COMP319 Algorithms 1, Fall 2021
Homework Programming Assignment 4 (HW4)
4 problems, total 100pts

Instructor: Gil-Jin Jang   Email: `gjang@knu.ac.kr`
School of Electronics Engineering, Kyungpook National University

Common Requirements (same as HW2 and HW3)

ID and name write your ID and name in a comment block at the beginning of the code.

Citation knapsack algorithm implementations are already given in the lecture slide. Moreover, there are huge number of knapsack algorithm implementations that are available on the Internet. Therefore, if you make use of them, there is a high chance that your code could be almost identical to that of other students. Even if you do not refer to any other source, your code may eventually become very similar to each other because you have to implement the same algorithms. To avoid any unwanted COPY, if you have referred to any material, write the web address, name of the textbook, lecture number and page number, etc., as comments near the relevant code lines. There is no reduction in scores by using publicly available codes.

Character code use standard ASCII characters only. Special characters or language-specific symbols can take more than 1 byte, and may cause compilation errors. There is no explicit penalty for using special characters, but students should be responsible for the compilation errors due to those.

---

1 Homework 4-1: 0-1 knapsack problem

Knapsack problem is finding a best subset of given items that satisfies:

- the sum of the item weights is less than the capacity of the knapsack (W).
- the sum of the item values is maximized. If we allow 0-1 choices only, the items are indivisible, and the relaxed problem can be solved with dynamic programming.

To do: design an algorithm that finds:

1. the subset of the items by their numbers (chosen from 1 to N).
2. the maximum value.

INPUT: The input is given by a text file, with each row represents each item's weight and value. The last line is the maximum weight followed by -1, so you can finish reading the file if the input number is -1.

```
2 3
3 4
4 6
```

```
5 7
8 -1
```

All the values are POSITIVE integers. Use file I/O with the input file name as `argv[1]`. From the input file, number of items $n = 4$, (weight, benefit) of the items are: $(2, 3)$, $(3, 4)$, $(4, 6)$, $(5, 7)$, and $W = 8$ (max weight).

OUTPUT: `2 4 11`

---

2 Homework 4-2: 0-1 knapsack with one item split

In this problem, it is assumed that ONLY ONE item can be split by HALVES — one item is split into two items with the same weight and the same value, half the weight and half the value, respectively. We may choose one half-split item, or whole item, whichever maximizes the value.

To do: design an algorithm that finds:

1. the subset of the items by their numbers (chosen from 1 to N).
2. the maximum value.
3. if necessary, specify one item to be split by halves by appending $\times 0.5$ to the item number.

Note: when the weights and values are odd numbers, their split weights and values may have fractional parts (0.5). If you use a dynamic programming algorithm (probably), those odd number cases should be considered.

INPUT: same as homework 4-1.
No other input files will be given. Find good examples yourself.

OUTPUT: `1x0.5 2 3 11.5`

The standard 0-1 knapsack solution is choosing items 2 and 4, and their value is 11, which is less than 11.5.

---

3 Homework 4-3: 0-1 knapsack with one duplicate item

Same as problem 2, except that *EXACTLY ONE ITEM* can be added *TWICE*.

To do: design an algorithm that finds:

1. the subset of the items by their numbers (chosen from 1 to N).
2. the maximum value.
3. if necessary, specify one item to be split by halves by appending $\times 2$ to the item number.

INPUT: same as homework 4-1.

OUTPUT: `3x2 12`

---

4 Homework 4-4: 0-1 knapsack with two identical knapsacks

Same as problem 4-1, except there are *TWO KNAPSACKS* to be filled.

To do: design an algorithm that finds:

1. the subset of the items by their numbers (1 to N) and **knapsack numbers** (1 or 2).
2. the maximum value.

INPUT: The input is given by a text file, with each row represents each item's weight and value. The last line is the maximum weights of the **TWO KNAPSACKS** followed by -1, so you can finish reading the file if the input number is -1.

```
2 3
3 4
4 6
5 7
8 7 -1
```

All the values are POSITIVE integers. Use file I/O with the input file name as `argv[1]`. From the input file, number of items $n = 4$, (weight, benefit) of the items are: $(2,3)$, $(3,4)$, $(4,6)$, $(5,7)$, and $W = (8,7)$ (max weights).

OUTPUT: `2 1 4 1 1 2 3 2 20`

Note: may not be solvable by dynamic programming (even the instructor does not know). Consider greedy algorithms if necessary.

Evaluation Scheme

The total score of homework 4 is 100, and 15% of the whole course evaluation.

10pts basic submission score

10pts ID/name exist and are correct

50pts no compilation error and correct execution result. Will be evaluated by the unknown examples. Score deduction if not implemented properly, for example, using wrong algorithm.

30pts code reading scores (requirements, etc.)

X 0 COPY makes whole scores 0. COPIED/BEING COPIED same. Leave citations as much as possible.

Submission format

Files to submit Source files only. Up to 10pts deduction when unnecessary files are included (input files, execution files, project files, etc.)

Submission
1. make a zip file `hw4.zip` of `hw4-1.c` $\simeq$ `hw4-4.c`.
2. upload it to `lms.knu.ac.kr`. The LMS system changes the submitted files, and this zip file submissoin is to preserve the source file name.

Due Friday 11/19 23:59 LMS time

Late submission Saturday 11/20 09:59 LMS time, -10pts per hour