Sorting

- Bubble sort:
  - Compares consecutive items : swaps them if they are in incorrect order; repeat until no more swaps are necessary
- Insertion sort:
  - Insert one item at a time in the right position in a partial sorted array.
- Selection sort:
  - Find the smallest item and move it in the first position; find the second smallest, move it in the second position and so on until the end of the array.

Bubble-sort example

0	1	2	3	4
3	1	5	4	2

1st pass swaps: (3,1), (5,4), (5,2) # swaps = 3

3 1 5 4 2 → 1 3 5 4 2 → 1 3 4 5 2 → 1 3 4 2 5

2nd pass swaps: (4,2) # swaps= 1

1 3 4 2 5 → 1 3 2 4 5

3rd pass: swaps (3,2), # swaps = 1

1 3 2 4 5 → 1 2 3 4 5

4th pass: no swaps # swaps = 0

1 2 3 4 5 STOP !!!

Bubble-sort pseudo-code

```
Inputs: array a, array size: size
Data:   nrPasses = 0;
        nrSwaps = 1;
        WHILE (nrSwaps > 0 && nrPasses < size);
            nrSwaps = 0
            FOR i = 0 TO size-nrPasses-1
                  IF (a[i] > a[i+1])
                        nrSwaps++
                        swap a[i] with a[i+1]
                  ENDIF
                 i++
            ENDFOR
            nrPasses++
        ENDWHILE
```

Insertion sort example

0	1	2	3	4
3	1	5	4	2

1st pass: key = 1

[3] 1 5 4 2 → [1 3] 5 4 2

2nd pass : key = 5

[1 3] 5 4 2 → [1 3 5] 4 2

3rd pass: key = 4

[1 3 5] 4 2 → [1 3 4 5] 2

4th pass: key = 2

[1 3 4 5] 2 → [1 2 3 4 5]

Blue box = Sorted array

Insertion sort pseudo-code

```
Inputs: array A, array size: size
1.      FOR ind_key = 1 to size-1
              Add A[ind_key] to the sorted sequence
              A[1, .. ind_key-1]
2.            key = A[ind_key]
3.            ind_sorted = ind_key – 1
4.            WHILE ind_sorted >= 0 and A[ind_sorted]>key
5.                  A[ind_sorted +1] = A[ind_sorted]
6.                  ind_sorted = ind_sorted -1
7.            ENDWHILE
8.            A [ind_sorted +1] = key
9.     ENDFOR
```

Selection sort example

A

0	1	2	3	4
3	1	5	4	2

1st pass: min(0, size-1) = 1, swap min to A[0]

1 3 5 4 2

2nd pass: min(1,size-1) = 2 swap min to A[1]

1 2 5 4 3

3rd pass: min(2,size-1) = 3 swap min to A[2]

1 2 3 4 5

4th pass: min(3,size-1) = 4 swap min to A[3]

1 2 3 4 5

Selection sort pseudo-code

```
Inputs: array A, array size: size

      FOR k = 0 TO k = size -1
            ind_min = k;
            FOR ind_not_sorted = k+1 TO size-1
                IF (A[ind_not_sorted) < A[ind_min])
                    ind_min = ind_not_sorted
                ENDIF
            ENDFOR
            swap A[ind_min] with A[k]
      ENDFOR
```

Questions

- What happens after the first pass of the bubble-sort? What about after the second pass?
- Do you need to check the whole array in each pass of the bubble sort?
- What is the minimum number of passes for an array of size N in case of:
  - Bubble-sort
  - Insertion-sort
  - Selection-sort

Questions

- Which sorting algorithm you think is better among the three discussed here?
- Why?