並び替え（整列）
sort, sorting

selection sort
bubble sort
insertion sort

1

sorting とは

与えられた n 個のデータを「大きさの順」に並び替えること

アルゴリズム的には，

・データ $a_1, a_2, \ldots, a_n$ は数値 or 文字

・昇順 accending order に並び替える

という問題を考えれば十分

- 実際のプログラムでは，データは大きさ n の配列 a[1], a[2],..., a[n] に格納
- 昇順 <-> 降順 descending order

2

sorting の種々のアルゴリズム

- 最小値選択法（直接選択法）
  selection sort (straight selection)
- バブルソート bubble sort
- 挿入ソート insertion sort
- ヒープソート heap sort
- クイックソート quick sort

etc.

3

selection sort

考え方

データ列の最小値を探して，順番に 1 番目，2 番目，．．．に置いていく

手順

[1] $\min\{ a_1, a_2, \ldots, a_n \}$ と a_1 とを交換
[2] $\min\{ a_2, a_3, \ldots, a_n \}$ と a_2 とを交換
・・・・
[n-1] $\min\{ a_{n-1}, a_n \}$ と a_{n-1} とを交換

4

selection sort

アルゴリズム

```
for(i=1; i<=n-1; i=i+1){
  min ← i ;
  for(j=i+1; j<=n; j=j+1){
    if (aj < amin){
      min ← j;
    }
  }
  swap(ai, amin);
}
```

説明

5

selection sort（例）

	i	j	min	a_1	a_2	a_3	a_4
start				12	14	11	13
1	1	2	1				
2		3	3				
3		4		11	14	12	13
4	2	3	3				
5		4		11	12	14	13
6	3	4	4	11	12	13	14

6

selection sort の計算量

比較回数によって，計算量を計測することにする。

i=1, 2, … , n-1 のそれぞれについて，
j=i+1, i+2, … , n の n-i 回の比較を行う

↓

総比較回数は，$\sum_{i=1}^{n-1}(n-i)=\frac{n(n-1)}{2}$

よって，計算のオーダーは，O(n^2)

課題：selection sort

次の操作を JS で書き，ファイルを提出せよ。

1. 誕生日 yyyymmdd と学籍番号 ???????? をそれぞれ2桁ずつに分割し,それを a[1], a[2],…, a[8] とする。（今後，このデータを「いつものデータ」と呼ぶことにする。）
2. a[1], a[2],…, a[8] を出力する。
3. selection sort を実行する。（option：実行中に，別紙の表の内容を出力せよ。）
4. a[1], a[2],…, a[8] を出力する。

＊ 提出する際のファイル名＝selection.html

bubble sort

考え方

隣接する２項を比較し，「$a_i>a_{i+1}$ ならば交換」を，左から右に操作する。

手順

- $a_1, a_2, \ldots, a_n$ に対して左から右に交換操作をする
  →a_n が，$a_1, a_2, \ldots, a_n$ の中での最大値となる。
- $a_1, a_2, \ldots, a_{n-1}$ に対して左から右に交換操作をする
  →a_{n-1} が，$a_1, a_2, \ldots, a_{n-1}$ の中での最大値となる。
  ・・・・
- a_1, a_2 に対して交換操作をする→$a_1<=a_2$ となる。

bubble sort

アルゴリズム

```
for(i=n; i>=2; i=i-1){
  for(j=1; j<=i-1; j=j+1){
    if (aj > aj+1){
      swap(aj , aj+1);
    }
  }
}
```

説明

bubble sort（例）

	i	j	swap?	a_1	a_2	a_3	a_4
start				12	14	11	13
1	4	1					
2		2	yes		11	14	
3		3	yes	12	11	13	14
4	3	1	yes	11	12		
5		2		11	12	13	14
6	2	1		11	12	13	14

bubble sort の計算量

比較回数

i=n, n-1, … , 2 のそれぞれについて，
j=1, 2, … , i-1 の i-1 回の比較を行う。
よって，総比較回数は，$\sum_{i=2}^{n}(i-1)=\frac{n(n-1)}{2}$
故に，比較回数のオーダーは，O(n^2)

交換回数

最良の場合，0 回
最悪の場合，n(n-1)/2 回
→データによって，大きく異なる。

課題：bubble sort

次の操作を JS で書き，ファイルを提出せよ。

1. いつものデータを用意する。
2. a[1], a[2],…, a[8] を出力する。
3. bubble sort を実行する。（option：実行中に，別紙の表の内容を出力せよ。）
4. a[1], a[2],…, a[8] を出力する。

* 提出する際のファイル名＝bubble.html

bubble sort の改良（例）

考え方

いずれかの i において交換が全く行われないとすると，その段階で整列済み→打ち切り

手順

- 「整列済みか否か」を表す変数 flag を用意する。（flag=1 は整列済み。flag=0 は未整列。）
- それぞれの i について
  - flag=1 とする（初期化）
  - j を動かしていくとき，交換が起これば，flag=0 とする。
  - flag=1 ならば，一度も交換が起こらなかったのだから，整列済み。よって，打ち切り。

bubble sort の改良（例）

アルゴリズム

```
for(i=n; i>=2; i=i-1){
  flag ← 1;
  for(j=1; j<=i-1; j=j+1){
    if (aj > aj+1){
      swap(aj , aj+1);
      flag ← 0;
    }
  }
  if (flag=1){
    break;
    }
}
```

説明

bubble sort（改）（例）

	i	j	flag	a_1	a_2	a_3	a_4
start				12	14	11	13
1	4	1	1				
2		2	0		11	14	
3		3	0	12	11	13	14
4	3	1	0	11	12		
5		2	0	11	12	13	14
6	2	1	1	11	12	13	14

bubble sort（改）（例）

	i	j	flag	a_1	a_2	a_3	a_4
start				11	12	13	14
1	4	1	1				
2		2					
3		3		11	12	13	14
4							
5							
6							

bubble sort（改）の計算量

比較回数

最良の場合，n-1 回 → O(n)

最悪の場合，n(n-1)/2 回 → $O(n^2)$

→データによって，大きく異なる。

交換回数

最良の場合，0 回 → O(1)

最悪の場合，n(n-1)/2 回 → $O(n^2)$

→データによって，大きく異なる。

課題：bubble sort（改）

次の操作を JS で書き，ファイルを提出せよ。

1. いつものデータを用意する。
2. a[1], a[2],…, a[8] を出力する。
3. bubble sort（改） を実行する。（option：実行中に，別紙の表の内容を出力せよ。）
4. a[1], a[2],…, a[8] を出力する。

* 提出する際のファイル名＝bubble_kai.html

insertion sort

考え方

a_1〜a_{i-1} までが既に整列しているとき，a_i (i=2,3,...,n) を，a_1〜a_{i-1} の間の「しかるべき位置」に「挿入」していく。

しかるべき位置

$a_1 \leqq a_2 \leqq \ldots \leqq a_{k-1} \leqq a_i \leqq a_k \leqq \ldots \leqq a_{i-1}$ のとき，a_{k-1} とa_kの間（k 番目）

a_i を k 番目 ($2 \leqq k \leqq i-1$) に挿入する手順

- a_i の値を覚えておく
- j = i-1, i-2, ... , k について，a_j を a_{j+1} に移動
- a_k に先程覚えておいた値を代入

insertion sort

アルゴリズム

```
for(i=2; i<=n; i=i+1){
  w ← ai;
  j ← i;
  while(aj-1 > w and j > 1){
    aj ← aj-1;
    j ← j-1;
  }
  aj ← w;
}
```

説明

insertion sort（例）

	i	j	w	a_{j-1}>w?	a_1	a_2	a_3	a_4
start					12	14	11	13
1	2	2	14	no				
2					12	14	11	13
3	3	3	11	yes			14	
4		2		yes		12		
5					11	12	14	13
6	4	4	13	yes				14
7		3		no				
8					11	12	13	14

insertion sort の計算量

while 文の比較の回数

少なくとも n-1回（i=2,...,n)

while 文の中身の実行回数

- 最良（昇順）の場合，0 回
- 最悪（降順）の場合，各 i について，j=i, i-1,...,2 の (i-1) 回だから，合計すると $\sum_{i=2}^{n}(i-1) = \frac{n(n-1)}{2}$

insertion sort の計算量（上の２つの合計）

- 最良（昇順）の場合，$n-1 = O(n)$
- 最悪（降順）の場合，$(n-1) + \frac{n(n-1)}{2} = O(n^2)$

insertion sort の平均計算量

while 文の中身の実行回数

仮定：各 i について，次のケースが等確率(=1/i)で起こる：

ケース	反復回数
$a_i < a_1$	i-1
$a_1 <= a_i < a_2$	i-2
・・・	・・・
$a_{i-2} <= a_i < a_{i-1}$	1
$a_{i-1} <= a_i$	0

このとき，各 i について，while 文の中身の平均反復回数は，

$$\sum_{k=0}^{i-1} \frac{k}{i} = \frac{i-1}{2}$$

insertion sort の平均計算量

while 文の中身の実行回数（続き）

よって，すべての i についての while 文の中身の平均反復回数の合計は，

$$\sum_{i=2}^{n} \frac{i-1}{2} = \frac{n(n-1)}{4}$$

insertion sort の平均計算量（while 文の比較の回数と，中身の平均反復回数の合計）

$$(n-1) + \frac{n(n-1)}{4} = O(n^2)$$

25

注

教科書には，insertion sort への補足として，

a_0 = 想定されるすべてのデータよりも小さなデータ

とおいて，これを「番兵 sentinel」として利用する方法が紹介されてる（p.31）

↓

データの特殊性を利用することになり，アルゴリズムの一般性を失うことになるので，余り使用しないほうがよい（使用するときには，慎重に利用すること！）

26

課題：insertion sort

次の操作を JS で書き，ファイルを提出せよ。

1. いつものデータを用意する。
2. a[1], a[2],…, a[8] を出力する。
3. insertion sort を実行する。（option：実行中に，別紙の表の内容を出力せよ。）
4. a[1], a[2],…, a[8] を出力する。

* 提出する際のファイル名＝insertion.html

27

課題：insertion sort

次の操作を JS で書き，ファイルを提出せよ。

1. いつものデータを用意する。
2. a[1], a[2],…, a[8] を出力する。
3. sentinel として使う値を求め，a[0] に代入する。
4. sentinel を使った insertion sort を実行する。（option：実行中に，別紙の表の内容を出力せよ。）
5. a[1], a[2],…, a[8] を出力する。

* 提出する際のファイル名＝sentinel.html

28