Suez Canal University
Faculty of science
Department of Mathematics

# Midterm Exam

## 1st Term: 2022 /2023



| Course: Design and Analysis of Algorithms | | Course code: MT364 |
|---|---|---|
| Level:3 | Total Marks: 30 | Date: 20/11/2022 |
| Total page: 4 | Lecturer: Dr. Dina El-Manakhly | Time allowed: 1 hour |

## Question 1(15 marks)

### Choose the correct answer:

1- Where is linear searching used?
a) When the list has only a few elements
b) When performing a single search in an unordered list
c) Used all the time
d) When the list has only a few elements and When performing a single search in an unordered list

2- What is the worst case for linear search?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(1)

3- Which of the following is a disadvantage of linear search?
a) Requires more space
b) Greater time complexities compared to other searching algorithms
c) Not easy to understand
d) Not easy to implement

4- What is the advantage of recursive approach than an iterative approach?
a) Consumes less memory
b) Less code and easy to implement
c) Consumes more memory
d) More code has to be written

5- In Binary search, given an input **arr** = {2,5,7,99,899}; key = 899; What is the level of recursion?
a) 5
b) 2
c) 3
d) 4

level 1: mid = 7
level 2: mid = 99
level 3: mid = 899 (this is the key).

6- Binary Search can be categorized into which of the following?
a) Brute Force technique
b) Divide and conquer
c) Greedy algorithm
d) Dynamic programming

7- In the following scenarios, when will you use selection sort?
a) The input is already sorted
b) A large file has to be sorted
c) Large values need to be sorted with small keys
d) Small values need to be sorted with large keys

8- What is the worst case complexity of selection sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) $O(n^2)$

9- Which of the following is not O(n^2)?
a) (15^10) * n + 12099
b) n^1.98
c) n^3 / (sqrt(n))
d) (2^20) * n

10- What is the auxiliary space complexity of merge sort?
a) O(1)
b) O(log n)
c) O(n)
d) O(n log n)

11- Which of the following method is used for sorting in merge sort?
a) merging
b) partitioning
c) selection
d) exchanging

12- How many sub arrays does the quick sort algorithm divide the entire array into?
a) one
b) two
c) three
d) four

13- Apply Quick sort on a given sequence 23, 12, -7, 16, 18, 35, 35, 28, 5. What is the sequence after first phase, pivot is last element?
a) 23, 18, -7, 16, 12, 35, 35, 28, 5
b) 5, 18, -7, 16, 12, 35, 35, 28, 23
c) -7, 12, 23, 16, 18, 35, 35, 28, 5
d) -7, 5,18, 23, 16, 12, 35, 35, 28

14- What is the complexity of the following code?

```
sum=0;
    for (i=1; I <= n; i*=2)
        for(j=1; j<=n;j++)
            sum++;
```

a) $O(n^2)$
b) O (n log n)
c) O(n)
d) O (n log n log n)

15-The exact running time for the following code snippet is

```
int n=2,m=3;

for(int i=0;i<n;i++)
{
    for(int j=0;j<m;j++)
    {
        for(int k=0;k<m;k++)
        {
            cout<<k;
        }
    }
}
```

a) $T(n)= 2+4n+4mn+3nm^2$
b) $T(n)= 2+3n+4mn+3nm^2$
c) $T(n)= 2+4mn+3nm^2$
d) $T(n)= 2+4n+4m^2n+3n^2m$

==========

## Question2 (5 marks)

Show that $3n^2-100n+6 = \Theta(n^2)$

| $3n^2-100n+6 = O(n^2)$ | $3n^2-100n+6 = \Omega(n^2)$ |
|---|---|
| $f(n) \leq cg(n)$ | $f(n) >= cg(n)$ |
| $3n^2-100n+6 <= cn^2$ | $3n^2-100n+6 >= cn^2$ |
| C=3, n>=1 | C=2, n>=100 |

==========

## Question3 (10 marks)

**Write the pseudo code for**



1) Linear search using recursion
2) Iterative binary search

```
public void linSearch(int[] arr, int first, int last, int key)
{
        if(first == last)
        {
                System.out.print("-1");
        }
        else
        {
                if(arr[first] == key)
                {
                        System.out.print(first);
                }
                else
                {
                        linSearch(arr, first+1, last, key);
                }
        }
}
```

```
public static int iterative(int arr[], int key)
{
        int low = 0;
        int mid = 0;
        int high = arr.length-1;
        while(low <= high)
        {
                mid = low + (high - low)/2;
                if(arr[mid] == key)
                {
                        return mid;
                }
                else if(arr[mid] < key)
                {
                        low = mid + 1;
                }
                else
                {
                        high = mid - 1;
                }
        }
        return -1;
}
```