Name: Rollno:

ESO207: Data Structures and Algorithms (Quiz 1)

28th August 2025

Total Number of Pages: 6 Time: 1 hr Total Points 45

Instructions

1. All questions are compulsory.
2. Answer all the questions in the question paper itself.
3. MCQ type questions can have more than one correct options.
4. The symbols or notations mean as usual unless stated.
5. You may cite and use algorithms and their complexity as done in the class.
6. Cheating or resorting to any unfair means will be severely penalized.
7. Superfluous and irrelevant writing will result in negative marking.
8. Using pens (blue/black ink) and not pencils. Do not use red pens for answering.

Question	Points	Score
1	3	
2	3	
3	3	
4	3	
5	3	
6	5	
7	3	
8	3	
9	4	
10	5	
11	5	
12	5	
Total:	45	

Helpful hints

1. It is advisable to solve a problem first before writing down the solution.
2. The questions are *not* arranged in increasing order of difficulty.

Question 1. (3 points) Let $f(x) = 5x^3 + 4x + 3$ and $g(x) = 8x^5 + 7x^2$. Which of the following statements are true? (Select all that apply)

- $\surd$ $f(x) = O(g(x))$
- ☐ $g(x) = O(f(x))$
- $\surd$ $f(x) = O(x^5)$
- $\surd$ $g(x) = O\big(\sqrt{x^{12} + 3x^9 + 8}\big)$
- ☐ $g(x) - f(x) = O(x^3)$

Question 2. (3 points) Which of the following operations on a doubly linked list can be performed in **constant** time if a pointer to the node is given, in addition? (Select all that apply)

- $\surd$ **Insert a new node before the given node**
- $\surd$ **Delete the given node**
- ☐ Find the middle node of the list (Pointer to head is given)
- ☐ Insert a node at the end of the list (Pointer to head is given)
- $\surd$ **Update the data of the given node**

Question 3. (3 points) Which of the following statements about a local minima in a 2D grid are correct? (Select all that apply)

- $\surd$ **A local minima is always smaller than all its neighbors up, down, left, and right.**
- ☐ Local minimum of the grid is always the global minima.
- $\surd$ **A local minima can be located on the boundary of the grid.**
- $\surd$ **If all the grid entries are distinct, there are always local minima.**
- ☐ The smallest element in any column is always a local minima.

Question 4. (3 points) Consider the following expression

$$(8 - 3) * 2\hat{}(1 + 2) + 7/(5 - 2) - 4.$$

What is the maximum number of symbols that may appear simultaneously on the operator stack during evaluation using the operator-stack algorithm (assume the usual precedence: ˆ highest and right-associative; $*$ and / next, left-associative; + and − lowest, left-associative)?

- ☐ 1
- ☐ 2
- ☐ 3
- $\surd$ 4
- $\surd$ 5

Question 5. (3 points) For the power operator ˆ which is right-associative, how does the expression evaluation algorithm ensure correct evaluation order?

- ☐ By assigning it the same inside and outside priority
- $\surd$ **By assigning a higher outside priority than inside priority**
- ☐ By assigning a higher inside priority than outside priority
- ☐ Right associativity is not handled by the algorithm

Question 6. (5 points) Fill in the blanks to complete the implementation of a queue using two stacks.

```
struct Queue:
    s1 = empty stack
    s2 = empty stack

    function enqueue(x):
        // Push item into the first stack
        s1.push(x)

    function dequeue():

        if s1.empty() AND s2.empty() :
            return -1

        // If s2 is empty, move elements from s1 to s2
        if s2.empty():
            while not s1.empty() :

                s2.push(s1.top())
                s1.pop()

        x = s2.top()
        s2.pop()
        return x
```

Question 7. (3 points) Consider the following algorithm for finding a local minimum of grid.

```
Algorithm 1: Explore(Grid)
  Let $c$ be any entry to start with;
  while $c$ is not a local minima do
  |   $c \leftarrow$ neighbor of $c$ storing smallest value;
  end
  return $c$
```

Construct a 5×5 grid for which the algorithm visits atleast 15 elements in the worst case. Briefly explain a strategy for which atleast $\Omega(n^2)$ elements will be visited.

Solution:
$$\begin{bmatrix} 20 & 19 & 18 & 17 & 16 \\ 100 & 99 & 98 & 97 & 15 \\ 10 & 11 & 12 & 13 & 14 \\ 9 & 96 & 95 & 94 & 93 \\ 8 & 7 & 6 & 5 & 4 \end{bmatrix}$$
(Any similar matrix)

If the algorithm starts at cell (0,0), it will always proceed to the neighbor with the smallest value, following this path: (0,0)→(1,0)→(2,0)→(3,0)→(4,0)→(4,1)→(4,2)→(4,3)→(4,4), and so on.
Startegy: By extending this pattern, the strategy for forcing the algorithm to visit $\Omega(n^2)$ elements is to arrange the values so that the decreasing path zig-zags through nearly the entire grid before reaching a local minimum, while considering alternate rows as shown in the given example.

Question 8. (3 points) Consider the $O(n \log n)$ algorithm for finding a local minimum in a $n \times n$. If we allow the presence of repeated values would the algorithm still work correctly? Give a brief explanation of your answer.

Solution: No, the $O(n \log n)$ algorithm for finding a local minimum in an $n \times n$ grid does not necessarily work if repeated values are allowed. This is because the standard definition of a local minimum used in the correctness proofs of these algorithms requires that the chosen cell is strictly smaller than all its immediate neighbors. Therefore

1. If repeated values are present, it's possible that there is no entry strictly smaller than all its immediate neighbors, meaning that there is no local minimum.
2. The $O(n \log n)$ algorithm relies on moving towards strictly smaller values. If all neighbors of a cell have the same value as the current cell, the condition to move or halt may never be triggered.

Question 9. Consider a generalized Fibonacci sequence defined as follows:

- The first k terms are all 1.
- For $n > k$,
$$f_n = f_{n-1} + f_{n-2} + \cdots + f_{n-k}.$$
where f_n denoted the n^{th} term of the sequence.

(a) (2 points) Consider a $k \times k$ matrix T_k such that

$$\begin{pmatrix} f_n \\ f_{n-1} \\ \vdots \\ f_{n-k+1} \end{pmatrix} = T_k \begin{pmatrix} f_{n-1} \\ f_{n-2} \\ \vdots \\ f_{n-k} \end{pmatrix}.$$

Give the matrix T_k.

Solution: $F_k = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}_{k \times k}$.

(b) (2 points) What is the complexity of computing f_n? Give your answer in the big O notation as a function of both n and k?

Solution: $O(k^3 \log(n-k))$

Explanation: We have $\begin{pmatrix} f_n \\ f_{n-1} \\ \vdots \\ f_{n-k+1} \end{pmatrix} = T_k^{n-1} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$.

Therefore Time complexity $= O(\text{Matrix exponentiation} + k) = O(k^3 \log(n-k) + k)$
Multiplying two $k \times k$ matrices takes $O(k^3)$ time.

Question 10. Consider an alternative algorithm to findal minima in a grid, where we alternately bisect along rows and columns.

Algorithm 2: FindLocalMinimum($grid$)

```
Let $top \leftarrow 0$, $bottom \leftarrow m-1$, $left \leftarrow 0$, $right \leftarrow n-1$;
Set $searchByRow \leftarrow$ true;
while $top \leq bottom$ and $left \leq right$ do
    if $searchByRow$ then
        Choose the middle row between $top$ and $bottom$;
        In that row, find the column containing the smallest value;
        if this element is smaller than all of its up/down/left/right neighbors then
            Return its position as the local minimum;
        end
        if the element above is smaller then
            Discard all rows below and including the current row;
        else
            Discard all rows above and including the current row;
        end
    else
        Choose the middle column between $left$ and $right$;
        In that column, find the row containing the smallest value;
        if this element is smaller than all of its up/down/left/right neighbors then
            Return its position as the local minimum;
        end
        if the element to the left is smaller then
            Discard all columns to the right and including the current column;
        else
            Discard all columns to the left and including the current column;
        end
    end
    Flip $searchByRow$ to the opposite value;
end
Return "no local minimum found";
```

(a) (3 points) Construct a grid storing distinct numbers where the above alternating row/column bisection algorithm fails to find a local minimum.

Solution: Consider the following Matrix- $\begin{bmatrix} 300 & 301 & 302 & 303 & 304 \\ 200 & 201 & 202 & 203 & 204 \\ 401 & 20 & 90 & 200 & 50 \\ 402 & 19 & 130 & 112 & 60 \\ 403 & 295 & 120 & 114 & 113 \end{bmatrix}$

(b) (2 points) Perform a step-by-step dry run of the algorithm on your counterexample grid that you gave in previous part.

Solution:

1. Initialize: top = 0, bottom = 4, left = 0, right = 4, searchByRow = true
2. Search by Row: Middle row is $[401, 20, 90, 200, 50]$, with Minimum Value 20 - not a local minima

3. Below is smaller(19) $\rightarrow$ discard all rows above and including current row. Update top = midRow + 1 = 3, searchByRow = false.
4. Search by Column: Middle column $[130, 120]$. Minimum value - 120- Not a local minimum.
5. Right is smaller $\rightarrow$ discard all columns to the left of current column. Update left = midCol + 1 = 3 , searchByRow = true.
6. Search by Row: Middle Row: [112, 60]. Minimum=60- Not a local minima.
7. Now after update top=3, bottom=2 $\rightarrow$ loop ends because top >bottom. Return *"No local minima found"*

Therefore no local minima is found.

Question 11. (5 points) Climbing Stairs: You are given a staircase with n steps. A person starts at the bottom (step 0) and wants to reach the top (step n). At each move, the person can climb either 1 step or 2 steps.

Give a $O(logn)$ pseudocode for the function $ClimbingStairs(n)$ that outputs the total number of distinct ways the person can reach the top.

Example: For $n = 3$, the expected output is 3, because there are three ways to climb 3 steps: $(1, 1, 1)$, $(1, 2)$, and $(2, 1)$.

Solution: The problem is equivalent to finding the n^{th} fibonacci number since the number of ways to reach n^{th} step can be given by the following recursion- $f_n = f_{n-1} + f_{n-2}$

Question 12. (5 points) You are given two binary search trees (BSTs), all elements in both the BSTs are distinct. Given an integer x, find the number of pairs (a, b) such that:

- a is an element from the first BST,
- b is an element from the second BST,
- $a + b = x$.

Design an algorithm that finds the count of such pairs in $O(n_1 + n_2)$ time, where n_1 and n_2 are the sizes of the two BSTs.

Give a pseudocode for your solution that clearly describes how you leverage the BST properties and achieve the required time complexity.

Solution: We traverse the first BST in ascending order using iterative inorder traversal, and the second BST in descending order using reverse inorder traversal. At each step, we check the sum:

$$\text{If } node_1 + node_2 = x, \quad \text{we increment the pair count and move both pointers forward.}$$

$$\text{If } node_1 + node_2 < x, \quad \text{we advance the pointer in the first BST (to get a larger value).}$$

$$\text{If } node_1 + node_2 > x, \quad \text{we advance the pointer in the second BST (to get a smaller value).}$$