Sorting

CS10003 PROGRAMMING AND DATA STRUCTURES

The Basic Problem

Given an array: `x[0], x[1], ... , x[size-1]` reorder the elements so that

```
x[0] <= x[1] <= ...  <= x[size-1]
```

- That is, reorder entries so that the list is in increasing (non-decreasing) order.

We can also sort a list of elements in decreasing (non-increasing) order.

We prefer not to use additional arrays for the element rearrangement.

Example

Original list:

```
10, 30, 20, 80, 70, 10, 60, 40, 70
```

Sorted in non-decreasing order:

```
10, 10, 20, 30, 40, 60, 70, 70, 80
```

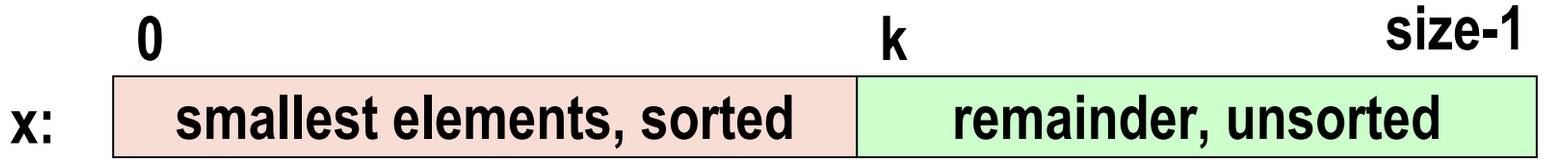
Sorted in non-increasing order:

```
80, 70, 70, 60, 40, 30, 20, 10, 10
```

Selection Sort

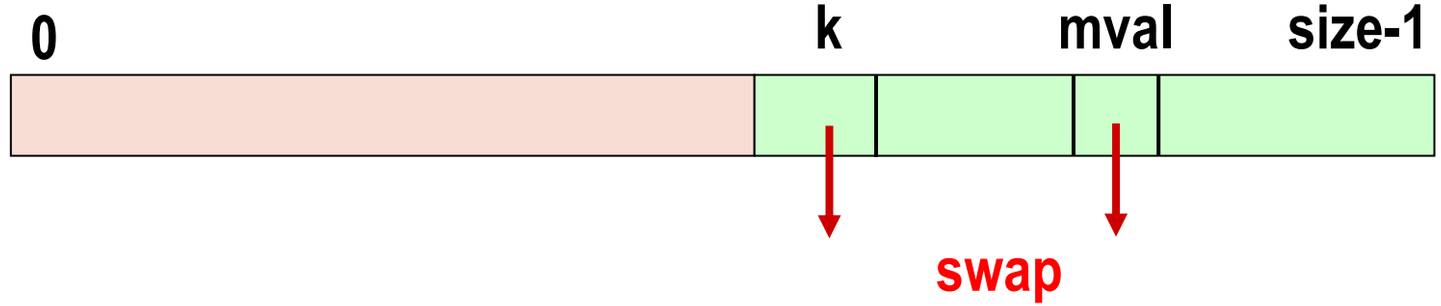
SELECTION SORT: The idea

General situation :



Steps:

- Initialize `k = 0`.
- Find smallest element, `mval`, in the array segment `x[k...size-1]`
- Swap smallest element with `x[k]`, then increase `k`.



Subproblem

```
/*  Find index of smallest element in x[k...size-1] */

int min_loc (int x[ ], int k, int size)
{
      int j, pos;

      pos = k;
      for (j=k+1; j<size; j++)
        if (x[j] < x[pos])
            pos = j;
      return pos;
}
```

Selection Sort Function

```
/* Sort x[0..size-1] in non-decreasing order */

int sel_sort (int x[], int size) {
      int k, m, temp;

      for (k = 0; k < size-1; k++) {
         m = min_loc (x, k, size);
            /* Swap x[k], x[m]*/
         temp = x[k];
         x[k] = x[m];
         x[m] = temp;
      }
}
```

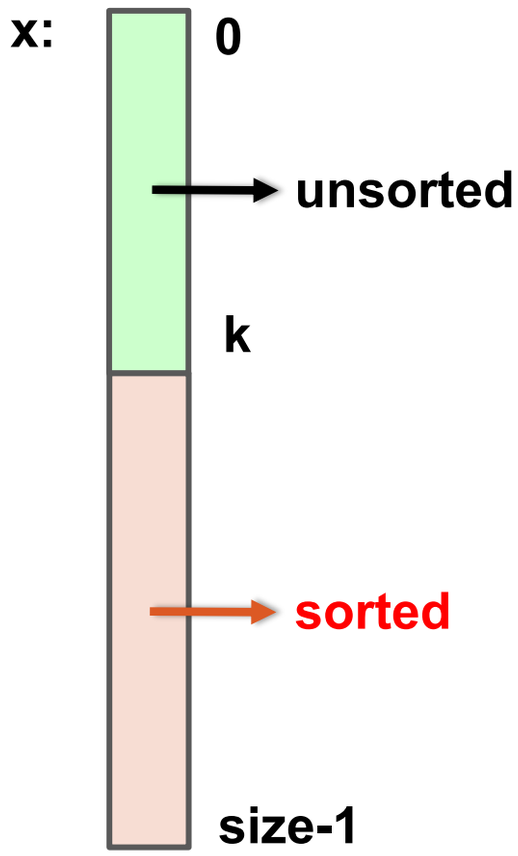
Example


x: 3 12 -5 6 142 21 -17 45
x: -17 12 -5 6 142 21 3 45
x: -17 -5 12 6 142 21 3 45
x: -17 -5 3 6 142 21 12 45
x: -17 -5 3 6 142 21 12 45
x: -17 -5 3 6 12 21 142 45
x: -17 -5 3 6 12 21 142 45
x: -17 -5 3 6 12 21 45 142
x: -17 -5 3 6 12 21 45 142

Bubble Sort

BUBBLE SORT: The idea

General situation:



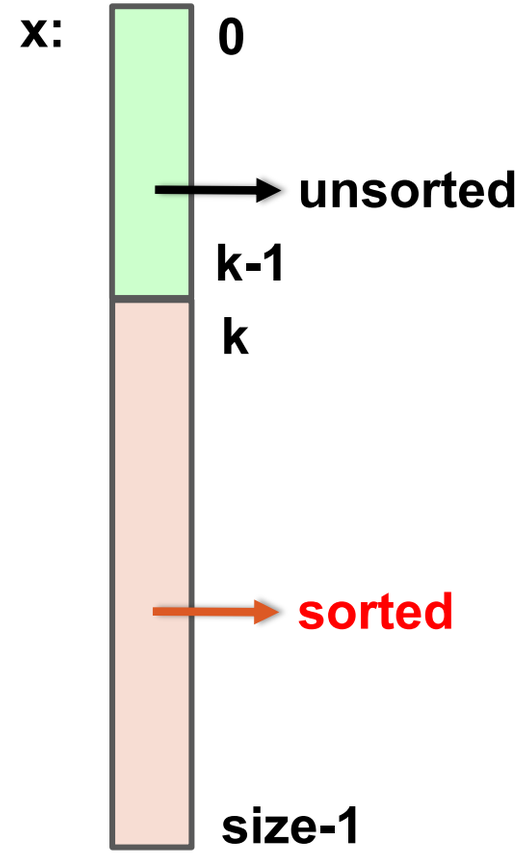
In every pass, we go on comparing neighboring pairs, and swap them if out of order.

```
for j = 0 to k-1
  if (x[j] > x[j+1])
      interchange them.
```

At the end of this iteration, the 'next largest' element (among the unsorted part) will settle at x[k].

Lighter elements bubble up.

Heavier elements settle down.



Bubble Sort

```
void bubble_sort (int x[], int size) {
    int t;
    for (i = 0; i < size; i++)
        for (j = 0; j < size-i-1; j++)
            if (x[j] > x[j+1]) {
                        // swap a[j] and
a[j+1]
                t = a[j];
                a[j] = a[j+1];
                a[j+1] = t;
            }
}
```

How do the passes proceed?

In pass 1, we consider index 0 to size-1
In pass 2, we consider index 0 to size-2
In pass 3, we consider index 0 to size-3
.....
.....
In pass size-1, we consider index 0 to 1.