9205

[This question paper contains 8 printed pages.]

**Your Roll No.....**

**Sr. No. of Question Paper : 2208** **IC**

Unique Paper Code : 32341401

Name of the Paper : Design & Analysis of Algorithms

Name of the Course : **B.Sc. (H) Computer Science**

Semester : IV

Duration : 3 Hours Maximum Marks : 75

**Instructions for Candidates**

1. Write your Roll No. on the top immediately on receipt of this question paper.
2. Question No. 1 is compulsory.
3. Attempt any **four** of Questions Nos. **2** to **7.**

1. (a) Consider an input of $n$ numbers that are all equal. What would be the running time of the following algorithms for the input :

(i) Merge Sort

(ii) Heap Sort (4)

P.T.O.

(b) Give a recurrence for the running time of the following algorithms to sort $n$ elements.

(i) Maxheapify

(ii) Quicksort (2)

(c) Can Insertion Sort be used as an intermediate sort for Radix sort? Explain why or why not. (3)

(d) Compare the worst case running times of the following operations with respect to red-black trees and binary search trees.

(i) Searching a given key

(ii) Inserting a given key (4)

(e) Recall that the usual implementation of quicksort makes two recursive calls. Consider a variant that optimizes on stack space as follows. It recurses on the smaller subarray as usual, but whenever it needs to recurse on the larger subarray, it uses an iterative module instead. What would be the depth of recursion for this variant? Compare it to the depth of recursion for usual implementation of quicksort. (4)

(f) Give an example graph with 5 nodes that gives two different Minimum Spanning Trees when computed with Prim's algorithm and Kruskal's algorithm. (4)

(g) Consider the following recurrence relation for computing the sum of n natural numbers.

$F(n) = F(n-1)+n. \quad n>1$

$F(1) = 1$

What is the running time of the recursive implementation of the above recurrence? Would memoizing the recursive solution improve the running time? Explain. (4)

(h) In a certain application it is required to find the nodes adjacent to a given node in a sparse graph. Which of the two representations, namely adjacency list and adjacency matrix, is more suitable? Justify. (4)

(i) Can Depth First Search algorithm be used to determine a shortest path from a source node to a destination node in an unweighted graph? Justify. (3)

P.T.O.

(i) Consider the Interval Scheduling problem wherein we are given a resource and a set of requests each having a start time and a finish time. The goal is to maximize the number of requests scheduled. Show that the following greedy strategy does not give an optimal solution for the above problem: select the request with fewest number of incompatible requests. (3)

2. (a) Let T be a red-black tree and x be a node in it. Let h be the height of the node x in T and d be its depth. Left rotate operation is performed on the node $x$ to give resultant tree $T'$.

(i) How does the height of $x$ change post rotation?

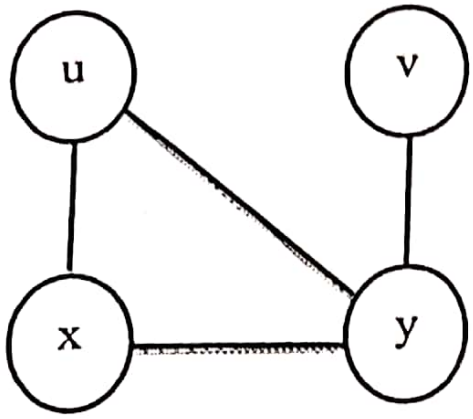
(ii) How does the depth of $x$ change post rotation? (4)

(b) Give a scenario in which the naive string matching algorithm demonstrates its worst case behaviour. (3)

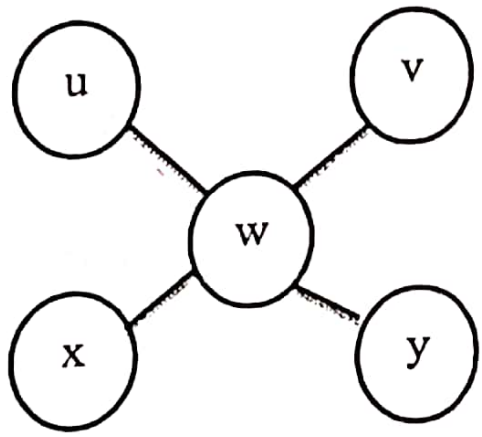
(c) Give a recurrence relation for solving the subset sum problem. (3)

3. (a) Consider the following graphs :

$G_1$




$G_2$




For each of the graphs, specify whether the graph is bipartite or not. If it is bipartite then give the two partitions else justify. (4)

(b) Recall the scheduling problem wherein we are given a single resource and a set of requests having deadlines. A request is said to be late if it misses the deadline. Our goal is to minimize the maximum lateness. With respect to a schedule S, we define idle time as the time during which the resource is idle, in between two requests. S is

said to have an inversion when request i has been scheduled before j, and d(i) > d(j) (d(k) being the deadline of a request k). Argue that all schedules with no idle time and no inversions have the same maximum lateness. (6)

4. (a) Consider the Heapsort Algorithm. Fill in the missing details correctly. (3)

*Assume A[1....length] be the array to be sorted*

```
MaxHeapify(A,i)
 l=2*i
 r=2*i+1
 if l <= A.heapsize and A[l]>A[r]
   largest = l
 else largest = i
 if r<=A.heapsize and A[r] >A[largest]
    largest = r
 if largest != i
    exchange A[i] and A[largest]
Maxheapify(A, _____ )


BuildMaxHeap(A)
A.heapsize = A.length
   for i = _____
MaxHeapify (A,i)

Heapsort(A)
BuildMaxHeap (A)
    for i = A.lengthdownto 2
        exchange A[ __ ] with A[i]
A.heapsize = A.heapsize -1
MaxHeapify ( A, __ )
```

(b) Is the following recurrence for the Knapsack problem correct? If not, give the correct recurrence. Justify your answer in either case.

If $w < w_i$ then $OPT(i,w) = OPT(i-1,w)$
Otherwise, $OPT(i,w) = \max(OPT(i-1),w), v_i + OPT(i-1, w)$ (4)

(c) Give an algorithm to determine if a given undirected graph is connected. (3)

(a) Consider a stack that supports the operations Push, Pop and Multi-push (pushes k items onto the stack). Suppose a sequence of n operations is performed on the stack. Would the amortized cost of an operation be O(1)? Explain. (4)

(b) If quick sort is run on an *n* sized array such that the array is always divided into 2 equal halves. How many times is the partition algorithm called? Explain briefly. (3)

(c) Consider the weighted interval scheduling problem. Will the following greedy strategy work? Justify.While requests remain, Choose and add a request to solution that has the largest starting time and delete all non-compatible requests. (3)

5. (a) In randomized-select algorithm randomized partition subroutine is used. If we replace the

randomized partition by a partition subroutine which chooses the last element of the list as pivot and call the modified algorithm as the *select* algorithm. What affect does it have on the running time of the select algorithm? (3)

(b) Show that any directed graph having a topological ordering must be acyclic. (4)

(c) For the elements 6, 2, 8, 3, 10; give a valid Red Black tree. (3)

7. (a) A d-ary heap is like a binary heap, except that nodes may have d children instead of two children. Consider a 3-ary heap represented using an array, how would the indices of the three children of a node be computed? (3)

(b) A thief wants to steal all the gold dust from a store having W kg of it. The thief has n sacks having different capacities. Give an efficient algorithm for the thief to fill his sacks with dust so that the number of sacks used is minimized. (3)

(c) Discuss the running time of the following function :

```
func(A)
    for i = 0 to A.length -2
      for  j = i+1 to  A.length – 1
        if (A[j-1] > A[j] )
          exchange (A[j-1] and A[j] )
```

(4)