

Instituto Politécnico Nacional

Escuela Superior de Cómputo

EVOLUTIONARY COMPUTING

Session 1

”Python and Greedy Algorithms”

Student: Naranjo Ferrara Guillermo

Teacher: Rosas Trigueros Jorge Luis

Lab session date: 04 February 2020

Delivery date: 11 February 2020

1 Theoretical Framework

1.1 Python

Python is an interpreted language that uses dynamic typing, which philosophy emphasizes the legibility of the code. It's a multi-paradigm language because it supports POO, imperative programming and functional programming.[1]

1.2 Greedy Algorithms

Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. So the problems where choosing locally optimal also leads to global solution are best fit for Greedy.[2]

1.2.1 Popular problems solved with Greedy

Let's check two popular problems that a Greedy algorithm can solve, the knapsack problem and the CMP problem.

Knapsack problem

Given weights and values of n items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack.[3]

There are some variants of this problem like the 0-1 where we choose to put or not only one of each item into the knapsack. Another variant is where we can choose more than once the same item. And other one is the fractional problem, which I'm going to explain because it is the one that can be solved correctly with a Greedy algorithm.

In Fractional Knapsack, we can break items for maximizing the total value of knapsack.[3]

The greedy approach is to calculate the ratio value/weight for each item and sort the item on basis of this ratio. Then take the item with the highest ratio and add them until we can't add the next item as a whole and at the end add the next item as much as we can. Which will always be the optimal solution to this problem.[3]

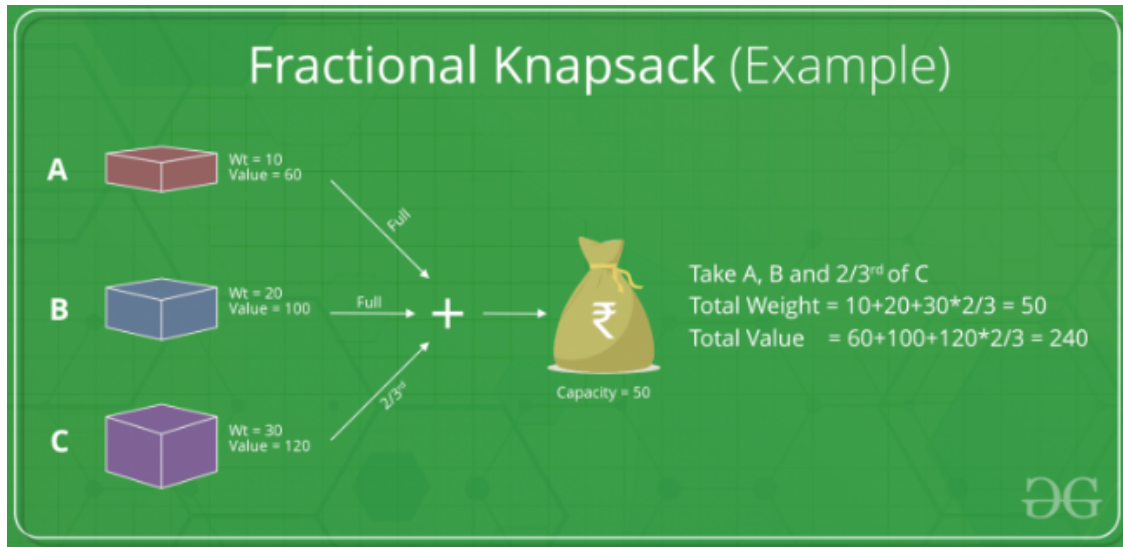


Figure 1: Fractional variant for the knapsack problem.

Minimum number of coins (CMP) problem

Given a number of denominations for coins we need to find the minimum number of coins required to give the change specified. We consider that the amount of coins we have is infinite.

As the knapsack problem, this is a typical problem for DP solution, but both have specific cases that can be solved with Greedy algorithms.

The idea from the Greedy approach is to start from largest possible denomination and keep adding denominations while remaining value is greater than 0.[4]

2 Material and Equipment

In order to this practice we need:

- A working computer.
- A search engine where I can search the answers to the questionnaire.
- Install python in my computer.

3 Development

3.1 Getting to know Python

1. Who created Python?

Guido van Rossum, from the Netherlands, at the late 80's

2. **Explain the game of the name.**

It comes from the fact that its creator is a fan of the British humorist Monty Python.

3. **What is the current version of Python?** 3.9

4. **Explain the term “pythonic”.**

It refers to the code that follows the principles of Python (written by the Python developer Tim Peters). Some of the principles are:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

5. **Explain the difference between the following [5]:**

- List. Is a collection which is ordered and changeable and allows duplicate members.
- Tuple. Is a collection which is ordered and unchangeable and allows duplicate members.
- Set. Is a collection which is unordered and unindexed and does not allow duplicate members.
- Dictionary. Is a collection which is unordered, changeable and indexed and does not allow duplicate members.
- Array. Is a special variable, which can hold more than one value at a time. Can hold many values under a single name, and you can access the values by referring to an index number.

3.2 Greedy Algorithms

3.2.1 0/1 Knapsack problem

Write an algorithm for the 0/1 Knapsack problem. Prove it works correctly with the next parameters:

$$C = 11$$

$$W = \{3, 4, 5, 9, 4\}$$

$$V = \{3, 4, 4, 10, 4\}$$

As mentioned earlier, because it isn't the fractional variant of the knapsack problem, the solution for this, with Greedy approach, will not always be the correct one.

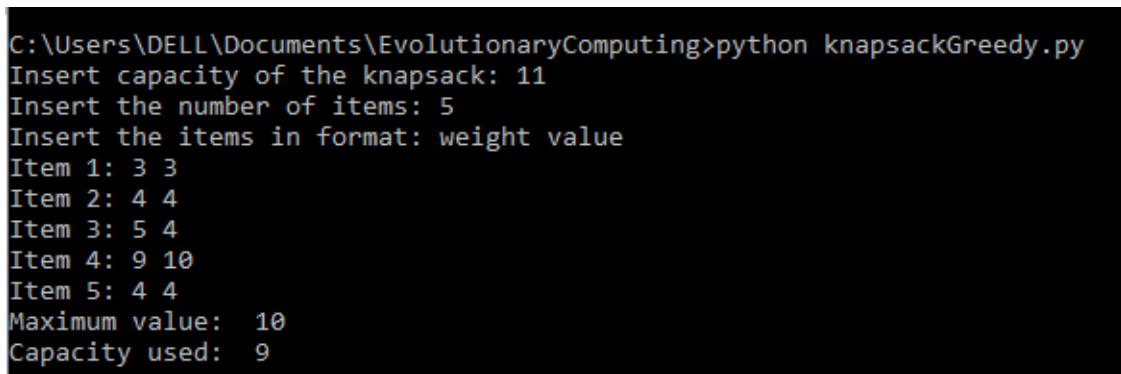
The first thing I think of was the form in which I would ask for the values. I decided to ask for the number of items available to subsequently ask for the values and weights for each of the items.

To store the values and weights of the items I first thought of using a dictionary, then I discarded it because I found inconvenient to access to it using a key and also what if the values or weights are repeated, so instead I used two separate lists (later, conversing with some partners, I saw that I could just use a list containing lists of pair elements).

For the Greedy algorithm I used a while loop and a for loop. The while repeats till the weight of the knapsack doesn't surpasses the value established. The for is in charge of seek for the max value of an item and adds its weight to the knapsack subsequently deleting this value from the list. Is worth to mention that I thought of first ordering the array but I discarded this idea because the list of the weights would be disordered (a consequence of not using the list of list as I mentioned above).

After adding the value of the item to the total value of the knapsack, it is verified if the item surpassed the weight allowed, if it does, the value is subtracted from the total value and changes the value of a flag to end the while loop.

Finally the maximum value of the knapsack as well as its weight are printed to the screen.



```
C:\Users\DELL\Documents\EvolutionaryComputing>python knapsackGreedy.py
Insert capacity of the knapsack: 11
Insert the number of items: 5
Insert the items in format: weight value
Item 1: 3 3
Item 2: 4 4
Item 3: 5 4
Item 4: 9 10
Item 5: 4 4
Maximum value: 10
Capacity used: 9
```

Figure 2: Working program for the knapsack problem.

3.2.2 Minimum number of coins (CMP) problem

Write an algorithm for the CMP problem. Prove it works correctly with the next parameters:

$$T = 9 \qquad D = \{5, 2, 1\}$$

I found the resolution of this problem with Greedy paradigm a lot easier than the knapsack problem.

For the entrance of the parameters I did the same as in the other algorithm.

For the Greedy algorithm I used a for loop that goes through all values of denominations. Inside the loop there's a condition: if the change requested hasn't got to 0, the change left to give is divided for the denomination, the result is the number of coins added to the number of coins necessary to give the change, and the total value of those coins is subtracted to the change. If the change left to give is equals to 0, the loop breaks.

Finally it's printed to the screen the number of coins necessary to give the change.

```
C:\Users\DELL\Documents\EvolutionaryComputing>python cmpGreedy.py
Insert total change required: 9
Insert the number of denominations available: 3
Denomination: 5
Denomination: 2
Denomination: 1
Less number of coins to complete change: 3
```

Figure 3: Working program for the CMP problem.

4 Conclusion

In this practice I could remember some of the facilities that gives python at programming as well as the way to work with its lists and different types of arrays, to call it somehow, in order to facilitate the implementation of an algorithm.

This practice also helped me to understand completely the Greedy paradigm, because even when I saw it in Algorithms Analysis and indeed I thought I'd understood it, I realized it wasn't that way. I knew it existed and I knew how it functions but I had blurry its concept, it means, why it works the way it works. Now I know it is because focuses in finding the locally optimal.

5 Bibliography

- [1]Wikipedia. (2020, February 9). Python (programming language). [Online]. Available: <https://en.wikipedia.org/wiki/Python>
- [2]GeeksforGeeks. Greedy Algorithms. [Online]. Available: <https://www.geeksforgeeks.org/greedy-algorithms/greedyAlgorithmsforSpecialCases>
- [3]GeeksforGeeks. Fractional knapsack problem. [Online]. Available: <https://www.geeksforgeeks.org/fractional-knapsack-problem/>
- [4]GeeksforGeeks. Fractional knapsack problem. [Online]. Available: <https://www.geeksforgeeks.org/fractional-knapsack-problem/>
- [5]w3schools. Python lists. [Online]. Available: https://www.w3schools.com/python/python_lists.asp