Discrete Optimization Assignment:
Knapsack

1 Problem Statement

In this assignment you will design an algorithm to solve the infamous *Knapsack Problem*, which plagues Indiana Jones. You are provided with a knapsack with limited space and a collection of items with different values and weights. Your task is to maximize the value of items packed into your knapsack without exceeding its total capacity.

2 Assignment

Write an algorithm to solve the knapsack problem. The problem is mathematically formulated in the following way. Given n items to choose from, each item $i \in 0 \dots n-1$ has a value v_i and a weight w_i. The knapsack has a limited capacity K. Let x_i be a variable that is 1 if you choose to take item i and 0 if you leave item i behind. Then the knapsack problem is formalized as the following optimization problem,

$$\begin{aligned} \text{maximize:} \quad & \sum_{i \in 0 \dots n-1} v_i x_i \\ \text{subject to:} \quad & \\ & \sum_{i \in 0 \dots n-1} w_i x_i \leq K \\ & x_i \in \{0, 1\} \quad (i \in 0 \dots n-1) \end{aligned}$$

3 Data Format Specification

A knapsack input contains $n+1$ lines. The first line contains two integers, the first is the number of items in the problem, n. The second number is the capacity of the knapsack, K. The remaining lines present the data for each of the items. Each line, $i \in 0 \dots n-1$ contains two integers, the item's value v_i followed by its weight w_i.

Input Format

```
n K
v_0 w_0
v_1 w_1
...
v_n-1 w_n-1
```

The output contains a knapsack solution and is made of two lines. The first line contains two values *obj* and *opt*. *obj* is the total value of the items selected to go into the knapsack (i.e. the objective value). *opt* should be 1 if your algorithm proved optimality and 0 otherwise. The next line is a list of n 0/1-values, one for each of the x_i variables. This line encodes the solution.

Output Format

```
obj opt
x_0 x_1 x_2 ... x_n-1
```

It is essential that the value order in the solution output matches the value order of the input. Otherwise the grader will misinterpret the output.

Examples

Input Example

```
4 11
8 4
10 5
15 8
4 3
```

Output Example

```
19 0
0 0 1 1
```

4 Instructions

Edit `solver.py` and modify the `solve_it(input_data)` function to solve the optimization problem described above. The function argument, `input_data`, contains the problem data in the format described above. The return value of `solve_it` is a solution to the problem in the output format described above. Your `solve_it` implementation can be tested with the command,

```
python ./solver.py ./data/<inputFileName>
```

You should limit the `solve_it` method to terminate within 5 hours, otherwise the submission will not be eligible for full credit. You may choose to implement your solver directly in python or modify the `solve_it` function to call an external application.

Resources You will find several knapsack instances in the `data` directory provided with the handout. `Solver.java` and `solverJava.py` are included in this assignment to demonstrate how to implement your assignment in another language. If you wish to use the Java solver, you will need to rename `solverJava.py` to `solver.py`.

Handin Run `submit.py` with the command, `python ./submit.py`. Follow the instructions to apply your `solve_it` method on the various assignment parts. You can submit multiple times and your grade will be the best of all submissions. However, it may take several minutes before your assignment is graded; please be patient. You can track the status of your submission on the *feedback* section of the assignment website.

Grading Infeasible solutions (i.e. those that do not conform to the output format or violate problem constraints) will receive 0 points. Feasible solutions will receive at least 3 points. Feasible solutions passing a low quality bar will receive at least 7 points and solutions meeting a high quality bar will receive all 10 points. The grading feedback indicates how much your solution must improve to receive a higher grade.

Collaboration Rules In all assignments we encourage collaboration and the exchange of ideas on the discussion forums. However, please refrain from the following:

1. Posting code or pseudo-code related to the assignments.
2. Using code which is not your own.
3. Posting or sharing problem solutions.

Discussion of solution quality (i.e. objective value) and algorithm performance (i.e. run time) is allowed and the assignment leader board is designed to encourage such discussions.

Warnings

1. It is recommended you do not modify the `data` directory. Modifying the files in the data directory risks making your assignment submissions incorrect.
2. You cannot rename the `solver.py` file or the `solve_it()` method.
3. Be careful when using global variables in your implementation. The `solve_it()` method will be run repeatedly and it is your job to clear the global data between runs.
4. `solver.py` must remain in the same directory as `submit.py`.

5 Technical Requirements

You will need to have python 2.7.x or 3.5.x installed on your system (installation instructions, `http://www.python.org/downloads/`).