National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Design & Analysis of Algorithms**	**Course Code:**	**CS-2009**
	Program:	**BS (Computer/Data Science)**	**Semester:**	**Fall 2022**
	Duration:	**180 Minutes**	**Total Marks:**	**50**
	Paper Date:	**24-Dec-22**	**Section:**	**ALL**
	Exam:	**Final Exam**	**Page(s):**	**10**
	Name		**Roll Number**	

Instruction/Notes: Ample space is provided for rough work; NO EXTRA sheets will be provided.

Question	1	2-4	5-7	8-11	12	13	14	Total
Marks	/5	/5	/8	/10	/7	/7	/8	/50

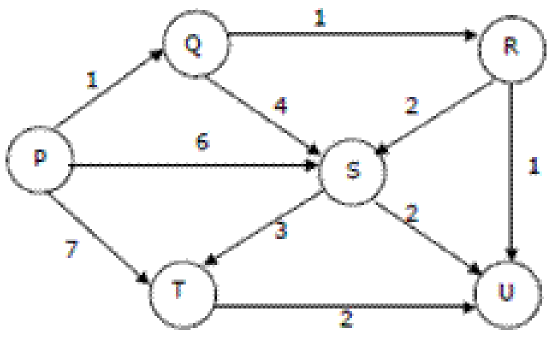
Q1) Solve the following recurrence and find out asymptotic time complexity. Show complete working. [5 Marks]

$T(n) = 5T(n/3) + \Theta(1)$

Q2) What is the worst case time complexity of the Quick sort? [1 Mark]

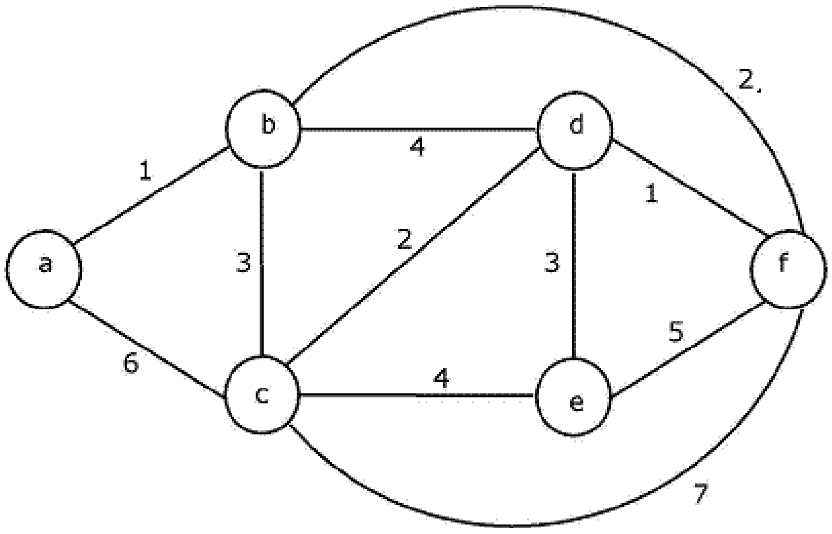
a) O(nlogn)
b) O(n)
c) O(n^3)
d) O(n^2)

Q3) Suppose we run Dijkstra's single source shortest-path algorithm on the following edge weighted directed graph with vertex P as the source. In what order do the nodes get included into the set of vertices for which the shortest path distances are finalized? [2 Marks]



a) P Q T S R U
b) P Q T S U R
c) P Q R S U T
d) P Q R U S T
e) P S Q R U T

Q4) Consider the following graph: [2 Marks]



Which one of the following cannot be the sequence of edges added, in that order, to a minimum spanning tree using Kruskal's algorithm?

I. (a—b),(d—f),(b—f),(d—c),(d—e)
II. (a—b),(d—f),(d—c),(b—f),(d—e)
III. (d—f),(a—b),(d—c),(b—f),(d—e)
IV. (d—f),(a—b),(b—f),(d—e),(d—c)

Q5) When does the worst case of Quick sort occur? [1 Mark]

Q6) Show how to sort n integers in the range 0 to $n^3 - 1$ in O(n) time. You can name the algorithm and explain why it will be linear. You do not need to write pseudocode of the algorithm but provide a justification why and how it will be linear. [4 Marks]

Q7) Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range [a . . b] in O(1) time. Your algorithm should use $\theta(n + k)$ preprocessing time. [3 Marks]

Q8) Dijkstra's algorithm does not guarantee shortest paths when the graph has negative edge weights. Prove this statement by giving an example. [2 Marks]

Q9) Adding a number w on the weight of every edge of a graph might change the shortest path between two vertices u and v. Is this statement True or False? Justify your answer with an example. [2 Marks]

Q10) let T be a minimum spanning tree of a graph G. Then for any two vertices u,v the path from u to v in T is a shortest path from u to v in G. Is this statement True or False? Justify your answer with an example. [2 Marks]

Q11) Given a weighted directed graph D, let T be its shortest path tree from a given vertex s. Let H = D + uv. H has one extra edge from vertex u to vertex v and the weight of this edge is w. Design an efficient algorithm to find shortest path tree of H from the same vertex s. Analyze its time complexity. [4 Marks]

Q12) Given a weighted undirected graph G, let T be its minimum spanning tree. Let H = G + uv. H has one extra edge between vertices u and v and the weight of this edge is w. Design an efficient algorithm to find minimum spanning tree of H. Analyze its time complexity. [10 Marks]

Q13) Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie. Each child i has a greed factor gi, which is the minimum size of a cookie that the child will be content with; and each cookie j has a size sj. If sj >= gi, we can assign the cookie j to the child i, and the child i will be content. Your goal is to maximize the number of your content children and output the maximum number. Analyze its time complexity. [10 Marks]

Note:
You may assume the greed factor is always positive.
You cannot assign more than one cookie to one child.

Example 1:
Input: [1,2,3], [1,1]
Output: 1
Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3. And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.
You need to output 1.
Example 2:
Input: [3,4], [1,2,4,5]
Output: 2
Explanation: You have 2 children and 4 cookies. The greed factors of 2 children are 3, 4.
You have 2 cookies whose sizes are big enough to gratify two children,
You need to output 2.

Q14) The longest increasing subsequence problem is to find a subsequence of a given sequence in which the subsequence's elements are in sorted order, lowest to highest, and in which the subsequence is as long as possible. This subsequence is not necessarily contiguous or unique. [10 Marks]

Please note that the problem specifically targets **<u>subsequences</u>** that need not be contiguous, i.e., subsequences are not required to occupy consecutive positions within the original sequences.
For example, the longest increasing subsequence of [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15] is [0, 2, 6, 9, 11, 15].
Following is a brute force recursive solution with exponential time complexity. For each item, there are two possibilities:

1. Include the current item in LIS if it is greater than the previous element in LIS and recur for the remaining items.
2. Exclude the current item from LIS and recur for the remaining items.

```
int LIS(int arr[], int i, int n, int prev)
{
    // Base case: nothing is remaining
    if (i == n) {
        return 0;
    }
    // case 1: exclude the current element and process the
    // remaining elements
    int excl = LIS(arr, i + 1, n, prev);

    // case 2: include the current element if it is greater
    // than the previous element in LIS
    int incl = 0;
    if (arr[i] > prev) {
        incl = 1 + LIS(arr, i + 1, n, arr[i]);
    }
    // return the maximum of the above two choices
    return max(incl, excl);
}
int main()
{
    int arr[] = { 0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15 };
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "The length of the LIS is " << LIS(arr, 0, n, INT_MIN);

    return 0;
}
```

Write an iterative dynamic programming solution (pseudocode) for this problem. Analyze its time complexity.

Extra space for Rough work