|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **National University of Computer and Emerging Sciences, Lahore Campus** | | | | |
| C:\Users\saif\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.Word\final design.jpg | **Course:** | **Design & Analysis of Algorithms** | **Course Code:** | **CS2009** |
| **Program:** | **BS (Computer Science)** | **Semester:** | **Spring 2023** |
| **Duration:** | **15 Minutes** | **Total Marks:** | **10** |
| **Paper Date:** | **30-March-2023** | **Weight:** | **4** |
| **Section:** | **G** | **Page(s):** | **1** |
| **Exam:** | **Quiz 3** | **Reg. No.** |  |
| **Instruction/Notes:** |  | | | |

**Question 1: CLO 1, [10 marks]**

Suppose your Design and Analysis of Algorithms professor has given you a really tough take home exam having total n challenging questions. Each question has different marks and different amount of time to solve the question. The time required to solve the question is also given with each question. It is not necessary that the question requiring more time has more marks. There is no relation between the time required to solve the question and its total marks. The professor has announced that she will not give partial credit to incorrect or incomplete solutions and there will be binary marking. The paper is so lengthy that you cannot attempt all questions in given time so you have to choose the questions wisely.

Devise an efficient bottom up dynamic programming algorithm to select which questions should be solved in order to maximize total marks. You are given n questions, with marks and time (minutes) for each question. Total time (T minutes) to solve the paper is also given.

For example, suppose the following table gives the 4 questions of the exam. The total time to solve the paper is 180 minutes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Problem 1 | Problem 2 | Problem 3 | Problem 4 |
| timei | 50 | 70 | 60 | 100 |
| marksi | 100 | 200 | 50 | 400 |

Optimal solution = 200 + 400 = 600 marks

Solution:

This problem can be mapped to binary knapsack problem as follows:

Items = Problems

Knapsack capacity = Total Time

Value of Item = marks of problem

Weight of Item = time required for problem

The time complexity is same as binary knapsack O(nT) where n is number of problems and T is total Time.