INTRODUCTION

Genetics algorithms are a very good aproach to optimization of a given problem, in our case we will try to get a program to optimizate a typical Knapsack problem.

ALGORITHMS SELECTED AND BEHAVIOUR

Selection algorithm

We chose tournament selection, this algorithm works in this way:

First of all we have to define the value of the variable “l”, it defines the size of our “arena”, this mean how many individuals are going to participate in the competition.

Once defined this terms just we have to select randomly a subpopulation of size “l” from our population and choose the 2 best individuals

Recombination algorithm

We chose one point crossover, the operation is how is next:

We want to get 2 children of 2 parents with a propability of mutation “p”, this means sometimes will not be crossover, therefore, in this case our children will be the same than parents.

We have to select randomly a point of crossover, then we apply recombination just moving the next genes to the oposite children, let’s see an example of this:

Let’s imagine our random index is 3 and the algorithm decided recombinate, so our children will be like this:

Parents	Our point of crossover	Children
0 1 0 1 1 1	0 1 0 | 1 1 1	0 1 0 0 1 0
1 1 0 0 1 0	1 1 0 | 0 1 0	1 1 0 1 1 1

Mutation algorithm

Sometimes one of the main problem of genetics algorithms is lack of diversity, it’s because through the diferent generations the population tends to a converge and loses special characteristic of individuals, it’s in part inspired by Darwin theory too, sometimes mutations in one or more specific genes are produced in the nature.

In order to fix this situation we add a mutation property, there are a lot of types of mutations, but we selected one point locus mutation, it works like this:

From a random index and a probability of mutation we alter (or not) that point to the oposite, in our case we are working with binary values, so if we have a 0 will change to 1, and the same backwards.

Sergio Barbero, Maaz Ashiq

Fitness evaluation

Once got here we have to evaluate our solution, in order to make it we reached to this algorithm:

- In the one hand we have to penalizate those solutions that don't accomplish the aim of maximum weight.
- In the oher hand we have to accumulate the values of each item took in one individual (unpolished fitness)

A good aproach is:

1. to sum all the weights took in one individual T
2. Then evaluate if it's larger than our maximum allowed value
3. If it is then we get the difference between our weight in the knapsack and the maximun value, and we multiply that diference by a parameter of penalization (in the program is initialized to 10).
4. Finally we substract that value to the "unpolished fitness" and we will get our final fitness.

In this way we penalizate to a individual is overtaking our maximum weight in a proportional way.

The genetic algorithm algorithm as a whole

We explained how the different parts of the algoritm work but not as a unic algorithm, as a entire genetic algorithm.

Here we have the pseudocode of our main function in our program

```
SIZE = population size
P = Random population of SIZE individuals
while not done
    C = create empty child population of SIZE individuals
    while not enough individuals in C
        parent1 = select parent
        parent2 = select parent
        child1, child2 = crossover(parent1, parent2)
        mutate child1, child2
        evaluate child1, child2 for fitness
        insert child1, child2 into C
    end while
    P = C
end while
```