

## Coping With Hardness

As you have already covered in class, there are many problems that we do not think to be solvable in polynomial time. Is it then the end of the world? Not at all. It turns out that many real world problems that we care about are NP-hard.

This week, we will solve the 0-1 Knapsack problem once again. Here there are  $n$  objects each with a weight  $w_i$  and a value  $v_i$ . We want to maximize the values of the items we take, but we are also bounded by a capacity  $W$ . Formally, we want to find the subset  $S$  of the universe of objects such that  $\sum_{i \in S} v_i$  is maximized subject to the constraint that  $\sum_{i \in S} w_i \leq W$ .

In this assignment, you will first implement Algorithm-II in the attached PDF. Augment your algorithm so that you can output the index of the values you take in your final solution.

Consider the input file to contain  $n$  and  $W$  in two separate lines. Then each of the following  $n$  lines will contain  $v_i$  and  $w_i$ .

Example Input:

```
8
65
30 15
45 15
140 17
15 50
154 50
7 25
77 23
15 50
```

You will print the total value of the items as well as the indices of the items in your final solution.

Example Output (only for the original instance):

```
Answer: 262
Used Weight: 55
Indices: 7 3 2
```

Note that  $v[2] + v[3] + v[7] = 45 + 140 + 77 = 262$ .

Knapsack is such a problem that admits an FPTAS. In easy terms, it can be arbitrarily approximated within a threshold  $\epsilon$ . The attached PDF contains the details. For each  $\epsilon \in [0.5, 0.2, 0.1, 0.05]$ , you will run your FPTAS algorithm for knapsack. For each  $\epsilon$ , print the following info.

Example Output (For attached in.txt):

```
Original Instance:
Answer: 12701
Used Weight: 659
Indices: 41 37 34 29 24 17 13 12 1
```

Rounded Instance with Eps: 0.5  
Theta: 8.404761904761905  
Answer of reduced instance: 1514  
Answer of reduced instance multiplied by theta: 12724.809523809525  
Indices: 41 39 27 24 19 12 11 10 4  
Answer of original instance (rounded up): 12680  
Used Weight: 649  
Ratio: 1.0016561514195583

Rounded Instance with Eps: 0.2  
Theta: 3.3619047619047624  
Answer of reduced instance: 3782  
Answer of reduced instance multiplied by theta: 12714.723809523812  
Indices: 41 37 34 30 28 24 21 12 10  
Answer of original instance (rounded up): 12694  
Used Weight: 654  
Ratio: 1.000551441625965

Rounded Instance with Eps: 0.1  
Theta: 1.6809523809523812  
Answer of reduced instance: 7562  
Answer of reduced instance multiplied by theta: 12711.361904761907  
Indices: 41 37 34 29 24 17 13 12 1  
Answer of original instance (rounded up): 12701  
Used Weight: 659  
Ratio: 1.0

Rounded Instance with Eps: 0.05  
Theta: 0.8404761904761906  
Answer of reduced instance: 15115  
Answer of reduced instance multiplied by theta: 12703.79761904762  
Indices: 41 37 34 29 24 17 13 12 1  
Answer of original instance (rounded up): 12701  
Used Weight: 659  
Ratio: 1.0

Try to generate test cases such that the ratio becomes closer to  $(1 + \epsilon)$ .

### Submissions Guidelines

Submit a your source files (.c/.cpp/.java/...) as 2105xyz.zip. In addition, attach some (at least three) input and output files you experimented with. Do not copy from anyone or anywhere. Usual guidelines will apply.

### Marks Distribution

| Item                                 | Marks |
|--------------------------------------|-------|
| Solving for Original Instance        | 40%   |
| Solving for Rounded Instance         | 50%   |
| Experimenting for Several Test Cases | 10%   |

If you can generate test cases where ratio is larger than  $(1 + 0.7\epsilon)$ , you can claim a 10% bonus.

### **Submission Deadline**

Monday, March 04, 2024, 11:55 PM.