Programming—ソート—

課題

選択ソートとクイックソートアルゴリズムを用いて，配列データA:=array([6,4,3,1,2,5])を小さい値順に並べるソートスクリプトを作れ．

課題の背景

簡単な挿入(insert)ソートを例に，配列に入った値がどのようにソートされていくかをMapleで表示してみよう．ソートでは配列の並べ替えをおこなうために，指数(index)が複雑な動きをする．普通のプログラマはこのような複雑な挙動を覚える必要はない．ゆっくり見て一度理解すれば十分．

挿入ソートのスクリプトは以下のとおり．

```
> insertsort:=proc(A::Array)
  local n,x,i,j;
  n:=op(2,op(2,eval(A)));
  for i from 1 to n do
    print("Before",i,A[i],A);
    x:=A[i];
    for j from i-1 by -1 to 1 do
      if (A[j]<x) then break; end if;
      A[j+1]:=A[j];
    end do;
    A[j+1]:=x;
    print("After ",i,j+1,A);
  end do:
  eval(A);
  end proc:
```

上の関数を使って，実際の並べ替えを行うと以下のように表示される．

```
> A:=Array([6,4,3,1,2,5]):
> insertsort(A);
```

$$\text{"Before"}, 1, 6, \begin{bmatrix} 6 & 4 & 3 & 1 & 2 & 5 \end{bmatrix}$$

$$\text{"After "}, 1, 1, \begin{bmatrix} 6 & 4 & 3 & 1 & 2 & 5 \end{bmatrix}$$

$$\text{"Before"}, 2, 4, \begin{bmatrix} 6 & 4 & 3 & 1 & 2 & 5 \end{bmatrix}$$

$$\text{"After "}, 2, 1, \begin{bmatrix} 4 & 6 & 3 & 1 & 2 & 5 \end{bmatrix}$$

$$\text{"Before"}, 3, 3, \begin{bmatrix} 4 & 6 & 3 & 1 & 2 & 5 \end{bmatrix}$$

$$\text{"After "}, 3, 1, \begin{bmatrix} 3 & 4 & 6 & 1 & 2 & 5 \end{bmatrix}$$

$$\text{"Before"}, 4, 1, \begin{bmatrix} 3 & 4 & 6 & 1 & 2 & 5 \end{bmatrix}$$

$$\text{"After "}, 4, 1, \begin{bmatrix} 1 & 3 & 4 & 6 & 2 & 5 \end{bmatrix}$$

$$\text{"Before"}, 5, 2, \begin{bmatrix} 1 & 3 & 4 & 6 & 2 & 5 \end{bmatrix}$$

$$\text{"After "}, 5, 2, \begin{bmatrix} 1 & 2 & 3 & 4 & 6 & 5 \end{bmatrix}$$

$$\text{"Before"}, 6, 5, \begin{bmatrix} 1 & 2 & 3 & 4 & 6 & 5 \end{bmatrix}$$

$$\text{"After "}, 6, 5, \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix} \tag{1.2.1}$$

ここで，挿入する前(Before)と後(After)の配列の違いを観察しよう．たとえば，4番目まで終わった，

"Before", 5, 2, [1, 3, 4, 6, 2, 5]

で内側のfor-loopで何をしているかを考える．5番目の要素の"2"が選ばれる(x:=A[i])．これを右側のもの(A[j])と順番に比べていく．大きければ空いた席(A[j+1])と入れ替える．

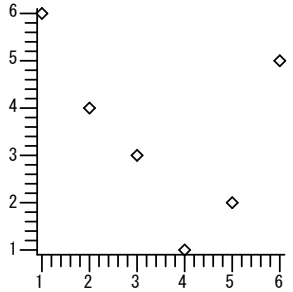
"Compare", 5, 5, [1, 3, 4, 6, _, 5]
"Compare", 5, 4, [1, 3, 4, _, 6, 5]
"Compare", 5, 3, [1, 3, _, 4, 6, 5]
"Compare", 5, 2, [1, _, 3, 4, 6, 5]

小さくなる(A[j] < x)か，あるいはj=1までいけば比較は終了．5番目の要素は1番目までいって，A[1] < xが成立して，A[2]に収まる．

Mapleの関連コマンド

ソートにともなう値の並べ換えを視覚化することを試みます．以下のようにすると

```
> A:=Array([6,4,3,1,2,5]):
  tmp:=[seq([i,A[i]],i=1..6)]:
  with(plots):
  pointplot(tmp,symbolsize=30);
```



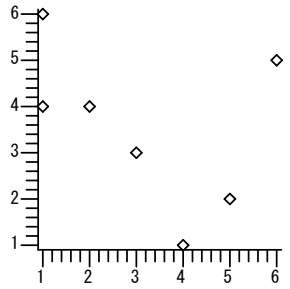
よこが指数，たてが入っている要素．これを使って先程のinsertsortの途中経過を表示させる．

insertsort2 (ソートの視覚化を組み込んだinsertsort)

```
> insertsort2:=proc(A::Array)
  local n,x,i,j;
  global tmp;
  n:=op(op(2,eval(A)))[2];
  for i from 2 to n do
  x:=A[i];
    for j from i-1 to 1 by -1 do
```

```
    tmp:=[op(tmp),
      pointplot([[j,x],seq([ii,A[ii]],ii=1..n)],symbolsize=30)];
      if (A[j]<x) then break; end if;
      A[j+1]:=A[j];
    od;
    A[j+1]:=x;
  od;
  end proc:
```

```
> tmp:=[]:
  A:=Array([6,4,3,1,2,5]):
  insertsort2(A):
  tmp:=[op(tmp),pointplot([seq([i,A[i]],i=1..6)],symbolsize=30)]:
  display(tmp,insequence=true);
```



tmpの中に画像が蓄えられている．Mapleを実際に手元で動かしているときは，displayで表示させた絵をつまんで，コマ送りで一つずつどのように位置が置き変わっていくかを追ってみよ．

解法のヒントと発展問題

挿入ソートのCompareへの改良

上記の"Compare"以下が出力されるようにinsertsortを改良せよ．ただし下線"_"部には違う数字が入っている．その数字はなにか？

選択ソートの作成

次に示した選択ソートの考え方にもとづいてMapleスクリプト(selectsort)を書け．

まず，全体の中で最小のものを見つけ，それを先頭のA[1]と交換する．

次にA[2..n]の中で最小のものを見つけ，\A[2]と交換する．

以下同様に続ける．

さらに，視覚化を試みよ．

quick sort

上のinsertsortを参考にして，選んだ数より大きな数を右に，小さな数を左に置きかえる関数partition(A,first,last)を考えよ．選ぶ数としてはA[last]が簡単．これを使えば，quicksortは以下のようなスクリプトで書ける．

```
> quicksort:=proc(A::Array,first::integer,last::integer)
  local p;
  if first<last then
    p:=partition(A,first,last);
    quicksort(A,first,p-1);
    quicksort(A,p+1,last);
  end if:
  eval(A);
  end proc:
```

発展問題：大きな要素数の乱数

以下のランダムな整数列の作成のprocを用いて要素数を大きくしたときに，insertsort, selectsort, quicksortそれぞれがどの程度の時間がかかるかを計測せよ

```
> BigRandom:=proc(max_n)
  local sel,A,i,j,k,tmp;
  sel:=rand(1..max_n):
  A:=Array([seq(i,i=1..max_n)]):
  for k from 1 to 2*max_n do
    i:=sel();j:=sel();
    tmp:=A[i];A[i]:=A[j];A[j]:=tmp;
  end do:
  return A;
  end proc:
```

時間計測は以下のようにして出来る．printf文をのぞいておくことを忘れないように．

```
> st:= time():
  n:=160:
  A:=BigRandom(n):
  quicksort(A,1,n);
  time() - st;
```

$$\mathrm{quicksort}(\mathrm{BigRandom}(160), 1, 160)$$

$$0.094 \qquad (1.4.4.1)$$