Introduction to Algorithms: 50.004
Singapore University of Technology and Design

Problem set 4

Due date and submission directions: Please check class website.
You have to do this problem set in a group of 2 to 3 students (3 recommended).
Please include name and student id of all members in the email.

(50 points) Dynamic programming – knapsack problem

You have a bag of capacity 6. Now there are 4 pieces of gold in front of you:

Piece number k	G_1	G_2	G_3	G_4
Value v(k)	2	3	4	7
Weight w(k)	1	2	3	5

To collect gold with the maximum value, we use the following dynamic programming approach:

$B_k(N)$ = *optimal value collected with bag of capacity N, using gold* G_1 *to* G_k
$= max \{ v(k) + B_{k-1}(N-w(k)) , B_{k-1}(N) \}$
$B_k(0) = 0$

For example, $B_2(3) = max \{ v(2) + B_{2-1}(3-w(2)) , B_{2-1}(3) \}$
$= max \{ 3 + B_1(3-2) , B_1(3) \}$
$= max \{ 5, 2 \} = 5$

Here it is a table of B(N):

N = Bag capacity	*1*	*2*	*3*	*4*	*5*	*6*
G_1	*2* (G_1)	*2*	*2*	*2*	*2*	*2*
G_1,G_2	*2*	*3* (G_2)	*5* (G_1+G_2)	*5*	*5*	*5*
G_1,G_2,G_3	*2*	*3*	*5*	*6* (G_1+G_3)	*6*	*9* ($G_1+G_2+G_3$)
G_1,G_2,G_3,G_4	*2*	*3*	*5*	*6*	*7* (G_4)	*9*

Todo:

Write knapsack.py that solve the knapsack problem. We provided input1.txt and input2.txt for you to test your program. We provided knapsack.py as a skeleton.

You can either use our knapsack.py or write your own code. But your code must print the output in the following format.

Knapsack.py input2.txt should give:

```
Best possible total value: 9
Items:
Value: 2, Weight: 1
Value: 3, Weight: 2
Value: 4, Weight: 3
```

Knapsack.py input2.txt should give:

```
Best possible total value: 309
Items:
Value: 92, Weight: 23
Value: 57, Weight: 31
Value: 49, Weight: 29
Value: 68, Weight: 44
Value: 43, Weight: 38
```

- The items need to be printed in descending order of value/weight.
- Note: you will find that the Greedy Algorithm is poorer compared with the dynamic programming approach