UNIVERSITY OF BIRMINGHAM | DUBAI دبي

LM Data Structures, Algorithms, and Databases (34140, 34141, 34139, 36989)

Exercise Sheet Solution

Week 01

Q1.

- Part A - Suppose you have a sorted list of 128 names, and you're searching through it using binary search. What's the maximum number of steps it would take? Answer: 7.
- Part B - Suppose you double the size of the list. What's the maximum number of steps now? Answer: 8.

Q2. Give the run time for each of these scenarios in terms of Big O.

- Part A- You have a name, and you want to find the person's phone number in the phone book. Answer: O(log n).
- Part B- You have a phone number, and you want to find the person's name in the phone book. (Hint: You'll have to search through the whole book!) Answer: O(n).
- Part C- You want to read the numbers of every person in the phone book. Answer: O(n).

Q3. Given a list of n numbers, create a straightforward python function **has_duplicates** to determine if there are any duplicate elements in the array. Additionally, calculate the time and space complexity for the worst-case scenario of your function.
Answer: In the worst case, the loop runs n times, and the check inside it takes O(n) time. Therefore, the overall time complexity is $O(n^2)$. In the worst case, if there are no duplicates, the seen_elements list will store all n elements of the input list. Therefore, the space complexity is O(n).

```
def has_duplicates(nums):
    seen_elements = []

    for num in nums:
        if num in seen_elements:
            return True
        seen_elements.append(num)

    return False

# Example usage:
numbers1 = [1, 2, 3, 4, 5, 6]
numbers2 = [1, 2, 3, 4, 5, 1]

result1 = has_duplicates(numbers1)
result2 = has_duplicates(numbers2)

print("List 1 has duplicates:", result1)
print("List 2 has duplicates:", result2)
```

Q4. Consider the following factorial program and let the input size be the value of n that is read. Counting one time unit for each assignment, read, and write statement, and one unit each time the condition of the while-statement is tested, compute the running time of the program

```
1. n = int(input("Enter a number: "))
2. i = 2
3. fact = 1
4. while i <= n:
5.     fact *= i
6.     i += 1
7. print(fact)
```

Answer: Lines (1) – (3) each time one time unit. For line (4), the test takes one unit, and it is executed n times. Lines (5) and (6) each take one unit and they are executed n- 1 times. Line (7) takes one unit. Thus, the total time taken by the program in is 3n+ 2 units. The running time of program is O(n).

Q5. What rule or criterion can be established to assess the overall time complexity of the selection statement, considering the functions f(n) and g(n) and the condition within the if statement?

```
if 1 == 2:
    # something with O(f(n)) complexity
else:
    # something with O(g(n)) complexity
```

Answer: The running time is O(g(n)). That is, the running time is that of the branch taken.

Q6.

- How can we express the number of iterations in a for-loop with the header for i in range(a, b+1) in terms of the initial value 'a' and the final value 'b'?
  Answer: The body of the for-loop for i in range(a, b+1) is iterated b - a+ 1 times, or 0 times if a > b.

- How can we express the number of iterations in for i in range(a, b-1, -1) in terms of the initial value 'a' and the final value 'b'?
  Answer: The body of the for-loop for i in range(a, b-1, -1) is iterated a-b+ 1 times, or 0 times if b> a.

Q7. In the context of Python, provide a big-O upper bound on the running time of the straightforward selection statement:

```
while C:
    # Empty body
```

Answer: If the condition 'C' evaluates to false, the running time of the while-loop is O(1). In the case where the condition is true, the while-loop executes indefinitely, and the running time is undefined.