DIT725 V16, Assignment 2: Time Complexity

This assignment is divided into three tasks, and a small individual assignment. The first group task is relatively small compared to the other two.

See the course homepage for deadlines and general instructions on group assignments. You can also find the Node.java and SortedList.java there.

1 Trees and tree algorithms

In Node.java there is an implementation of a simple recursive tree data structure, and a main function that creates a tree and tries to print it. It uses a function printBF that prints the elements of a tree in breadth first order (level by level, from left to right on each level). Your assignment is to implement this method.

To do this you are given the following slightly cryptic suggestion for an algorithm that may work:

1. Start with a list containing the root node
2. Take the first node from the list and print its element
3. Add its child nodes to the end of the list in the correct order
4. Repeat from step 2 until all nodes have been processed

For this task you should hand in Node.java. No report is needed, but comment your code enough to make it understandable.

Hint: java.util.LinkedList has special methods for adding/removing to/from front or end of list, see:
https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html

Hint: Think of how we used Stacks to avoid recursion (in one of the lectures), this algorithm is similar but instead of Last-In-First-Out (LIFO) we are using First-In-First-Out (FIFO). The list is used as a queue (see section 6.2 in G,T&G).

Challenge: This task can be completed using around 10 lines of carefully considered code.

Challenge: Can you make a recursive version? (May require using other data structures than lists)

2 Implementing a data structure for sorted list

In SortedList.java there is an interface, describing a sorted list abstract data type. Your assignment is to write a class that implement this interface.

In the interface file there are comments describing the methods, some complexity requirements and some hints and challenges. You are also given TestList.java, containing a method using the interface and some comments containing the expected output of running the program with your implementation. Use this to test your implementation and to understand how it is expected to work.

You should analyse the time-complexity of your implementation. For each method the interface provides you should state which complexity class the method is in and motivate your claim in terms of your code. Include relevant code pieces (loops, recursive functions etc.) and explain how you calculated the complexity for each method.

You should submit a file SortedListImpl.java with your java class. You should also submit a file report.pdf, containing a short description of your implementation and your complexity analysis.

Challenge: It is possible to implement this in 80 lines of code (not counting comments and empty lines). How close to that can you get? (Keep your code readable though!)

Challenge: Feel free to implement more than one version if you have the time - for instance an array and a tree-bases solution, you can skip some methods in the alternative solution or even just sketch how you think it could be done.

Challenge: First make a more precise analysis of the time complexity and use the formal defintion of big O to argue for which complexity class your method is in.

3 Time complexity of insertion sort

Here is an implementation of an algorithm called insertion sort for `int` arrays (see section 3.1.2 in G,T&G).

```
public static void insertionSort (int [] a) {
  for (int i = 1; i < a.length; i++) {
    // Invariant: a[0..i-1] is already sorted.
    final int cur = a[i];
    // Now insert cur into a[0..i-1],
    // shifting greater elements upward.
    int j = i;
    while (j > 0 && a[j-1] > cur)
      a[j] = a[--j];
    a[j] = cur;
  }
}
```

a) Analyse the algorithm and determine the complexity class it is in ($O(f(N))$, where N is the length of the array to be sorted).

b) The complexity talks about the worst case scenario. What kind of input corresponds to the worst case?

c) What kind of input gives the best performance? What is the running time then ($O(f(N))$)?

d) Empirically compare the running time of your insertion sort implementation with the built-in sorting method for int arrays in Java, which is `Arrays.sort` from package `java.util`. Run each sorting functions on a number of arrays of different sizes. Make a program that automatically runs the two methods a large number of times. Use as large values as possible without using too much time or memory. The arrays you test with should contain random values. Plot the running times in a diagram.

e) Any guesses at what the running time of the built-in sorting method is?

You should submit a) a Java program (.java file) with the code that runs the tests, and b) a pdf answering the questions and showing the graph.

Hint: There are several plotting libraries for Java (e.g. JFreeChart) but the easiest solution may be to print the numbers and paste them in a spreadsheet application.

Hint: you can use `System.currentTimeMillis()` to do some (basic) calculations regarding how much time Java spends on a method. For instance, you can write:

```java
long start = System.currentTimeMillis();
sort_array(array); // Task we want to benchmark
long end = System.currentTimeMillis();
long duration = end - start;
System.out.println("Sorting␣took␣" + duration + "␣ms");
```

4 Individiual assignment

Every member of the group should individually work on this task, then all your solutions should be submitted along with the group solutions to the other tasks. Each member should provide a .pdf file with the group members name in the file name, containing the solution to the assignment. The individual part of the assignment is mostly intended as an exercise, but all members are required to submit a (mostly) correct solution to (individually) pass the assignment.

The task: Find a (relatively small) piece of code you have written (in a course/project or on your spare time, or write some code now if you prefer) and analyze its complexity! Particularly you should determine what complexity class it is in, motivating your answer. Perhaps you have some method that was too slow to use, and a complexity analysis to give some insight into why?

Also: Do you think it is possible to get better complexity solving the same problem, or is this as good as it gets? Could you have used any of the data structures we have looked at in the course to make the code smaller or more efficient?

Hint: Since you need to have some sort of input size parameter, code dealing with lots of data is good. Try to avoid code that uses lots of library methods, or you will have to guess the complexity of those.