National University of Computer & Emerging Sciences, Karachi
Fall-2019 Department of Computer Science
Mid Term-1
24th September 2018, 11:00 AM – 12:00 PM

Course Code: CS302	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Waqas Sheikh, Zeshan Khan	
Student Roll No:	**Section:**

Instructions:

- Return the question paper.
- Read each question completely before answering it. There are **5 questions** on **2 pages.**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.

Time: 60 minutes. **Max Marks: 12.5**

Question # 1 **[1.5 marks]**

Are these following statement true or false? Prove your answer by computing the values of n_0, c_1, c_2 or by contradiction. [Θ is Theta]
[Remove One]

A. $n^2 + 4^5 = \Theta(n^2)$
B. $2\^n + 2n = \Omega(n^2)$
C. $2n + 4^{\log_2 n} - 5 = \Theta(n^2)$

Question # 2 **[1.5 marks]**

Question # 3 **[1.5 marks]**

(a) What is meant by Design and Analysis of Algorithms?
(b) List two topics in Computer Science that are more important than studying computer program performance.
(c) Write down the formal definition of Small-Oh Notation i.e. in terms of f(n) and g(n)

Question # 4 **[4 marks]**

Given a sorted array containing duplicates, Design efficient algorithm using divide & conquer approach to find the frequency of each element. For example, Input = {1,1,1,5,5,6,6,8,9}. Output:
1 appears 3 times
5 appears 2 times
6 appears 2 times
8 appears 1 time
9 appears 1 time

Question # 5 **[2+1+1.5=4.5 marks]**

Solve the following recurrences to compute the time complexity.

A. $T(n) = 2T(n-1) + 1$ [Master Theorem]
B. $T(n) = 32T\left(\frac{n}{4}\right) - n^2 logn$
C. $T(n) = 7T\left(\frac{n}{3}\right) + n^2$

BEST OF LUCK

National University of Computer & Emerging Sciences, Karachi
Fall-2019 Department of Computer Science
Mid Term-1
24th September 2018, 11:00 AM – 12:00 PM

Course Code: CS302	**Course Name:** Design and Analysis of Algorithm
Instructor Name / Names: Dr. Muhammad Atif Tahir, Waqas Sheikh, Zeshan Khan	
Student Roll No:	**Section:**

Instructions:

- Return the question paper.
- Read each question completely before answering it. There are **4 questions** on **2 pages.**
- In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.

Time: 60 minutes. **Max Marks: 12.5**

Question # 1 **[1 marks]**

Are these following statement true or false? Prove your answer by computing the values of n_0, c_1, c_2 or by contradiction. [Θ is Theta]

$\boldsymbol{n^2 + 4^5 = \Theta(n^2)}$
True
$c_1 n^2 \leq n^2 + 4^5 \leq c_2 n^2$
$c_1 \leq 1 + 4^5/n^2 \leq c_2$
$for\ n_0 = 1$
$c_1 \leq 1 + 1024 \leq c_2$
$c_1 \leq 1025 \leq c_2$
$for\ n = \infty$
$c_1 \leq 1 + 0 \leq c_2$
$c_1 \leq 1 \leq c_2$

$\boldsymbol{2 + 2n = \Omega(n^2)}$
False
$n! + 2n \leq c_1 n^2$
$\frac{n!}{n^2} + \frac{2}{n} \leq c_1$
$for\ n = 1$
$1 + 2 \leq c_1$
$3 \leq c_1$
$for\ n = \infty$
$\lim_{n\to\infty} \frac{n!}{n^2} = \infty$
$\infty + 0 \leq c_1$
$\infty \leq c_1$ there does not exists a real positive number greater than infinity.

$\boldsymbol{2n + 4^{\log_2 n} - 5 = \Theta(n^2)}$

True

$c_2n^2 \le 2n + 4^{\log_2 n} - 5 \le c_2n^2$

$c_2n^2 \le 2n + 2^{\log_2 n^2} - 5 \le c_2n^2$

$c_2n^2 \le 2n + n^2 - 5 \le c_2n^2$

$c_2 \le 2/n + 1 - \frac{5}{n^2} \le c_2$

$for\ n_0 = 4$

$c_2 \le \frac{2}{4} + 1 - \frac{5}{4} \le c_2$

$c_2 \le \frac{1}{4} \le c_2$

$for\ n = \infty$

$c_2 \le 0 + 1 - 0 \le c_2$

$c_2 \le 1 \le c_2$

$\mathbf{\log_2 4^n + 2n - 5 = \Theta(n^2)}$

False

$c_1n^2 \le \log_2 4^n + 2n - 5 \le c_2n^2$

$c_1n^2 \le \log_2 2^{2n} + 2n - 5 \le c_2n^2$

$c_1n^2 \le 2n + 2n - 5 \le c_2n^2$

$c_1n^2 \le 4n - 5 \le c_2n^2$

$c_1 \le \frac{4}{n} - \frac{5}{n^2} \le c_2$

$for\ n_0 = 2$

$c_1 \le \frac{4}{2} - \frac{5}{4} \le c_2$

$c_1 \le \frac{3}{4} \le c_2$

$for\ n = \infty$

$c_1 \le \infty - \infty \le c_2$

There doesn't exist any real positive value for c2.

Question # 2 **[0.25*8=2 marks]**

Prove the accuracy of the Dijkstra algorithm for the computation of single source shortest path with assumption of the graph of only positive weighted edges.

Solution

Initialization: Initially, $S = \emptyset \wedge Q = G.V \wedge \forall x : x \in S\ d(x) = minimum$

Maintenance: at each iteration another vertex is added into S with minimum cost from source. So at ith iteration, $|S| = i \wedge |Q| = |G.V| - i \wedge \forall x : x \in S\ d(x) = minimum$

Termination: At the termination of the algorithm, $Q = \emptyset$ Since $Q = V - S, S = V$.

Question # 3 **[2 marks]**

(a) What is meant by Design and Analysis of Algorithms?

(b) List two topics in Computer Science that are more important than studying computer program performance.
(c) Write down the formal definition of Big-Oh Notation i.e. in terms of f(n) and g(n)

Solution

(a) The analysis of algorithm is the theoretical study of computer program performance and resource usage. Algorithm design include creating an efficient algorithm to solve a problem in an efficient way using minimum time and space.
(b) Correctness, Security, Stability etc
(c)

$$O(g(n)) = \{f(n) : \text{ there exists positive constants } c \text{ and } n_0 \text{ such that } f(n) \leq cg(n) \text{ for all } n \geq n_0\}$$

<u>Question # 4</u> **[2 marks]**

```
#include <iostream>
#include <unordered_map>
using namespace std;
// Function to find frequency of each element in a sorted array
void findFrequency(int arr[], int n, unordered_map<int, int>
&count)
{
	// if every element in the subarray arr[0..n-1] is equal,
	// then increment the element count by n
	if (arr[0] == arr[n - 1]) {
		count[arr[0]] += n;
		return; }
	// divide array into left and right sub-array and recur
	findFrequency(arr, n/2, count);
	findFrequency(arr + n/2, n - n/2, count);
}
```

```
// Find Frequency of each element in a sorted array containing duplicates
int main()
{       int arr[] = { 2, 2, 2, 4, 4, 4, 5, 5, 6, 8, 8, 9 };
        int n = sizeof(arr) / sizeof(int);
        // find frequency of each element of the array and store it in map
        unordered_map<int, int> map;
        findFrequency(arr, n, map);

        // print the frequency
        for (auto &p: map) {
                cout << p.first << " occurs " << p.second << " times\n";
        }
        return 0;
}
```

Question # 5 **[2+1.5=3.5 marks]**

$$T(n) = T(n-1) + logn$$

$$log_a pq = log_a p + log_a q$$

$$T(n) = 2T(n-1) + 1$$

$$a + ar + ar^2 + ar^3 + \cdots + a^k = \frac{a(r^{k+1} - 1)}{r - 1}$$

$T(n) = 2T(n-1) + 1$

$= 2[2T(n-2)+1]+1$

$4T(n-2)+2+1$

$= 4[2T(n-3)+1]+2+1$

$= 8T(n-3)+4+2+1$

$= 2^3T(n-3)+2^2+2^1+2^0$

Repeat K Times.

$= 2^kT(n-k)+2^{(k-1)}+2^{(k-2)}+2^{(k-3)}+\cdots 2^1+2^0$

$n-k=0 \;;\; n=k.$

$= 2^kT(0)+2^{k-1}+2^{k-2}+\cdots 2^1+2^0$

$= 2^k+2^{k-1}+2^{k-2}+\cdots 2^1+2^0$

$= 2^n+2^k-1$

$= \Theta(2^n)$

$$T(n) = 32T\left(\frac{n}{4}\right) - n^2 logn$$

Recurrence: $T(n) = 32\,T(n/4) + \Theta(n^2)$.

Solution: $T \in \Theta(n^{\log_4 32}) \approx \Theta(n^{2.500})$.

$$T(n) = 7T\left(\frac{n}{3}\right) + n^2$$

Recurrence: $T(n) = 7\,T(n/3) + \Theta(n^2)$.

Solution: $T \in \Theta(n^2)$.

<u>BEST OF LUCK</u>