Data Structures and Algorithms

COSC 336 Assignment 1

Instructions.

1. Due Feb 16.

2. This is a team assignment. Work in teams of 2-3 students. Submit on Blackboard one assignment per team, with the names of all students making the team.

3. Your programs must be written in Java.

4. Write your programs neatly - imagine yourself grading your program and see if it is easy to read and understand.

   Comment your programs reasonably: there is no need to comment lines like "i++" but do include brief comments describing the main purpose of a specific block of lines.

5. You will submit on **Blackboard** two files.

   The **first file** is a pdf file (produced ideally with latex and Overleaf) and it will contain the following:

   (a) The solution to the Exercise.

   (b) A short description of your algorithm for the Programming Task, where you explain the dynamic programing approach (see the sketch of the **Algorithm** below). More precisely, you need to indicate how you compute $d[0]$ (this is the initialization step), and how you compute for every $i \geq 1$, the value of $d[i]$ using the values of some of the previous $d[j]$'s, for $j < i$).

   (c) A table with the results your program gives for the three data sets given below.

   (d) The java code (so that the grader can make observations).

   The **second file** is the .java file containing the java source code, so that the grader can run your program.

Exercise
Consider the following three program fragments (a), (b), and (c).

(a)
```
sum = 0;
for (int i = 0; i< n ; i++) {
        sum++;
}
```

(b)
```
sum = 0;
for (int i = 0; i< 2*n ; i++) {
        sum++;
}
```

(c)
```
sum = 0;  i=n*n;
while (i > 1) {
        sum++;
        i= i/2;
}
```

We denote by $T_a(n), T_b(n), T_c(n)$ the running time of the three fragments.

1. Give Θ evaluations for $T_a(n), T_b(n), T_c(n)$.
2. Is $T_b(n) = O(T_a(n))$? Answer YES or NO and justify your answer.
3. Is $T_c(n) = \Theta(T_a(n))$? Answer YES or NO and justify your answer.

Programming Task.

You will write a program that computes the length of a longest increasing subsequence of a sequence of integers.

Formally, an increasing subsequence of the sequence $a_1, a_2, \ldots, a_n$ of length k is given by k indices $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ such that $a_{i_1} < a_{i_2} < \ldots < a_{i_k}$. So the goal is to find the largest k for which there exists an increasing subsequence of the input sequence of length k. Note: There is one major difference from the problem with *max contiguos subsequence sum* which we discussed in class, namely in this problem the subsequence is **not contiguous**, meaning that the numbers in the subsequence do not have to be in consecutive positions.

For example, if the input sequence is $10, 9, 2, 5, 3, 101, 7, 18$ then a longest increasing subsequence is $2, 5, 7, 18$, which has length 4 (there is another increasing subsequence, namely $2, 3, 7, 18$, also of length 4). Therefore your program should return 4 because there is no increasing subsequence of length 5 or larger.

Your program will read the initial sequence which is entered by the user, and will print the length of a longest subsequence. As a bonus, you may want your program to also print one longest increasing subsequence.

Algorithm You will implement an algorithm using the dynamic programming paradigm, which is similar to Algorithm 3 for *max contiguous subsequence sum* that we discussed in our meeting (see Notes1-Intro on Blackboard).

Suppose the initial sequence is $a_0, a_1, \ldots, a_{n-1}$. Then, you can calculate in order, one by one, the elements of an array $d[0], \ldots, d[n-1]$, in which $d[i]$ is the length of the longest increasing subsequence whose last term is a_i. Think how to calculate $d[0]$ and then how to calculate $d[i]$ as a function of the previous entries $d[1], \ldots, d[i-1]$ and the sequence $a[]$.

Example:

Input: $10, 9, 2, 5, 3, 101, 7, 18$. Output: 4, or for the bonus solution $4, (2, 5, 7, 18)$.

Test your program on the following sequences and insert to the first file that you submit screenshots with the computer screen showing the results for each sequence:

- $10, 9, 2, 5, 3, 101, 7, 18$
- 186, 359, 274, 927, 890, 520, 571, 310, 916, 798, 732, 23, 196, 579, 426,188, 524, 991, 91, 150, 117, 565, 993, 615, 48, 811, 594, 303, 191, 505, 724, 818, 536, 416, 179, 485 , 334 , 74, 998, 100, 197, 768, 421, 114, 739, 636, 356, 908 , 477, 656
- 318 , 536 , 390 , 598 , 602 , 408 , 254 , 868 , 379 , 565 , 206 , 619 , 936 , 195 , 123 , 314 , 729 , 608 , 148 , 540, 256 , 768 , 404 , 190 , 559 , 1000 , 482 , 141 , 26, 230 , 550 , 881 , 759 , 122 , 878, 350, 756, 82, 562, 897, 508, 853, 317 , 380 , 807 , 23 , 506 , 98 , 757 , 247