Master of Computer Applications

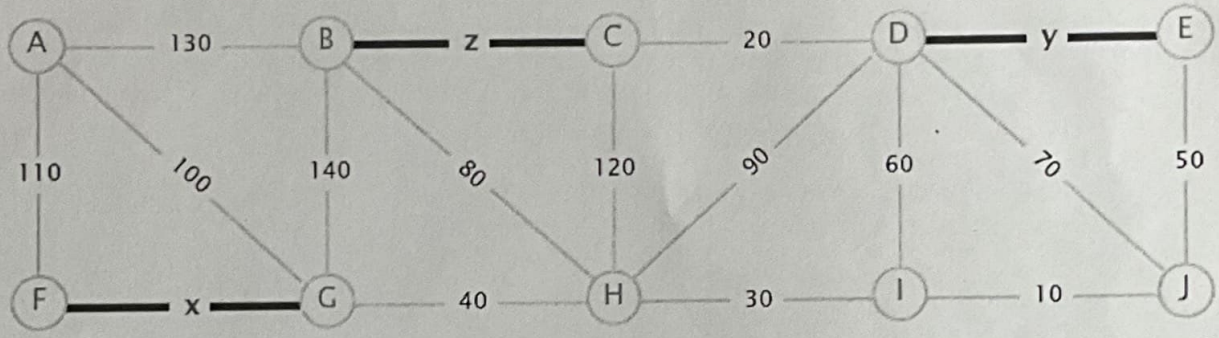
MCAC 201: Design and Analysis of Algorithms
Unique Paper Code: 223421201
Semester II (Reg/ER/Imp/Ex. Students)
May-June, 2024

Time: Three Hours **Maximum marks: 70**

Note: Answer all the questions. Each question carries equal marks.

1. (a) Consider the following set of symbols and their frequencies. [6]
$\{a : 0.15,\ b : 0.25,\ c : 0,\ d : 0.1,\ e : 0.12,\ f : 0.18,\ g : 0,\ h : 0.2\}$ –
Perform the following:
(i) Construct the Huffman code and list the codeword for each symbol. When combining two trees, ensure that the tree with the smaller aggregate frequency is placed on the 0 branch.
(ii) Draw the corresponding Huffman tree.
(iii) List the merged operations performed in the sequence.

(b) Suppose that a minimum spanning tree (MST) of the following edge-weighted graph contains the edges with weights x, y, and z.



What will be the maximum value of $x + y + z$? Show all the intermediate steps. [8]

2. (a) Is it feasible to solve the 0-1 knapsack problem using a greedy strategy? Explain your answer with the help of an example. [2]

(b) Given the following jobs, where each job i has a start time s_i, a finish time f_i, and a weight w_i. Compute the maximum weight subset of compatible jobs. Additionally, provide the recurrence relation used to solve the problem. [6]

i	1	2	3	4	5	6	7	8
s_i	1	3	0	4	3	5	6	8
f_i	4	5	6	7	8	9	10	11
w_i	3	2	1	5	8	4	6	4

(c) Using the dynamic programming algorithm, determine the minimum number of multiplications to calculate the matrix product ABCD, where the dimensions of A, B, C and D are given as: $A : 10 \times 5, B : 5 \times 20, C : 20 \times 10, D : 10 \times 5$. Show all intermediate steps. [6]

3. (a) Consider a situation where there is no predefined function to calculate power, and the task is to calculate x^n where x can be any number and n is a positive integer. What can be the best possible time complexity of the power function in this scenario? [2]

(b) Solve the following recurrence relation using the recursion tree approach. [5]

$$T(n) = T(n/4) + T(n/2) + cn^2$$
$$T(1) = c$$
$$T(0) = 0$$

(c) Given the text $T =$ *"abdcababdcabdcband"* and the pattern $P =$ *''abdcabd"*, use Knuth-Morris-Pratt (KMP) algorithm to find all occurrences of the pattern P in the text T. Show the computation of the next array used in the KMP algorithm, which encapsulates knowledge about how the pattern matches against shifts of itself. Explain each step of the matching process. [7]

4. (a) Suppose an array of eight integers is being sorted using quicksort, and the first partitioning is just completed resulting in an array as follows:

$$2, 5, 1, 7, 9, 12, 11, 10$$

Which of the following statement(s) is(are) correct? Justify your answer.

(i) The pivot could be either the 7 or the 9.

(ii) The pivot could be the 7, but it is not the 9.

(iii) The pivot is not the 7, but it could be the 9.

(iv) Neither the 7 nor the 9 is the pivot. [2]

(b) Given a sorted array of distinct integers $A[1, ..., n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(\log n)$ to find out whether there is an index i for which $A[i] = i$. [5]

(c) Write a divide-and-conquer algorithm that efficiently finds the maximum difference between any two elements in a given array of n numbers (not necessarily distinct). For example, given the input array $A = [10, 5, -2, 40, 18]$, your algorithm should return 42. Additionally, provide the recurrence relation and the time complexity of your algorithm. [7]

5. (a) What value does the following function return? Provide your answer as a function of n. Also, determine the worst-case time complexity using an iterative approach. [4]

```
function conundrum(n)
    r := 0;
    for i := 1 to n do
        for j := i + 1 to n do
            for k := i + j - 1 to n do
                r := r + 1
    return(r)
```

(b) Consider the following algorithm for finding the maximum element in an array of n distinct integers:

```
function randomized_maximum(arr)
    n := length(arr)
    max_element := arr[0]
    for i from 1 to n do
        index := random_integer(0, n-1)  // Generate a random index
        if arr[index] > max_element then
            max_element := arr[index]
    return max_element
```

Discuss whether this algorithm always finds the maximum element in the array. If not, under what conditions does it succeed? Determine the probability that the algorithm returns will not find the maximum element. [5]

(c) Write down the steps you would take to sort the array $A = [170, 45, 75, 90, 802, 24, 2, 66]$ using Radix sort. [5]