Question: Recursion & Dynamic Programming

2

a) Show that the complexity of $f(x) = x^3 + 2x + 1$ is in $\mathcal{O}(n^3)$.

2

b) Are the following statements true? Prove your answer.

1. $3\sqrt{n} + 5 = \mathcal{O}(n)$
2. $5n + 3 = \mathcal{O}(\sqrt{n})$

3

c) Use R to visualize the growth of different complexity classes, namely, linear, quadratic and exponential growth. Plot each for $x \in [0, 5]$. Make sure to adjust your x- and y-axis to fit each function into the plot. What pattern do you observe? How does the result change when plotting $x \in [0, 15]$?

3

d) Write a recursive function `isPalindrome` which takes a string as input and returns a boolean value whether it is a palindrome. Test your function with the following inputs:

- a
- abbbab
- bacbcab
- aca

4

e) Change the bottom-up dynamic programming for computing Fibonacci numbers to have a space complexity of $\mathcal{O}(1)$. What is the 1000-th element in the Fibonacci sequence?

What is the advantage of the previous bottom-up approach using a space of $\mathcal{O}(n)$ compared to this approach?

4

f) Utilize top-down dynamic programming to write a function that calculates binomial coefficients. Use the following definition as your recursion rule:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \tag{1}$$

$$\binom{n}{0} = \binom{n}{n} = 1 \tag{2}$$

Test your function by computing *12 choose 7*. Is this algorithm efficient?

$\boxed{6}$

g) Consider the maximum value contiguous subsequence problem. Given a vector $V = [v_1, \ldots, v_n]^T \in \mathbb{R}^n$, the task is to the find a value for the maximimation problem

$$\sigma = \max_S \sum_{i \in S} v_i \tag{3}$$

for a subsequence $S = [s, s+1, \ldots, s+m]$. Implement a procedure to solve the above problem by using dynamic programming.

What is the value of the maximum contiguous subsequence of the following vector?

```
target_sequence <- c(12, -5, 7, -21, 3, 1, 13, -12,
                     11, 3, -3, -1, 5, -2, -7, 5)
```

$\boxed{3}$

h) Adapt your program from the previous exercise to return the subsequence S for which the sum of all elements is maximal.

$\boxed{3}$

i) Write a very simple greedy algorithm to find the minimum in a sequence as follows. The algorithm starts at the first element. If the next element is larger, it returns the current element. Otherwise, it moves to the next element.

Test your algorithm on the following two R sequences. Does it find the global minimum?

```
seq1 <- abs(-10:10)
seq2 <- c(abs(-10:3)+1, abs(-2:5))
```