コンピュータ

北海道大学理学部数学科

1　並べ替え (ソート)

配列に数値が格納されているとする。これを昇順あるいは降順に並べ替えるにはどの程度の計算量を必要とするだろうか。

必要な処理は次の二通りである。これらの実行回数を減らすことが目的である。

1. 配列要素の間の大小比較の回数
2. 配列要素の値の交換回数

1.1　準備

まず、配列を用意する。ランダムな浮動小数点数を得るには `rand()` 関数を使う。

```
for k=[1:10]
  a(k)=rand();
endfor
```

あるいは、`a=rand(1,N);` としてもよい。`rand(M,N)` で $M \times N$ 配列の各要素にランダムな値を格納する。

2　単純なアルゴリズム

1. 先頭の配列要素と、それ以降の配列要素を順次比較する。
2. 先頭の配列要素の方が大きい場合、値を入れ替える。
3. 結果として、先頭の配列要素には配列中の最小値が格納される。

$\underline{4}$	$\underline{2}$	3	1	5
$\underline{2}$	4	$\underline{3}$	1	5
$\underline{2}$	4	3	$\underline{1}$	5
$\underline{1}$	4	3	2	$\underline{5}$

以下、二番目の配列要素、

1	$\underline{4}$	$\underline{3}$	2	5
1	$\underline{3}$	4	$\underline{2}$	5
1	$\underline{2}$	4	3	$\underline{5}$

三番目の配列要素と繰り返し、全体の並べ替えが終了する。

1	2	$\underline{4}$	$\underline{3}$	5
1	2	3	4	$\underline{5}$

このアルゴリズムは比較回数、配列要素の交換回数ともに $O(n^2)$ となる。

2.1 プログラム例

```
clear;
N=10;
for k=[1:N]
  a(k)=rand();
  printf( "%f ", a(k));
endfor
printf("\n");

for k=[1:N-1]
  for l=[k+1:N]
    if( a(k) > a(l) )
       # a(l) と a(k) を入れ替える
       s=a(l); a(l)=a(k); a(k)=s;
    endif
  endfor
endfor

for k=[1:N]
  printf( "%f ", a(k));
endfor
```

```
printf("\n");
```

2.2 アルゴリズムの改良

配列要素の交換回数を減らすことにする。先のアルゴリズムと同様に比較を続け、配列要素を最小にする添字の値を変数 pos へ記録していく。比較を終えたら最後に先頭の配列要素と位置 pos の要素とを入れ替える。

4	$\underline{2}$	3	1	5	a(pos) と a(2) を比較、$pos = 2$
4	2	$\underline{3}$	1	5	a(pos) と a(3) を比較、$pos = 2$ のまま
4	2	3	$\underline{1}$	5	a(pos) と a(4) を比較、$pos = 4$
4	2	3	1	$\underline{5}$	a(pos) と a(5) を比較、$pos = 4$ のまま
1	2	3	4	5	交換の実行

比較回数は $O(n^2)$ だが、入れ替える回数は $O(n)$ になる。

2.3 プログラム例

```
clear;
N=10;
for k=[1:N]
  a(k)=rand();
  printf( "%f ", a(k));
endfor
printf("\n");

for k=[1:N-1]
  pos=k;
  for l=[k+1:N]
    if(___________ )
      ___________;
    endif
  endfor
  if( pos != k)
     s=a(pos); a(pos)=a(k); a(k)=s;
  endif
endfor
```

```
for k=[1:N]
  printf( "%f ", a(k));
endfor
printf("\n");
```

2.4 補足

ソートのアルゴリズムは多様である。ここでは比較回数の改善に手をつけなかったが、クイックソートなどのアルゴリズムは比較回数を劇的に改善することが知られている。

また、octave では sort 関数が用意されており、`sort(a)` と実行することで配列 a をソートできる。実際にプログラムを組む際はこれを使えば良い。

3 探索

数値が格納されている配列の要素のうち、x を越えない要素の数はどれだけかという問題を考える。$x < a(m)$ かつ $x > a(m-1)$ となるような m を求める問題である。

配列がランダムに与えられていれば全ての要素を探索しなければならないから $O(n)$ の計算量を必要とする。並べ替えが済んでいれば次のように $O(\log n)$ 程度で済む。

3.1 二分法による探索

まず、配列の中央の要素を x と比較する。x との比較結果に応じて、m が左右どちらの区間にあるかを判定し、m を含む区間の幅を半分にする。これを繰り返し、区間の幅が 1 になれば m は定まる。

$N = 16$,$x = 13$ としよう。

1	2	3	4	5	6	7	<u>8</u>	9	10	11	12	13	14	15	16

$8 < 13$ だから、考える区間を右側とし、区間の幅を 1/2 にする。

1	2	3	4	5	6	7	8	9	10	11	<u>12</u>	13	14	15	16

$12 < 13$ だから、考える区間はさらに右側である。

1	2	3	4	5	6	7	8	9	10	11	<u>12</u>	13	<u>14</u>	15	16

$14 > 13$ だから、考える区間は左側である。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

幅が 1 になったので、終了する。

配列のサイズが 2 のべき乗であれば上のように簡単だが、そうでない場合を考慮して次のようなプログラムになる。

3.2 プログラム例

```
#配列 a とサイズ N は与えられているものとする
x=0.5;

pos=ceil(N/2);     # 開始位置は中央
width=ceil(N/2);   # 幅の初期値は N/2

while( width > 1 && pos > 0 && pos <= N )
  if( a(pos) > x )
    width=ceil(width/2);
    pos=pos-width;
  else
    width=ceil(width/2);
    pos=pos+width;
  endif
  printf("pos=%d\n",pos);
endwhile
```

4 課題

1. プログラム 2.1 を、実行結果が逆順になるように変更して実行する。
2. プログラム 2.3 を実行する。
3. プログラム 3.2 を実行する。

```
octave> a
a =
  0.084545
  0.120949
```

```
  0.265265
  0.279845
  0.290658
  0.306255
  0.323794
  0.423783
  0.439643
  0.441726
  0.457665
  0.508502
  0.518225
  0.726694
  0.841195
  0.937261
  0.945928
  0.976772
  0.996304

octave> source "binsearch.m"
pos=15
pos=12
pos=10
pos=11
octave>
```

```
octave> a
a =
  0.071477
  0.094077
  0.144646
  0.244630
  0.290444
  0.303339
  0.422668
  0.509879
  0.628741
  0.678464
  0.708614
  0.826093
  0.844383
  0.897644
  0.933116
  0.943971
  0.981363

octave> source "binsearch.m"
pos=4
pos=7
pos=9
pos=8
octave>
```