|  |  |
| --- | --- |
| **SESSION** | **NOV-DEC2023** |
| **PROGRAM** | **MASTER OF COMPUTER APPLICATIONS (MCA)** |
| **SEMESTER** | **III** |
| **COURSE CODE & NAME** | **DCA7104 - ANALYSIS AND DESIGN OF ALGORITHM** |
|  |  |
|  |  |

**Assignment Set – 1**

**1a. Compare and contrast the stack space utilisation in recursive and non-recursive algorithms by analysing a factorial function.**

**Ans 1a.**

**Comparison of Stack Space Utilization in Recursive and Non-Recursive Algorithms: A Case Study of Factorial Function**

When examining algorithms, particularly through the lens of the factorial function, a key aspect to consider is the stack space utilization. This comparison between recursive and non-recursive (iterative) approaches offers insights into their operational dynamics, efficiency, and suitability for different applications.

Recursive ApproachIts Half solved only

Buy Complete from our online store

<https://smuassignment.in/online-store/>

MUJ Fully solved assignment available for**session SEPT 2023.**

Lowest price guarantee with quality.

Charges**INR 198 only per assignment.**For more information you can get via mail or Whats app also

Mail id is [aapkieducation@gmail.com](mailto:aapkieducation@gmail.com)

Our website www.smuassignment.in

After mail, we will reply you instant or maximum

1 hour.

Otherwise you can also contact on our

whatsapp no 8791490301.

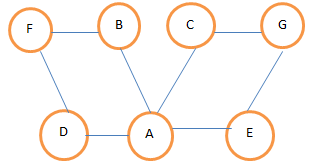
**2a. As the head of the IT department at the university library, you have been presented with a challenge. The library's current book search system has been identified as inefficient—especially during peak hours when student traffic is high, resulting in slow search response times. Considering that the library's database of book titles is sorted, propose a search algorithm to expedite search times.**

**Ans 2a.**

**Introduction**

In the context of the university library's need to improve its book search system, particularly during peak hours, it is essential to utilize a search algorithm that leverages the sorted nature of the library's database. The objective is to significantly reduce search response times, thereby enhancing user experience and operational efficiency.

**3a. Traverse the following graph by Breadth First search and constructthe corresponding BFS tree. Start the traversal at vertex ‘a’ and name the respective edges.**



**Ans 3a.**

To perform a Breadth-First Search (BFS) on the graph you've provided, we'll follow these steps:

1. Start at the initial vertex 'A' (as per the graph, we'll assume it is case insensitive and 'a' refers to 'A').
2. Visit all of

**Assignment Set – 2**

**1a. Apply bottom-up dynamic programming technique for the following instance of the knapsack problem with capacity M=5.**

|  |  |  |
| --- | --- | --- |
| **Item** | **Weight** | **Value** |
| **1** | **2** | **$ 12** |
| **2** | **1** | **$ 10** |
| **3** | **3** | **$ 20** |
| **4** | **2** | **$ 15** |

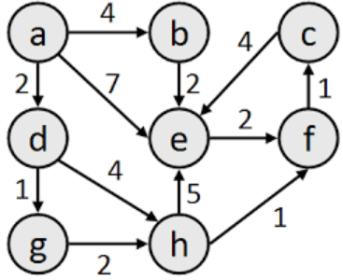
**Ans 1a.**

The analysis and design of algorithms, especially in the context of solving optimization problems like the knapsack problem, is a cornerstone in computer science. One efficient approach to tackle such problems is the use of dynamic programming techniques. In this instance, we'll apply a bottom-up dynamic programming approach to solve a specific instance of the knapsack problem.

**Understanding the Problem**

Top of Form

**2a. Solve the following Single Source Shortest path problem assuming a suitable vertex and obtain its time efficiency.**



**Ans 2a.**

To solve the Single Source Shortest Path (SSSP) problem, we need to select a suitable algorithm. The most common algorithms for SSSP are Dijkstra's Algorithm and Bellman-Ford Algorithm.

Dijkstra's algorithm is more efficient when all edge weights are non-negative, as it can provide a time

**3a. Explain the concept of backtracking and discuss its application in solving the N-Queens puzzle. Outline the steps and provide the pseudocode for the algorithm.**

**Ans 3a.**

Backtracking is a systematic method for solving problems algorithmically, particularly for constraint satisfaction problems. This technique incrementally builds candidates to the solutions and abandons each partial candidate ("backtracks") as soon as it determines that this candidate cannot possibly be completed to a valid solution.

The backtracking algorithm is a depth-first search with some additional features. The main feature