Notes on Chapter 12 - Knapsack and Graph Optimization Problems

Swarup Tripathy [*]

June 2022

A curated list of important points for my reference.

1. Notion of an optimization problem provides a structured way to think about solving lots of computational problems.
2. An **Optimization Problem** has two parts
   - An *optimization function* that is to be maximized or minimized. For eg. the airfare between Boston and Istanbul
   - A *set of constraints* that must be honored. For eg. an upper bound on the travel time.
3. **Knapsack Problems**
   - Knapsack → is a simple bag people used to carry on their back-long before 'backpacks' became fashionable.
   - For the burglar problem, the simplest way to find an approximate solution to this problem is to use a *Greedy Algorithm*

     The thief would choose the best item first, then the next best, and continue until he reached his limit. Here, the 'best' could mean the item with the highest value or the item with the lowest weight or the item with the highest value to weight ratio.
4. **The Burglar Scenario for Knapsack Problem**

```
1. Initialising a class Item
(Item Specifications)
    - init
    - name
```

[*]John V Guttag

```
    - value
    - weight
    - str

2. define function
    - to get the value of item
    - to get the weightInverse of item
    - to get the density of item

3. define 'greedy algorithm'
The arguments to be passed are items, maxWeight and keyFunction
    - sorting of items list in itemsCopy depending
        on keyFunction(value or weightInverse or density)
    - initialise totalWeight and totalValue = 0.0
    - for loop in the range of length of itemsCopy
        - if the totalWeight + weight of Item <= maxWeight
            - we append it to result list
            - we now update the totalWeight = totalWeight + the Item Weight
            - we also update the totalValue = totalValue + the Item Value
    - once done we return the result list and totalValue

4. Using greedy algorithm to choose items
    - we define buildItems
        - here we specify the items, values and their respective weights
            Items is a list which comprises of items
            in the format <'item_name',value,weight>
    - we now define testGreedy
        - the arguments are items, maxweight and keyFunction which is
            passed on to 'greedy algorithm'. Here 'taken' is the result list
                and val is the totalValue obtained from the result list.
    - defining the final function 'testGreedys'
        - initialises the building of items and then passes the arguments
            to the function 'testGreedy'. Here, keyfunction is passed as value,
                weightInverse and density that determines the sorting of lists.
```

5. sf