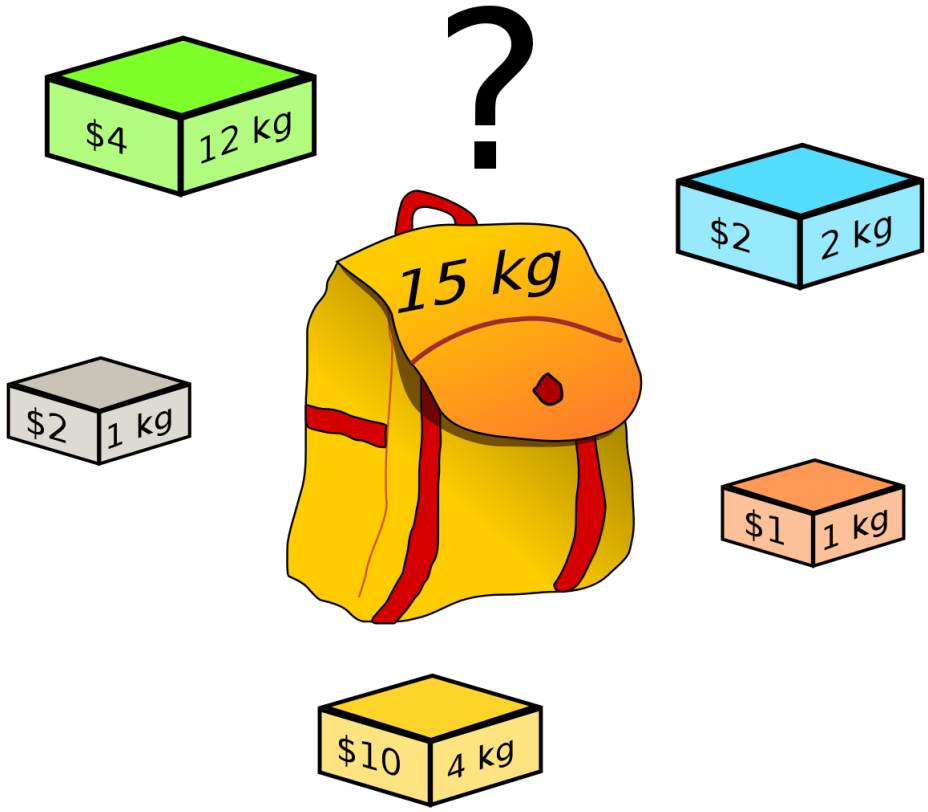
1. Introduction to the Knapsack Problem

The knapsack problem is a well-known problem in industry. If we have a set of items, each with a weight and a value, we need to determine the number of each item to include in a collection such that the total weight is less than or equal to a given limit and the total value is as large as possible.

Example: lets say if we want to go on a long hike – we need to make sure that the knapsack we carry does weigh too much while we want to also make sure we have all necessary items needed inside the knapsack so that the journey is smooth. Hence we restrict the weight to a limit while also making sure we have maximum number of items in the knapsack.



Source: Wikipedia

2. How we tackled the problem?

```
items = {}

# Create random items and store them in the items' dictionary.

for i in range(NBR_ITEMS):

    items[i] = (random.randint(1, 10), random.randint(1, 100))
```

The above code created a knapsack of 100 items.

Since there are 100 items, we should have an individual of 100 values. Since we are given a choice to either pick an item once, twice or not pick it at all – we can have values 0,1 and 2 in the individual. We used the below code for handling this – wherein I registered the function MultipleConstrained for the toolbox.

toolbox.register("MultipleConstrained", random.randint, 0, 2)

The fitness function Knapsack_Fitness calculates the value and weight inside every individual and returns the value and weight, if the weight is lesser than the max_weight allowed i.e. 1000. Value is taken as the first value of the tuple returned by the function because we want to maximize value while keeping weight restricted at 1000. If the weight exceeds 1000, the fitness function penalizes the value and returns value- ((weight-Max_weight)*alpha). We penalize this way to account for the excess weight. We restrict weight at 1000 if the weight is above the max_weight constraint. Hence, the function returns the weight of 1000 in that case.

3. Experiments

In all experiments, we keep random seed constant at 42, population size constant at 500 and make changes to the probability of crossover, probability of mutation, maximum generations and alpha value for penalizing for excess weight.

1. We take the below parameters for the first experiment:

   Probability of Crossover = 0.7

   Probability of Mutation = 0.05

   Maximum Generations = 100

   Alpha for penalizing for excess weight = 1.5

2. We take the below parameters for the second experiment. Here we identify the effect of changing probability of crossover, mutation and alpha value.

   Probability of Crossover = 0.9

   Probability of Mutation = 0.01

   Maximum Generations = 100

   Alpha = 2

3. We take the below parameters for the third experiment. Here, we only increase max generations while keeping other things constant.

   Probability of Crossover = 0.9

   Probability of Mutation = 0.01

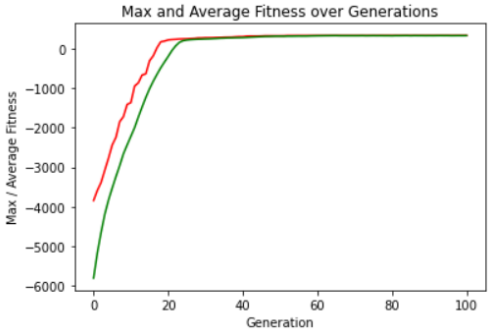
   Maximum Generations = 300

   Alpha = 2

4. Results Section

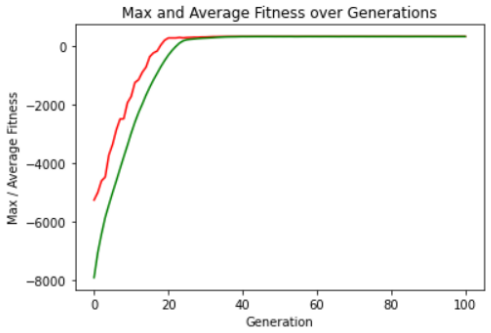
1. Here, we get a fitness score of 335.

```
Best Ever Individual =  [2, 1, 0, 0, 2, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2, 1, 0, 2, 2,
0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 1, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 2, 0, 2, 0, 2, 0, 2, 0, 1, 0]
Weight of the best individual: 1000
Value of the best individual: 335.0
```



2. Here, we get a fitness score of 321.

```
Best Ever Individual =  [2, 0, 2, 0, 2, 0, 1, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0, 2, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 2, 0, 0, 1, 0, 1, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 2, 2, 0, 1, 0, 1, 0, 2, 0, 0, 0]
Weight of the best individual: 996
Value of the best individual: 321
```



3. Here, we get the fitness score of 331.

```
Best Ever Individual =  [2, 0, 2, 0, 2, 0, 1, 0, 0, 2, 2, 0, 0, 2, 0, 0, 0, 2, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0, 2, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 1, 0, 1, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 0, 1, 0, 1, 0, 2, 0, 1, 0]
Weight of the best individual: 1000
Value of the best individual: 331
```