ソートと探索

ソート 与えられた配列の各要素を順番に並べ替える操作。

探索 与えられた配列の各要素から必要な要素を抽出する操作。あらかじめソートされていると効率がよい。

いずれも重要かつ基本的なアルゴリズムである。

準備

rand()

関数はランダムな浮動小数点数を返す。これを用いて配列を作成する。

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

octave では `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。

単純なソート

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。

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。

手順

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

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

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

アルゴリズムの改良

配列要素の交換回数を減らす。
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。
比較回数は $O(n^2)$ だが、入れ替える回数は $O(n)$ になる。

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

二分法による探索

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。
ソート済みなら次のように $O(\log n)$ 程度で済む。

手順

$N = 16, x = 13$ とする。

1	2	3	4	5	6	7	$\underline{8}$	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。

手順

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

$12 < 13$ だから、考える区間はさらに右側である。

手順

1	2	3	4	5	6	7	8	9	10	11	$\underline{12}$	13	$\underline{14}$	15	16

$14 > 13$ だから、考える区間は左側である。

手順

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資料のようなプログラムになる。