Fletcher Baker
3/18/2019

2D 0/1 Knapsack

We solved the 0/1 Knapsack problem earlier in class, where each item had a weight, or cost, associated with it, as well as a benefit. This reminded me of old-school video games (such as Diablo) where you had to manage your inventory. Each item has a value associated with it (such as value to sell) and not a weight, but a size and shape, while your inventory (the knapsack) was a two-dimensional grid. So, for my problem, I am modifying the 0/1 knapsack problem from earlier in class to handle this relationship of space vs benefit, instead of weight vs benefit.

Our adventurer left home to be a hero instead of going to school and never learned how to solve a maximization problem. They need your help determining the best combination of treasures to bring back after defeating the dragon!

Input

The input will consist first of a number k ($100 \geq k > 0$), specifying the number of data sets to process. Following this, for each data set, will be the two positive, non-zero integers x and y (separated by a space) that represent the dimensions of his backpack (both max of 1000), followed by an integer n ($100 \geq n > 0$) specifying the number of types of treasure available (the dragon's hoard is large enough that you do not have to account for how many treasures of any type are available). Following this number n will be a character t denoting the treasure type ($Z \geq t \geq A$; upper or lowercase), followed by a space, then the treasure's value v, another space, the treasure's width w ($100 \geq w > 0$), another space, and then the treasure's height h ($100 \geq h > 0$). All numbers are integers. The input is guaranteed to exist and to be well formatted.

Output

For each data set, output the size of the backpack (width by height), followed by the value in the backpack. See Sample Output for more details.

Sample Input

```
2
5 5
3
R 5 2 2
S 9 2 3
G 1 1 1
5 4
2
W 2 1 3
A 5 2 4
```

Sample Output

```
Backpack size: 5x5, Value: 33
Backpack size: 5x4, Value: 12
```