National University of Computer and Emerging Sciences 0068

Islamabad Campus

Design and Analysis of Algorithms (CS2009)

Final Exam PART-B

Course Instructor(s):
Ms. Noor ul Ain, Mr. Muhammad Owais Idrees, Ms. Hira Mastoor

Section(s): (A,B,C,D,E,F,G)-CS

Total Time (Hrs): 3
Total Marks: 130
Total Questions: 5

Date: Dec 18, 2025

23I-0708 Roll No

CS-5B Course Section

Student Signature

Do not write below this line.

Attempt all the questions

Instructions

- Both parts (PART-A & PART-B) will be given together. You have 3 hours to solve both parts in any order.
- Attempt **all questions** in the **space provided** on this question paper.
- Verify that you have **Eight (08)** printed page of the PART-B question paper including this page.
- Use the **extra sheet only for rough work**. Any answers written on the extra sheet **will not be evaluated**.
- **Using Pencil is prohibited. Any part done with the pencil will not be marked and cannot be claimed for rechecking.** Write **neatly, concisely, and within the given space.**
- Overwriting and cutting in the final answer on the question paper will result in a zero score and cannot be claimed for rechecking.
- **Sharing calculators or any other tools is prohibited.**

Question No.	Q1	Q2	Q3	Q4	Q5	Total
Total Marks	30	25	25	20	30	130
Marks Obtained	27.5	20	25	20	15	107.5

[CLO 2: Analyze the time and space complexity of different algorithms by using standard asymptotic notations for recursive and nonrecursive algorithms.]

Q1: You are provided with 5 different code snippets and four recurrence relations. Please write the time complexities them space given below. **[4x5 + 2.5x4 =30 marks]**

Answer:

1.	$O(n)$	6.	$O(n)$
2.	$O(n^3)$	7.	$O(n^3 \log^2 n)$
3.	$O(\log n)$	8.	$O(n \log^{1.5} n)$
4.	$O(n^2)$	9.	$O(n \log^3 n)$
5.	$O(n)$		

7.5

Code Snippets	Recurrence Relation

Snippet : 1

```
for (int i = 1; i <= n; i *= 2) {
    for (int j = 1; j <= i; j++) {
        for (int k = 1; k <= 10; k++) {
            printf("*");
        }
    }
}
```

O(n)

Recurrence : 6

$$T(n) = 3T\left(\frac{n}{4}\right) + n$$

$n^{\log_4 3}$

O(n)

Snippet : 2

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
        for (int k = 1; k <= i; k++) {
            printf("*");
        }
    }
}
```

$O(n^3)$

Recurrence : 7

$$T(n) = 8T\left(\frac{n}{2}\right) + n^3 \log n$$

$\log_2 8$

n^3

Snippet : 3

```
for (int i = 1; i <= n; i *= 2) {
    for (int j = 1; j <= i; j += i) {
        printf("*");
    }
}
```

O(log n)

Recurrence : 8

$$T(n) = 2T\left(\frac{n}{2}\right) + n(\log n)^{0.5}$$

~~log~~

Snippet : 4

```
for (int i = 1; i <= n; i=i*2) {
    for (int j = 1; j <= n; j++) {
        for (int k = 1; k <= i; k++) {
            printf("*");
        }
    }
}
```

n^2 ~~logn~~

Recurrence : 9

$$T(n) = 3T\left(\frac{n}{2}\right) + nlog^4 n$$

$O(n\sqrt{n})$

Snippet : 5

```
for (int i = 1; i <= n; i++)
    if(((i/2)%2)==0)
        for (int j = i; j <= i; j++)
            printf("*");
```

O(n)

[CLO 1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program]

Q2: For the given 6 items below, find most valuable subset of the items that fit into the 0-1knapsack of capacity 6? **[25 marks]**

item	weight	value	
1	3	Rs. 25	8.33 ✓
2	1	Rs. 12	12 ✓
3	4	Rs. 30	7.5
4	2	Rs. 14	7
5	5	Rs. 40	~~10~~ 8
6	2	Rs. 18	9 ✓

a. Compute the maximum total cost achieved using the greedy approach along with the item selected. [5 marks]

total profit = 55   items = 1, 2, 6

2

b. Compute the maximum total cost achieved using Dynamic Programming along with the item selected. [15 marks]

20

Fill the table using the recursive definition for solving the 0-1 Knapsack problem;
[Note: Use columns to show bag capacity]

Item\Capacity	0	1	2	3	4	5	6
v w ~~25~~ ~~3~~ 0	0	0	0	0	0	0	0
25 3 1	0	0	0	25	25	25	25
12 1 2	0	12	12	12	37	37	37
30 4 3	0	12	12	12	37	42	42
14 2 4	0	12	14	26	37	42	54
40 5 5	0	12	14	26	37	42	52
18 2 6	0	12	18	30	37	44	55

5

Also show complete trace of item selection. [5 marks]

dP(6,6) ≠ dP(5,6) => 6 -> 1 , dP(5,4) = dP(4,4) -> 5 = 0
dP(4,4) = dP(3,4) => 4 = 0
dP(3,4) = dP(2,4) => 3 = 0
dP(2,4) ≠ dP(1,4) => 2 = 1
dP(1,3) ≠ dP(0,3) => 1 = 1

[1 2 3 4 5 6]
[1 1 0 0 0 1]

5

[CLO 1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program]

Q3: Provide the order of multiplication and total number of scalar multiplications for the following matrices **[10+10+5=25 marks]**

P0=2, P1=30, P2=4, P3=10, P4=5

Cost Table

	A1	A2	A3	A4
A1	0	240	320	420
A2	x	0	1200	800
A3	x	x	0	200
A4	x	x	x	0

Solution table

	A1	A2	A3	A4
A1	0	1	2	3
A2	x	0	2	2
A3	x	x	0	3
A4	x	x	x	0

Total Scalar Multiplication required: 420

Please write down the final bracket formation of the above provided matrices [5]

(((A1 × A2) × A3) × A4) .

[CLO 1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program]

Q4: Read the problem carefully and answer the following questions **[20 marks]**

Given two text strings A of length n and B of length m, you want to transform A into B with a minimum number of operations of the following types:

1. Delete a character from A
2. Insert a character into A
3. Replace some character in A with a new character.

The minimal number of such operations required to transform A into B is called the edit distance between A and B.

Part a: Write the recursive call that computes the minimum distance required to transform one string to another. Assuming the strings are X [1…i] and Y [1…j]. The base case and match recursive call are provided.

Base Cases

$$ED(0, j) = j \qquad \text{(insert all } j \text{ characters)}$$

$$ED(i, 0) = i \qquad \text{(delete all } i \text{ characters)}$$

Recursive Step

If the current characters match:

$$X[i-1] = Y[j-1] \quad \Rightarrow \quad ED(i, j) = ED(i-1, j-1)$$

You just have to provide the recursive formula for string mismatch *(Hint: This recursive call caters to all operations of insert, delete, and replace)*. Make sure the recursive formula is correct, as you will convert this same formula into a DP table in the next part. **[5]**

$$DP[i,j] = \begin{cases} DP[i-1,j-1] & \text{if } (x[i] == Y[j]) \\ \min\{DP[i-1,j-1], DP[i-1,j], DP[i,j-1]\}+1 & \text{if } (x[i] \neq Y[j]) \end{cases}$$

+5

Part b: Convert the string "Heart" into "Smart". The DP table would be as follows. **[15]**

	" "	H	E	A	R	T	
" "	0	1	2	3	4	5	2
S	1	1	2	3	4	5	3
M	2	2	2	3	4	5	3
A	3	3	3	2	3	4	2
R	4	4	4	3	2	3	3
T	5	5 6	5	4	3	2	2

20

National University of Computer and Emerging Sciences

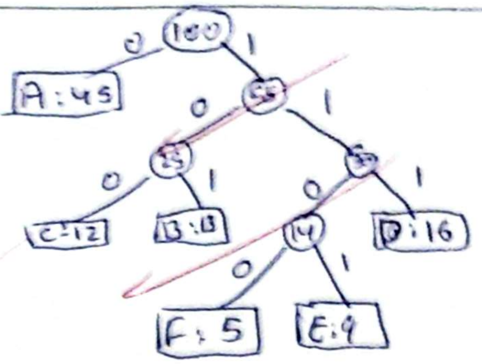
Islamabad Campus

[CLO 1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program]

Q5. You are designing a **lossless compression module** for a real-time data transmission system used in IoT sensors. The system must minimize bandwidth usage while ensuring fast encoding and decoding.

A stream of symbols with the following observed frequencies: **[6+5+5+2+12=30 marks]**

Sr.	Symbol	Frequency	Variable Len Codes	Tree
1	A	45	0	
2	B	13	101	
3	C	12	100	
4	D	16	111	
5	E	9	1101	
6	F	5	1100	



1. Construct the Huffman Tree using the standard greedy algorithm. (fill above table)
2. Generate the Huffman codes for each symbol. (fill above table)
3. In the target hardware, the decoder can only handle codes of **fixed maximum length, i.e., 4 bits**. If they do, propose a modification in current algorithm to solve the conflict.
   Answer:
   no modification Required since the maximum code length ~~for~~ of our codes is ~~for~~ 4 i.e for E and F.
4. What will be the limitation of new modified approach(fixed length).
   Answer:
   if the codes were of Fixed length, we need more storage space to save the compression data
5. Compare the compression efficiency of all three techniques by calculating the number of bits required(all necessary bits) for the above given dataset:

Table data + code data
66 224
72 400

- Fixed-length encoding : 472
- Original Huffman encoding : 290
- Modified (len-limited) Huffman encoding : 290