Data Structures and Algorithms

Exam Revision

Amilcar Aponte

amilcar@cct.ie

Today’s plan

- Exam structure
- Time management
- Sample questions
- Topic revision

Exam Structure

- You will have to answer **ten questions** in total.
- Time allowed: **100 minutes** (the time will start as soon as you begin the online test).

- The ten questions will be randomly selected for you from five pools of questions (two questions from each pool). The topics of each pool are:
  - Linear Data Structures
  - Non-Linear Data Structures
  - Searching Algorithms
  - Sorting Algorithms
  - Algorithm Complexity

Use your time wisely!

- Total time - 100 mins

- Read the whole exam first – 10 mins
- Answer all your exam – 80 mins
  - 8 min/questions
- Read all your answers at the end – 10 mins

Question Bank 1 (Linear Data Structures) – Sample Question

A queue is a data structure that follows the FIFO principle and could be implemented using an array, arrayLists or linked structure as the underlying data container.

A customer requires a system that can store data in a queue that also allows fast access to elements other than the first and the last.

- State which data structure you will use as the underlying data container.
- Explain in your own words how this data structure works in this context.
- Cite one benefit and one drawback of the data structure you chose.

Question Bank 1 (Linear Data Structures) – Sample Answer

- A good structure would be an arrayList.
- There is no limit in the number of elements that can be stored and more importantly, there is index access to all of the elements at a fast speed.
- A benefit is that it allows easy navigation of the elements back and forward.
- A drawback is that adding elements in the correct position of the list could be time-consuming if the collection is too big.

Question Bank 2 (Non Linear Data Structures) – Sample Question

The three methods shown below are part of a binary tree class and they allow the tree to be traversed following one of the three major traversals for binary trees: IN ORDER TRAVERSAL. Assuming that the Node class has a "getLeftChild()" and a "getRIghtChild()" getter:

- Write the code missing in the traverse method.
- Explain in your own words how the code works.
- How does this traversal differ from the other two?

Question Bank 2 (Non Linear Data Structures) – Sample Answer

- // Whatever code is missing.
- The code will visit (calling the method "visit") each one of the nodes in the corresponding order: Left, root, right.
- The only difference is the order in which the nodes are visited.

Question Bank 3 (Searching Algorithms) – Sample Question

The code shown below implements the binary search algorithm. But there is one small bug in the code.

- What is the bug in the code?
- How could you fix it?
- What happens if the target value is included more than once in the array?

Question Bank 3 (Searching Algorithms) – Sample Question

```
public int search(Integer[] array, int e){
    int low = 0;
    int high = array.length - 1;
    while(low <= high){
        int mid = (high + low)/2;
        if(array.get(mid) == e){
            return mid;
        } else if(array.get(mid) > e){
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    return -1;
}
```

Question Bank 3 (Searching Algorithms) – Sample Answer

- The input of the method is an array and arrays don't have .get(i) method.
- Change the variable .get(i) for [i]
- The algorithm will return the position of the first one that appears.

Question Bank 4 (Sorting Algorithms) – Sample Question

Consider the standard Bubble Sort Algorithm shown below in the form of pseudo-code.

- Consider A = {31, 15, 12, 27}. Indicate the order of the array in each pass of the algorithm.
- What changes do you have to do to the code to sort the array in reverse order?

Question Bank 4 (Sorting Algorithms) – Sample Question

```
bubbleSort(A)
    for i=0 to length(A) - 2
        for j = 0 to length(A) - 2 - i
         if  A[j] > A[j+1] then
      swap(A[j], A[j+1])
          endif
      endfor
    endfor
endalg
```

Question Bank 4 (Sorting Algorithms) – Sample Answer

- {15, 31, 12, 27}
- {15, 12, 31, 27}
- {15, 12, 27, 31}
- {12, 15, 27, 31}
- {12, 15, 27, 31}
- {12, 15, 27, 31}
- Change the simbol ">" for "<".

Question Bank 5 (Algorithm Cmplexity) – Sample Question

Consider the following algorithm.

- What is the complexity of the algorithm in Big-Oh notation?
- What is the purpose of the algorithm?
  - HINT: Assess the value of count for n = 1, n=2 and n = 3.
- Without writing any code. Describe another more efficient implementation for this algorithm?

Question Bank 5 (Algorithm Complexity) – Sample Question

```
public int method(int n) {
    int count = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i; j > 0; j--) {
            count = count + 1;
        }
    }
    return count;
}
```

Question Bank 5 (Algorithm Complexity) – Sample Answer

- O(N^2)
- The algorithm is adding up consecutive numbers.
  - n = 1 => count = 1
  - n = 2 => count = 3
  - n = 3 => count = 6
- Using a single for-loop that add the value of "i" to the "count" variable.

Topics

- Linear Data Structures
  - What are they?
  - What are they used for?
  - What is meant by FIFO and LIFO
  - What methods do they have?
  - Complexity
  - Advantages
  - Disadvantages

- Stacks, queues, priority queues, Lists.

Topics

- Searching Algorithms
  - Linear search, binary search.
  - Complexities
  - Logic
  - Advantages
  - Disadvantages
  - Code/ Pseudo Code

Topics

- Sorting Algorithms
  - Bubble Sort, insertion sort, merge sort.
  - Complexities
  - Logic
  - Advantages
  - Disadvantages
  - Code/ Pseudo Code

Topics

- Trees
  - What are they?
  - What are they used for?
  - How are they used?
  - What methods do they have?
  - Binary trees
  - Tree concepts
    - Leave, height, level, node, child…

Topics

- Trees Transversals
  - InOrder
  - PreOrder
  - PostOrder
  - Code/Pseudo code
  - Complexity

Topics

- Complexity
  - What is it?
  - Big O notation categories
  - Complexity Equations
  - Calculate the complexity of an algorithm