7. (a) The BFS algorithm has been used to produce the shortest paths from a node s to all other nodes in a graph G. Can the Dijkstra's algorithm be used in place of BFS? In a different scenario, the Dijkstra's algorithm has been used to produce the shortest paths from a node s to all other nodes in a graph G'. Can BFS be used in place of the Dijkstra's algorithm? Explain your answers for both the scenarios. (6)

(b) Write a pseudocode for the memorized recursive algorithm to compute the nth Fibonacci number. What would be its time complexity? (4)

[This question paper contains 8 printed pages.]

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

Sr. No. of Question Paper : 4523 **E**

Unique Paper Code : 32341401

Name of the Paper : Design and 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) Use the Master's Theorem to give tight asymptotic bounds for the recurrence $T(n) = 8T(n/2) + \theta(n^2)$. (3)

(b) Discuss the running time of the following snippet of code :

```
count = 0
for (i=1, i<=n, i++)
      for (j=1, j<=n, j = 2 * j)
            count++
```
(3)

(c) A team of explorers is visiting the Sahara desert. As the weather is very hot, they are having n bottles of different sizes to carry water and keep them hydrated. In covering few kilometres, they used all of their water but fortunately found a water source nearby. This water source is having only L litres of water which is way lesser then the capacity of all bottles. They want to fill L litres of water in minimum number of bottles. Describe a greedy algorithm to help them fill U litres of water in minimum number of bottles. (3)

(d) Will the greedy strategy with the greedy parameter being value per unit weight of the items yield an optimal solution for the 0-1 knapsack problem? Justify. (3)

(ii) The maximum element found in step 1 is placed at the beginning of the not-yet-sorted portion of the array.

This algorithm is given as input a list already sorted in decreasing order. What would be the time complexity of the algorithm on this input? Explain. (4)

6\. (a) (i) What is the smallest possible depth of a leaf in a decision tree for a comparison sort? Name a sorting technique to which this smallest depth would correspond. (6)

(ii) What is the minimum number of leaves in the decision tree for a comparison sort? Use this observation to derive a lower bound on the number of comparisons performed by a comparison sort in the worst case.

(b) Show that at most 3* floor (n/2) comparisons are sufficient to find both the minimum and maximum in a given array of size n. (4)

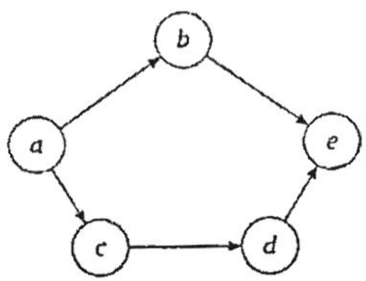
subarray in the array –1, 2, 3, –2, 5, –6, 7, –8 is 9 (which is the sum of the subarray 2, 3, –2, 5, –6, 7). Complete the following Dynamic Programming solution for the above problem :

DP[0] = A[0]

For i = 1 to n

DP[i] = max(A[i], ______) (4)

5. (a) How many topological orderings does the following graph have? Specify all of them. (6)



(b) A student was asked to sort a list of n numbers in decreasing order. The student writes an algorithm which works iteratively as follows. In every iteration, the following two steps are done :

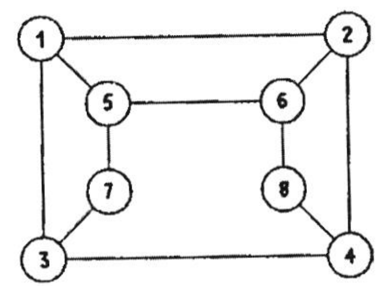
(i) Linear search is used to find the maximum element in the portion of the array which is not yet sorted.

(e) Can dynamic programming be applied to all optimization problems? Why or why not? (3)

(f) Let G be a tree-graph. Further, let T_B and T_D be the trees produced by performing BPS and DFS respectively on G. Can T_B and T_D be different trees? Why or why not? (4)

(g) Why is the worst-case running time for bucket sort $\theta(n^2)$? What changes would you make to the algorithm so that its worst-case running time becomes $O(n \lg n)$? (4)

(h) Consider the following graph: (4)



Specify whether the above graph is bipartite or not. If yes, give the partition, else justify.

(i) We are given a weighted graph G in which edge weights are not necessarily distinct. Can graph G have more than one minimum spanning tree (MST)? If yes, give an example, else justify. (4)

(j) Show that in any subtree of a max-heap, root of the subtree contains the largest value occurring anywhere in that subtree. (4)

2. (a) Consider the scheduling problem wherein you are given a single resource and a set of requests having deadlines. A request is said to be late be late if it misses the deadline. Your goal is to minimize the maximum lateness. With respect to a schedule S, idle time is defined 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 $d(i) > d(j)$, where d(i) and d(j) are the deadlines of the requests i and j respectively. Argue that all schedules with no idle time and no inversions have the same maximum lateness. (6)

(b) For each of the following sorting algorithms, merge sort and insertion sort, discuss whether or not it is

(i) stable

(ii) in-place (4)

3. (a) Let G = (V,E) be a directed unweighted graph. Given two vertices s and t in V, what is the time required to determine if there exists at least one s-t path in G? Can we use the DFS algorithm to find the shortest-path distance from the s to t? If yes, justify, otherwise give a counter example. (6)

(b) Suppose we perform a sequence of stack operations on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes. Show that the cost of n stack operations, including copying the stack, is O(n) by assigning suitable amortized costs to the various stack operations. (4)

4. (a) Show that for an n-element max heap (having distinct elements) represented through an array, the leaves are the nodes indexed by floor $(n/2 + 1)$, floor $(n/2 + 2)$,........, n. What would be the location of the minimum clement in the above heap? (6)

(b) Given an array A of n integers, you need to find the maximum sum of any contiguous subarray. For instance, the maximum sum of any contiguous