2 algorithms to sort an array of numbers.

Table of Contents

```
Author: Pedro Ciller Cutillas
Date: 3-3-2015
Information: The numbers are sorted in nondecreasing order.
```

First algorithm: auxiliary function

An auxiliary function that changes the positions of two elements in an array has been used. The code of this function is:

```
function c=change(a,b)

    % c=change(a,b)
    % a is a vector of numbers
    % b is an array of two numbers that contains the two indexes of
    % a that are to be changed.
    %
    % The output vector c is the original vector a with the elemenents
    % whose indexes appear in b permuted

    aux=a(b(1));
    a(b(1))=a(b(2));
    a(b(2))=aux;

    c=a;
```

First algorithm: code

```
clear all;
u=input('introduce an array of numbers using the format [a,b,...]:');


sorted=[(max(u)+1).*ones(1,length(u))];

% The initialization values of the array 'sorted' are greater than
% the maximum element of vector u. This is required in the if conditional
% inside the following nested loop.

for j=1:length(u)
    for i=1:length(u)-(j-1)
```

```
         if( u(i)<=sorted(j))
        sorted(j)=u(i);
        position=i;
         end
    end
    u=change(u,[position,length(u)-(j-1)]);
end

% For each iteration of the index j, the j-th minimum value of the 'u'
% vector is calculated and stored in the j-th position of 'sorted'. Then,
% this value is moved to the end of the 'u' vector using the function
% 'change' in order to avoid being considered in the next iterations.

% Display the results
disp(sprintf('The introduced array has been sorted and the result is: '));
sorted_to_string=sprintf('%d ', sorted);
fprintf('%s\n', sorted_to_string);
```

Second algorithm: code

```
clear all;
u=input('introduce an array of numbers using the format [a,b,...]:');

smaller=zeros(1,length(u));
equal=zeros(1,length(u));

% Arrays 'smaller' and 'equal' are going to store in their i-th posicion
% the number of elements of the input vector 'u' that are smaller and equal
% than the i-th element of the input 'u' vector. In the 'equal' array, the
% fact that each element is equal to itself will not be considered in the
% final count.

for i=1:length(u)
        for j=1:length(u)
            if(i~=j)
                            if(u(j)<u(i))
                                smaller(i)= smaller(i)+1;
                            elseif(u(j)==u(i))
                                equal(i)= equal(i)+1;
                            end
            end
        end
end

sorted=(max(u)+1).*ones(1,length(u));

% The initialization values of the array 'sorted' are greater than
% the maximum element of vector u. This is required in the if conditional
% inside the following nested loop. It would be enough to initialize the
% array 'sort' with any number that is not contained in the 'u' vector.

for i=1:length(u)
    for j=1:equal(i)+1
```

```
        if(sorted(smaller(i)+j)==max(u)+1)
            sorted(smaller(i)+j)=u(i);
        end
    end
end

% The previous nested for loops determine each element of the 'sort' array
% using the information previously stored in 'smaller' and 'equal'. The case
% equal(i)~=0 is solved taking into account the initialization values of
% the 'sort' array.

% Display the results
disp(sprintf('The introduced array has been sorted and the result is: '));
sorted_to_string=sprintf('%d ', sorted);
fprintf('%s\n', sorted_to_string);
```

Published with MATLAB® R2014a