CSE2046 Assignment 2

0-1 Multi-Constraint Knapsack Problem

Due date: June 13th, 2021 (for the test outputs), June 15th, 2021 (for the report and codes)

In this assignment, you are asked to design and implement an algorithm for 0-1 Multi-constraint (or Multi-dimensional) Knapsack Problem that is described below.

0-1 Multi-constraint Knapsack problem (MCKP) is an extension of Knapsack problem, where we have multiple constraints (for example we may have a volume limit and weight limit, where the volume and the weight of each item is not related). If there are *m* constraints, we may consider that there are *m* knapsacks, each with capacity of W_j. There are *n* items and weight of item *i* in knapsack *j* is defined as $w_{i,j}$. When we select an item, we may imagine that we insert a copy of it to all the knapsacks simultaneously (although we have a single instance from each item). (Note again that items weights are typically not same for different knapsacks).

Formally:

Maximize $\sum_{i=1}^{n} v_i x_i$ // The objective is to maximize total value of selected items.

Subject to $\sum_{i=1}^{n} w_{i,j} x_i \leq W_j$ for all $1 \leq j \leq m$ // Capacity constraints

$x_i = \{0,1\}$ for all $1 \leq i \leq n$ // x_i is equal to 1 if item *i* is selected, and zero otherwise.

MCKP can be used to model many practical problems.

Project Specification:

Your team (up to 3 students) is asked to design and implement an algorithm for MCKP described above.

Since MCKP is an NP-hard problem, it is difficult to find optimal solutions especially for large instances. Your goal is not to design an algorithm for the optimal solution, but you are requested to do your best.

You may do the following:

- Read as much as you want to learn about how to solve the multi-constraint Knapsack problem. But **you have to cite** any resources you use. You may want to start with some approximation algorithms (such as local search heuristics) in chapter 12.
- You may use whichever programming language you want.

You may **not use** the following:

- Existing implementations or subroutines
- Extensive libraries (if you are not sure, check with the instructor)
- Other people's code.

Input format: Inputs will always be given as a text file. Input file format, using 10 columns, whenever possible, should be as follows:

```
m n                                  //<m := #knapsacks> <n := #items>
v1 v2 ... vn                         //<n values of items>
W1 W2 ... Wm                         //<m knapsack capacities>
w1,1 w2,1 ... wn,1                   //<nxm matrix of constraints>
w1,2 w2,2 ... wn,2
.................
w1,m w2,m ... wn,m
```

In a single row, there should be maximum of 10 values. For more than 10 values, we should continue with the next row.

No other input format will be accepted!

Output format for Problem 1: Output should be a text file including following:

Total_Value
x_1
x_2
:
x_n

First line should include Total_Value which is the sum of the values of the included items ($\sum_{i=1}^{n} v_i x_i$). Next lines include zeros and ones. **No other output format will be accepted!**

Example instances: You may find example instances in the google classroom. There are three example inputs (sample1.dat, sample2.dat, and sample3.dat). The optimal values are given in the following table:

Input File	Optimal Solution
Sample1.txt (m=2, n=28)	141278
Sample2.txt (m=5, n=40)	5605
Sample3.txt (m=30, n=60)	7772

A sample output file is also provided for sample1.txt, however it is not the optimal solution. Output files for optimal solutions are not available.

Test instances: On June 12th, we will announce 4 test instances. By June 13th, 23:59, you will be required to submit 4 separate output text files (with the correct output format) corresponding to each of these test instances. These files should be called output1.txt,

output2.txt, output3.txt, output4.txt and should be submitted via google classroom. The deadline is strict.

Project report: **The project report should describe the ideas behind your algorithms as completely as possible**. It should not exceed 3 pages in length in no less than 10pt. Also please include a README file to briefly describe how to run your code. You should submit a zip file (with name **studentID1_studentID2_studentID3.zip**) including your project report, README file and commented source codes by June 15th, 2021, 23:59.

Grading policy:

60% of your grade will be determined by your project report. Clarity and creativity of your work will significantly affect your grade.

40% of your grade will be determined by your solutions to the test instances. Studies that find solutions closer to the best possible solution will get higher grades.

Best projects will be awarded with bonus points.