**ALGORITHMS & DATA STRUCTUREs**

**Lahcen Ouarbya**

**TOPIC 3: SORTING**

**WORKING SESSION 1**

In this working session you have to work with 3 comparison-based sorting algorithms: Bubble Sort, Insertion Sort and Selection Sort. For each one of them, you will:

* Solve exercises that help you understand the basic idea behind every algorithm
* Calculate their worst and best case time complexities
* Implement them

**Learning Objectives:**

1. **Understand the operation of every algorithm** by determining the state of the array as it gets processed by different sorting algorithms
2. **Calculate the time complexity** of worst and best cases of Bubble, Insertion, and Selection Sort
3. **Implement** them in either C++ (Games programming students) or Java (Computer Science students)

|  |
| --- |
| **PART 1: BUBBLE SORT** |

|  |
| --- |
| 1. **function** BubbleSort(A)  2. n=A.size()  3. change=true  4. **while** (change) **do**  5. change=false  6. **for** 0 <= i < n-1 **do**  7. **if** list[i] > list[i+1] **then**  8. swap(list[i],list[i+1])  9. change=true  10. **end if**  11. **end for**  12. n=n-1  13. **end while**  14. return A  15.**end function** |

1. For the array A=[45,12,7,49,23], please answer the following questions:

* How many iterations does the outer (while) loop do in total?
* What is the content of the array immediately after finishing each iteration of the outer loop?

Iteration 1:

Iteration 2:

….

1. What line(s) of the pseudocode given would you change for Bubble to sort from largest to smallest (instead of smallest to largest)?
2. Please, describe the worst and best case for the running time of Bubble Sort. Remember: the worst-case (best-case) corresponds to the case where the maximum (minimum) number of operations is executed
3. Please, find a generic expression (“generic” means you do not need to calculate the exact value of the constants; you can use C0, C1, etc.) for the running time of Bubble Sort for the worst (TW(n)) and best (TB(n)) cases.
4. Please, for the worst and best cases, calculate the time complexity of Bubble Sort in terms of the number of elements of array (n). Remember that O() and Ω()refer to a set of functions. Please, write at least 3 for each.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **O()** | **Ω()** | **Θ()** |
| Worst-case |  |  |  |
| Best-case |  |  |  |

1. Implement both versions of Bubble Sort (sorting from smallest to largest & sorting from largest to smallest) in either C++ (Games programming Students) or Java (Computer Science students).

|  |
| --- |
| **PART 2: INSERTION SORT** |

|  |
| --- |
| 1. **function** InsertionSort(A)  2. n=A.size()  3. **for** 1 <= j <= n-1 **do**  4. ins=A[j]  5. i=j-1  6. **while** (i>=0 and ins<A[i])  7. A[i+1]=A[i]  8. i --  9. **end while**  10. A[i+1]=ins  11. **end for**  12. **end function** |

1. For the array A=[45,12,7,49,23], please answer the following questions:

* How many iterations does the (outer) for loop do in total?
* What is the content of the array immediately after finishing each iteration of the outer loop?

Iteration 1:

Iteration 2:

….

1. What line(s) of the pseudocode given would you change for Selection to sort from largest to smallest (instead of smallest to largest)?
2. Please, describe the worst and best case for the running time of Selection Sort. Remember: the worst-case (best-case) corresponds to the case where the maximum (minimum) number of operations is executed
3. Please, find a generic expression (“generic” means you do not need to calculate the exact value of the constants; you can use C0, C1, etc.) for the running time of Bubble Sort for the worst (TW(n)) and best (TB(n)) cases.
4. Please, for the worst and best cases, calculate the time complexity of Bubble Sort in terms of the number of elements of array (n). Remember that O() and Ω()refer to a set of functions. Please, write at least 3 for each.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **O()** | **Ω()** | **Θ()** |
| Worst-case |  |  |  |
| Best-case |  |  |  |

1. Implement both versions of Insertion Sort (sorting from smallest to largest & sorting from largest to smallest) in either C++ (Games programming Students) or Java (Computer Science students).

|  |
| --- |
| **PART 3: SELECTION SORT** |

|  |
| --- |
| **1. function** SelectionSort(A)  2. n = A.size()  3. **for** 0 ≤ i ≤ n-2 **do**  4. min\_idx = i  5. for i+1 < j <n  6. min\_idx = j; 7. end for  8. temp = A[min\_idx]  9. A[min\_idx] = A[i]  10. A[i] = temp  11. **end for**  **12 . end function** |

1. For the array A=[45,12,7,49,23], please answer the following questions:

* How many iterations does the (outer) for loop do in total?
* What is the content of the array immediately after finishing every iteration of the outer loop?

Iteration 1:

Iteration 2:

….

1. What line(s) of the pseudocode given would you change for Selection to sort from largest to smallest (instead of smallest to largest)?
2. Please, describe the worst and best case for the running time of Selection Sort. Remember: the worst-case (best-case) corresponds to the case where the maximum (minimum) number of operations is executed
3. Please, find a generic expression (“generic” means you do not need to calculate the exact value of the constants, you can use C0, C1, etc.) for the running time of Bubble Sort for the worst (TW(n)) and best (TB(n)) cases.
4. Please, for the worst and best cases, calculate the time complexity of Bubble Sort in terms of the number of elements of array (n). Remember that O() and Ω()refer to a set of functions. Please, write at least 3 for each.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **O()** | **Ω()** | **Θ()** |
| Worst-case |  |  |  |
| Best-case |  |  |  |

1. Implement both versions of Selection Sort (sorting from smallest to largest & sorting from largest to smallest) in either C++ (Games programming Students) or Java (Computer Science students).