**Assignment 1 part 1: Knapsack Problem**

Student: Mirza Sijamić

Student ID: msc25001

# Explanation of the problem:

* + 1. Give the representation of a solution (answer) of the problem, as explained during the course.

A possible representation of a solution of the problem is as follows:

X = {X1, X2, X3, …, Xn}

Where n is a number of items, domain of is Xi = {0,1} which means it can be in the bag or not.

* + 1. Give the equation of the objective function (what we want to maximize)

We want to maximize the sum of profits where Xi is binary decision variable and bi is the benefit of item.

* + 1. Give the equation for the restriction(s) of the problem

Restriction of the problem is that sum of weights of the included items must be smaller than 420. Xi is binary decision variable and Wi is the weight of item i.

* + 1. What is the branching factor (b) and the maximum depth (m)? Give an explanation of your answer.

At each node the algorithm generates two child nodes, so the branching factor is b=2.

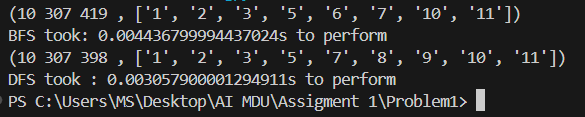
Maximum depth of these search algorithms is equal to the number of items, so maximum depth m=n, where n is the number of items.

# Comparison of the algorithms

1. Comparison of the time expended by the algorithms

To compare the time of these search algorithms I used time library in python. To calculate the time each algorithm took I made two variables, timeb (time before the start) and timea (time after finishing), and then subtracted them to get the time expended.

The results were as following:



We can see that DFS runs faster than BFS algorithm.

1. Comparison of the space used in memory at a time by the algorithms

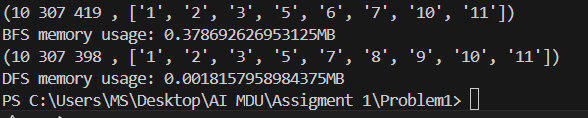
To compare the memory usage of these two algorithms I used tracemalloc library in Python. To do this I used the following commands:

tracemalloc.start()

current, peak = tracemalloc.get\_traced\_memory()

tracemalloc.stop()

And then just divided peak by 1024\*1024 (peak/1024\*1024) and printed it.



From this picture we can see that DF uses less memory than BFS