Hudson Cho, Ryan Wilson, Jesse Washburn, Colin Shuster, Samhith Patibandla
COSC 336
02/18/2025

Assignment 1

Exercise 1.
Consider the following three program fragments (a), (b) and (c)

(a)
```
sum = 0;
 for (int i = 0; i < 100 * 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 $T_a(n), T_b(n), T_c(n)$.

   $T_a(n) = \Theta(n)$
   $T_b(n) = \Theta(n)$
   $T_c(n) = \Theta(\log(n))$

2. Is $T_b(n) = O(T_a(n))$? Answer YES or NO and justify your answer.

   YES. This is because the are both equal to $\Theta(n)$ and since $O(n)$ can be intuitively thought of as "less than or equal to", $T_b(n) = O(T_a(n))$.

3. Is $T_c(n) = O(T_a(n))$? Answer YES or NO and justify your answer.

   YES. as we have learned in class $\log(n)$ has a slower growth rate than $O(n)$ and therefore $T_c(n) = O(T_a(n))$.

Exercise 2.

Give an example of a function $f(n)$ with the property that $f(n)$ is $\omega(n^2)$ and also $f(n)$ is $o(n^3)$.

$f(n) = \Theta(n^{2.5})$.

Exercise 3.

Indicate the runtime of the following program fragment in the $\Theta(\cdot)$ notation.

```
x = 0;

for (i = 1; i <= 2n + 3; i++)
    for (j = 1; j < 3n + 7; j++)
        x = x + 1
```

Since the inner loop runs $3n+6$ times for each iteration of the outer loop, and the outer loop runs $2n + 3$ times, the total runtime can be expressed as $(3n + 6)(2n + 3)$ or $\Theta(n^2)$.

Programming Task 1 Our algorithm used a secondary array $d[j]$ which held the length of each subsequence whose values increased by at most 1. The code iterates through the array once and as it goes through it checks each value of $a[i]$ against the previous to ensure that there is an increase of no more than 1. $d[j]$ is then calculated by adding 1 to $d[j-1]$, essentially extending the length of the current subsequence. Finally we just get the max value from $d[j]$ which is the length of the largest subsequence.

Input	**Output**
2, 5, 5, 1, 11, 12, 13, 3, 52, 53, 53, 54, 4, 7	4
1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1	9
1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 9, 7, 7, 7, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2	22
4,000 bit pseudo-randomly generated number using java Random class	17

Raw Code

```
import java.util.Random;
public class Assignment1Task1 {
    public static void main(String[] args) {
        int[][] sequences = { // Define the sequences
                {2, 5, 5, 1, 11, 12, 13, 3, 52, 53, 53, 54,
                   4, 7},
                {1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1,
                    0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
                    1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                    0, 1},
                {1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5,
                    6, 6, 6, 6, 6, 6, 7, 9, 7, 7, 7, 7, 7, 1,
                    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                    1, 1, 2},
                generateRandomSequence(4000)
        };

        // Process each sequence
        for (int i = 0; i < sequences.length; i++) {
            int maxLength = findLongestSubsequence(sequences[
               i]);
            System.out.println("Sequence " + (i + 1) + ":");
            System.out.println("Length of longest subsequence
               : " + maxLength + "\n");
        }
    }

    public static int findLongestSubsequence(int[] nums) {
        int n = nums.length;
        int[] d = new int[n]; // Array to store the lengths
           of subsequences
        d[0] = 1; // The first element is at least a
           subsequence of length 1
        int maxLength = 1;

        for (int i = 1; i < n; i++) {
            if (0 <= nums[i] - nums[i - 1] && nums[i] - nums[
               i - 1] <= 1) {
                d[i] = d[i - 1] + 1; // increase subsequence
                   length
            } else {
                d[i] = 1; // reset subsequence length
            }
```

```
            if (d[i] > maxLength) {
                maxLength = d[i];
            }
        }
        return maxLength;
    }

    //Helper method to generate sequence of 1s and 0s of
       length @param length
    public static int[] generateRandomSequence(int length) {
        Random random = new Random();
        int[] sequence = new int[length];
        for (int i = 0; i < length; i++) {
            sequence[i] = random.nextInt(2); // Generates 0
               or 1
        }
        return sequence;
    }
}
```