Análise de Algorítimos
Trabalho 3

Luiz Carlos Rumbelsperger e David Beyda

Tarefa 1:

Nesta tarefa se pede para que se encontre a equação de recorrência da função OPT(i, b) que retorna o valor da solução ótima para o *10-Knapsack problem* utilizando os primeiros *i* itens de uma instância fixa para uma mochila de tamanho b. A equação de recorrência é dada por:

$$unit\ value(x) = OPT(i\ -\ 1,\ b - x\ \cdot w_i)$$
$$units\ = arg\ max_{x\ \in\ 0..10,\ x \le b/w_i} unit\ value(x)$$
$$OPT(0,\ b) = 0$$
$$OPT(i,\ 0) = 0$$
$$OPT(i,\ b)\ = unit\ value(units)\ + units \cdot v_i$$

Note que para o caso units = 0, a solução contempla a situação em que nenhuma unidade de *i* será alocada na mochila, caso em que a solução ótima é $OPT(i\ -\ 1,\ b)$.

A solução em si poderia ser dada da mesma forma pela equação:

$$decrease(i)\ = \sum_{j=i+1}^{n} solution(j) \cdot w_j$$
$$decrease(n) = 0$$
$$solution(i)\ = arg\ max_{x\ \in\ 0..10,\ x \le b/w_i} unit\ value(x\ + \frac{decrease(i)}{w_i})$$

A função retorna, para cada item *i,* o número de unidades que devem ser alocadas para *i* na solução ótima. Note que $unit\ value(x\ + \frac{decrease(i)}{w_i}) = OPT(i\ -\ 1,\ b - x\ \cdot w_i - decrease(i))$.

Tarefa 2:

Esta tarefa pede que apresentemos a função principal de nosso código, que gera as tabelas de memoização para a resolução do problema. A função a seguir foi definida para a classe *ProblemInstance*, que modela uma instância do problema:

```
def solve(self):
        """
```

```
        Solves the knapsack problem using DP (Dinamic Programming)
        in an iterative form.
        To fill each element of the DP matrix, it uses one element
        to the left and up to 10 elements up, which means it finds
        the solution in O(n^2).

        :return: (best solution value, quantity of each item)
        :rtype: tuple
        """
        # start a matrix M [n_item][backpack_size] to hold solution
values
        M = [ [0 for x in range(self.backpack_size + 1) ] for y in
range(self.n_items + 1)]
        # start matrix L to hold the items quantities of each solution
        L = [ [[] for x in range(self.backpack_size + 1) ] for y in
range(self.n_items + 1)]

        for size in range(1, self.backpack_size+1):
            for n_item in range(1, self.n_items+1):
                # here the magic happens
                # with_item will hold all values possible with
                # 1..n (n<=10) items of type item_n in the backpack
                # each time we access w or v we use [n_item-1] because
n_item
                # is zero indexed, meaning M[n_item = 0] represent no
items,
                # and M[n_item = 1] represent solutions including only
the first item.
                with_item = []
                for i in range(1, 10):
                    discounted_weight = size - i * self.w[n_item-1]
                    if discounted_weight < 0:
                        # does not fit in the backpack
                        continue
                    with_item.append(M[n_item-1][discounted_weight] +
i*self.v[n_item-1])
```

```
                max_idx, max_value = self.get_max([M[n_item-1][size],
*with_item])

                # if max_idx:
                L[n_item][size] = copy.deepcopy(L[n_item-1][size -
max_idx * self.w[n_item-1]])
                L[n_item][size].append(max_idx)
                # else:


                # L[size].append(max_idx)
                M[n_item][size] = max_value

        return M[-1][-1], L[-1][-1]
```

Soluções:

Melhor valor obtido na instância 1: 525
    9 do item numero 2
    5 do item numero 10

Melhor valor obtido na instância 2: 2770
    9 do item numero 2
    3 do item numero 3
    9 do item numero 11
    4 do item numero 12
    9 do item numero 15
    9 do item numero 19

Melhor valor obtido na instância 3: 44
    4 do item numero 1

Melhor valor obtido na instância 4: 30
    5 do item numero 1

Melhor valor obtido na instância 5: 69
    8 do item numero 1
    9 do item numero 2
    2 do item numero 3

Melhor valor obtido na instância 6: 107

1 do item numero 1
1 do item numero 4

Melhor valor obtido na instância 7: 333

9 do item numero 4

Melhor valor obtido na instância 8: 2770

2 do item numero 1
9 do item numero 2
1 do item numero 4
9 do item numero 7
8 do item numero 10
9 do item numero 12
8 do item numero 16