CSE2046 Assignment 2

0-1 Multi-Constraint Knapsack Problem

The knapsack problem is a maximization problem and in which Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible but this assignment about multi-constraint problem which has multiple knapsacks with different capacities and changing weights according to knapsacks.

We solve Knapsack 0/1 problem using Greedy Method (recursively)

Greedy Method is one of the strategy or approach for solving optimization problem which require either minimum or maximum result.

1.First read the file then create structs according to values and weights. Each struct repesents an item and it has integer values such as id, value and array for weights (weight x number of knapsack) and a double value representing their profit.

2. We created an array of these structs, we calculated their profit with following formula:

$$\sqrt[n]{\left(\frac{value}{weight1} * \frac{weight1}{capacity1}\right) * \left(\frac{value}{weight2} * \frac{weight2}{capacity2}\right) * \dots\dots * \left(\frac{value}{weightn} * \frac{weightn}{capacityn}\right)}$$

3.We sorted the array of structs according to their profit values in descending order.

4.We placed it in all knapsack, starting with the most valuable item and then we started filling each bag until its capacity was full.

During this process, we marked the items that is placed in the bags (using flag). If an item fits in a bag, we make its flag 1 for the bag it fits. After all bag were filled, we revisited all items again and checked their flags, If all flags of an item are 1, it means that item can fit in all bags, otherwise item cannot fit into all of bags.

5.We summed the flags of the items. If the sum of the flags is equal to the number of bags, we summed up its value since it indicates that the item should be selected. So first iteration was done.

6.Then we repeated the above step by decreasing number knapsack by 1. Thus, we continued to place items recursively until the capacity of the bags was full.

Here our results:

Test File	**Test1**	**Test2**	**Test3**	**Test4**
Result	1087949	10656	8563	679