**Department of Computer Science and Engineering**

|  |  |
| --- | --- |
| **Course Code: CSE221** | **Credits: 1.5** |
| **Course Name: Algorithms** | **Semester: Fall’18** |

**Lab 07  
0-1 Knapsack**

1. **Topic Overview:**

The idea of 0-1 Knapsack is that a thief wishes to fill his knapsack with items that would maximize his profit. However, the items cannot be broken which means the thief should take the item as a whole or should leave it. This is the reason behind calling it 0-1 Knapsack. However, greedy approach i.e., selecting items with maximum values does not ensure optimal solution always. Therefore, we would use dynamic programming to solve this problem. Dynamic programming is an optimization technique that would help us to manipulate the combinational optimization criteria (i.e., optimal combination of items that ensures profit maximization) required for 0-1 Knapsack problem. Dynamic programming gives us an optimal way to consider all possible subsets of items that maximize the profit.

1. **Lesson Fit:**

To solve this problem, the students must have a basic idea on the following concepts:

* 1. Dynamic Programming
  2. Overlapping Sub-problem
  3. Optimal Substructure

1. **Learning Outcome:**

After this lecture, the students will be able to:

* 1. Learn how to use dynamic programming to solve problems that have overlapping sub-problem property. Dynamic Programming is mainly used when solutions of same sub-problems are needed again and again. In dynamic programming, computed solutions to sub-problems are stored in a table so that they don’t have to be recomputed. So, Dynamic Programming is not useful when there are no common (overlapping) sub-problems because there is no point in storing the solutions if they are not needed again. Since, 0-1 Knapsack problem satisfies overlapping sub-problem property, dynamic programming will be particularly useful here. The students can avoid re-computations of sub-problems by constructing a temporary array to store solutions of sub-problems.
  2. Learn how to manipulate optimal substructure property that is getting the solution of a particular problem by using the existing solutions of its sub-problems.

1. **Anticipated Challenges and Possible Solutions**
   1. The students often fail to grasp the notion of 0-1 Knapsack problem. Therefore, they often fail to apply this solution to other scenarios besides the *thief and knapsack* theme.

**Solutions:**

* + 1. The students need to think beyond the concept of thief and knapsack and have a clear idea about the notion of 0-1 Knapsack problem.

1. **Acceptance and Evaluation**

Students will show their progress as they complete each problem. They will be marked according to their class performance. There may be students who might not be able to finish all the tasks, they will submit them later and give a viva to get their performance mark. The mark distribution for the lab will be as follows:

Code: 05

Viva: 05

1. **Activity Detail**
   1. **Hour: 1  
      Explanation:**The teachers will explain the basic algorithm of 0-1 Knapsack problem. The teachers will discuss how to use a temporary array to store solutions of optimal sub-problems and then use it to get solutions of other sub-problems.
   2. **Hour: 2**

**Implementation:**

After explanation, the students will implement the 0-1 Knapsack algorithm for the thief

**Problem Task:**

* + 1. Task 1
  1. **Hour: 3**

**Evaluation:**

The teachers will check the status of Task 1 and explain Task 2.

1. **Home tasks**
   1. Unfinished tasks

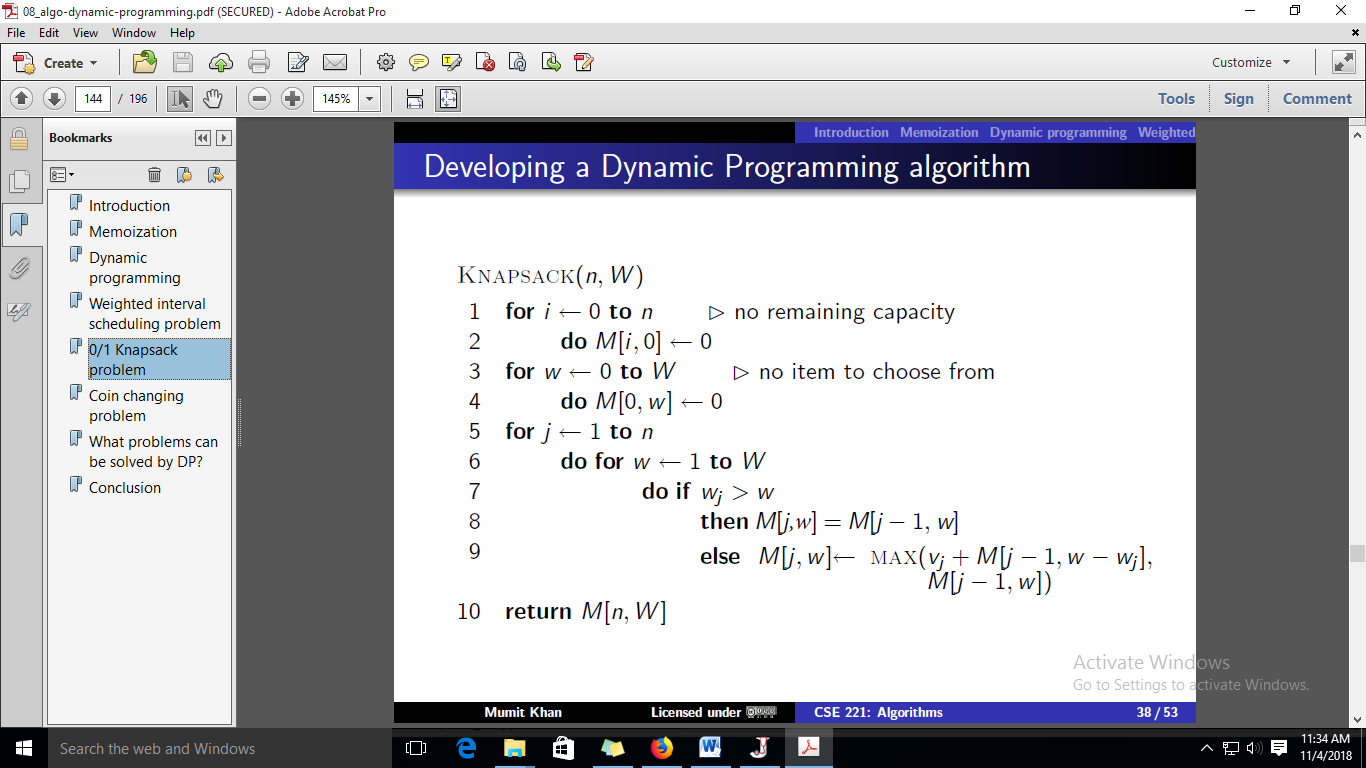
**Lab 7 Activity List**

**Task 1: A day for the Thief**

A thief breaks into a house with only one bag (knapsack). The knapsack can carry only 7 kg in total. The thief realizes that he cannot enter other rooms so he has limited. In the room he entered the following items were found. Laptop, Phone, Speaker, Necklace, Watch. The total weight of all items is greater than the knapsacks capacity. So the thief decided to call you to help him select the items that will provide the maximum total sale value. The weight and sale value of each item is:

|  |  |  |
| --- | --- | --- |
| **Items** | **Weight(KG)** | **Value(TK)** |
| Laptop | 3 | 50000 |
| Phone | 1 | 100000 |
| Speaker | 3 | 20000 |
| Necklace | 3 | 30000 |
| Watch | 2 | 60000 |

You will implement the 0-1 knapsack algorithm to find which items the thief should steal and what will be his total benefit (or sale value).



**Task 2: Fair Distribution of Money**

Rahim & Karim are two brothers. Their father left them a bag full of gold coins when he died. Now, the contents of the bag appeared not to be equally divisible. And the two brothers were fighting over the fair share of the coins because they couldn’t stand that one would benefit over the other. Don’t assume the problem is about dividing odd numbers of equal valued gold coins between two brothers.

**The problem is that the bag is full of gold coins of different values! That’s what makes the fare division difficult!**

That’s what this whole problem is about. Not everyone is capable of seeing instantly what the fairest division of a bag of coins between two persons is. Your help is asked to solve this problem. Given a bag with a maximum of 100 gold coins, determine the fairest division between two persons. This means that the difference between the amounts each person obtains should be minimized. The value of a coin varies from 1 tk to 10 tk. It’s not allowed to split a single coin.

**Input:**A line with the number of problems n, followed by n times: • a line with a non-negative integer m (m ≤ 100) indicating the number of coins in the bag • a line with m numbers separated by one space, each number indicates the value of a coin

**Output:**The output consists of n lines. Each line contains the fair distribution of coins between two persons and minimal positive difference between the amount they obtained.

**Sample Input:**2   
3   
2 3 5   
4   
1 2 4 6

**Sample Output:**Karim: 2 3, Rahim: 5, difference: 0  
Karim: 1 6, Rahim: 2 4, difference: 1 (Or, Karim: 6, Rahim: 1 2 4, difference: 1)