値)を選ぶ
31

- 上記の1つの整数から, 中央値(1番目に小さい値)を選ぶ→31

- これを軸要素として分割を行なう

- 軸要素31により分割

軸要素未満: 23, 29, 22, 30 (4個)

軸要素以上: 43, 45, 31, 33, 41 (5個)

- 軸要素未満の整数の数4個<9なので, 軸要素以上の整数の中で
 $9-4=5$ 番目に小さい値を求めればよい



レポートの解答例: SELECTを用いた場合

43, 45, 31, 33, 41 (5個の整数) から5番目に小さい値を求める

- 5個ずつのグループに分割(1グループ)

43, 45, 31, 33, 41

- それぞれのグループから中央値(3番目に小さい値)を選ぶ

41

- 上記の1つの整数から, 中央値(1番目に小さい値)を選ぶ→41

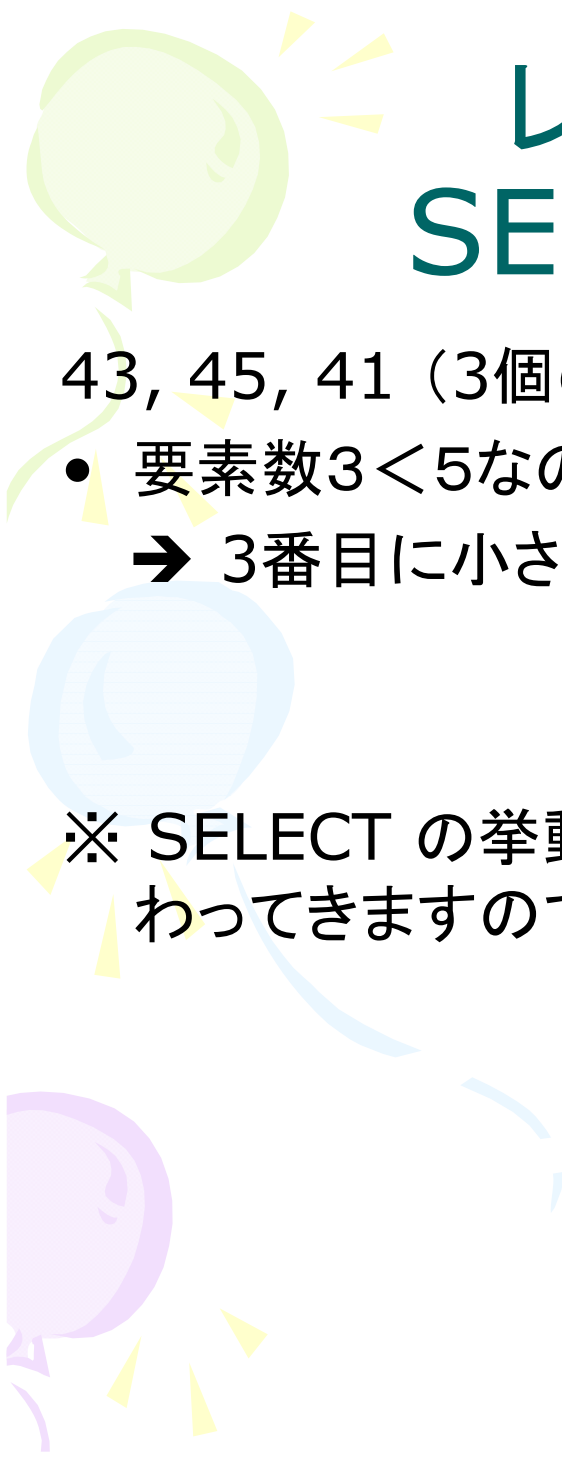
- これを軸要素として分割を行なう

- 軸要素41により分割

軸要素未満: 31, 33 (2個)

軸要素以上: 43, 45, 41 (3個)

- 軸要素未満の整数の数2個<5なので, 軸要素以上の整数の中で
5-2=3番目に小さい値を求めればよい



レポートの解答例: SELECTを用いた場合

43, 45, 41 (3個の整数) から3番目に小さい値を求める

- 要素数 $3 < 5$ なので, グループ分けが出来ない
→ 3番目に小さい値を計算して, 出力 → 45 (終了)

※ SELECT の挙動は, 各反復での整数のグループ分けによって変わってきますので, こ個で示した例と違う挙動になることもあります.