**Genetic Algorithm: The Knapsack Problem**

**Problem**:

We are using Genetic Algorithm to solve the Knapsack Problem. Knapsack problem states that, from a set of items, where each item has a weight and a value, we must determine the number of items to be include in a bag or collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

**Implementation:**

*Genotype:* A gene is encoded with a string of 0’s and 1’s, where 0 represents absence from the solution and 1 represents its presence. Each gene represents a solution and is then evolved during time with either mutation or crossover.

*Gene expression:* Every index of the gene represents an item that has a weight and value associated to it.

*Fitness Function:* The fitness function gives the total weight of the gene which is then compared to the capacity of bag. The fitness of the gene is calculated on the basis of 0 or 1. We take the indexes of 1 from the gene and lookup for items using that index and take the sum of their values and weight separately. If the weight exceeds the capacity, the fitness score becomes 0 or else we store the total value as its fitness score.

*Mutation:* A user inputs the mutation probability and based on that value we decide whether to mutate a gene or not. When mutation occurs, we take a random index at which mutation the value is changed from 0 to 1 or 1 to 0, i.e. we either include item or remove the item of that index. For e.g.:

Gene: 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0

Perform mutation at random index 4

Mutated Gene: 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0

*Crossover:* Two parents and an index. Is selected randomly We swap values from the beginning till that index of parent 1 with parent 2 and from that index of parent 2 with parent 1 resulting in creation of two children. For e.g.:

Parent 1: 0110001 | 1100  
 Random index 7 is selected  
 Parent 2: 1100010 | 0011

We swap the values that gives us two children

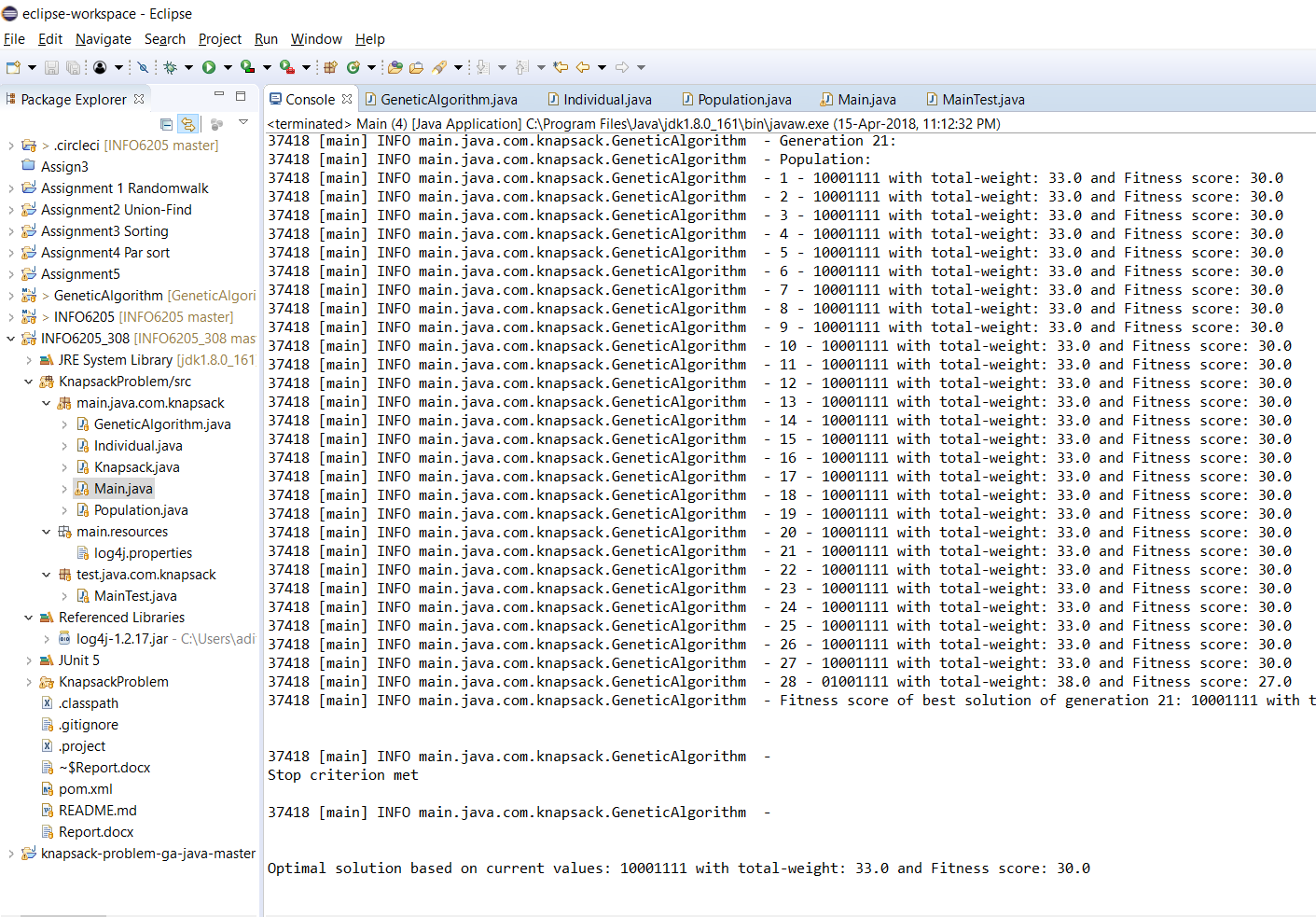
Child 1: 0110001 | 0011

Child 2: 1100010 | 1100

*Evolution:* An initial population of 10 genes is created and stored in a list. Children are created using crossover and they experience mutation depending on user input. Fitness values are calculated for each gene and depending on user input for maximum population the next population is maintained. Only the fittest candidates are kept and rest are removed from the population. The process of evolution is repeated again with the remaining population which doubles the population and then is reduced based on the user input.

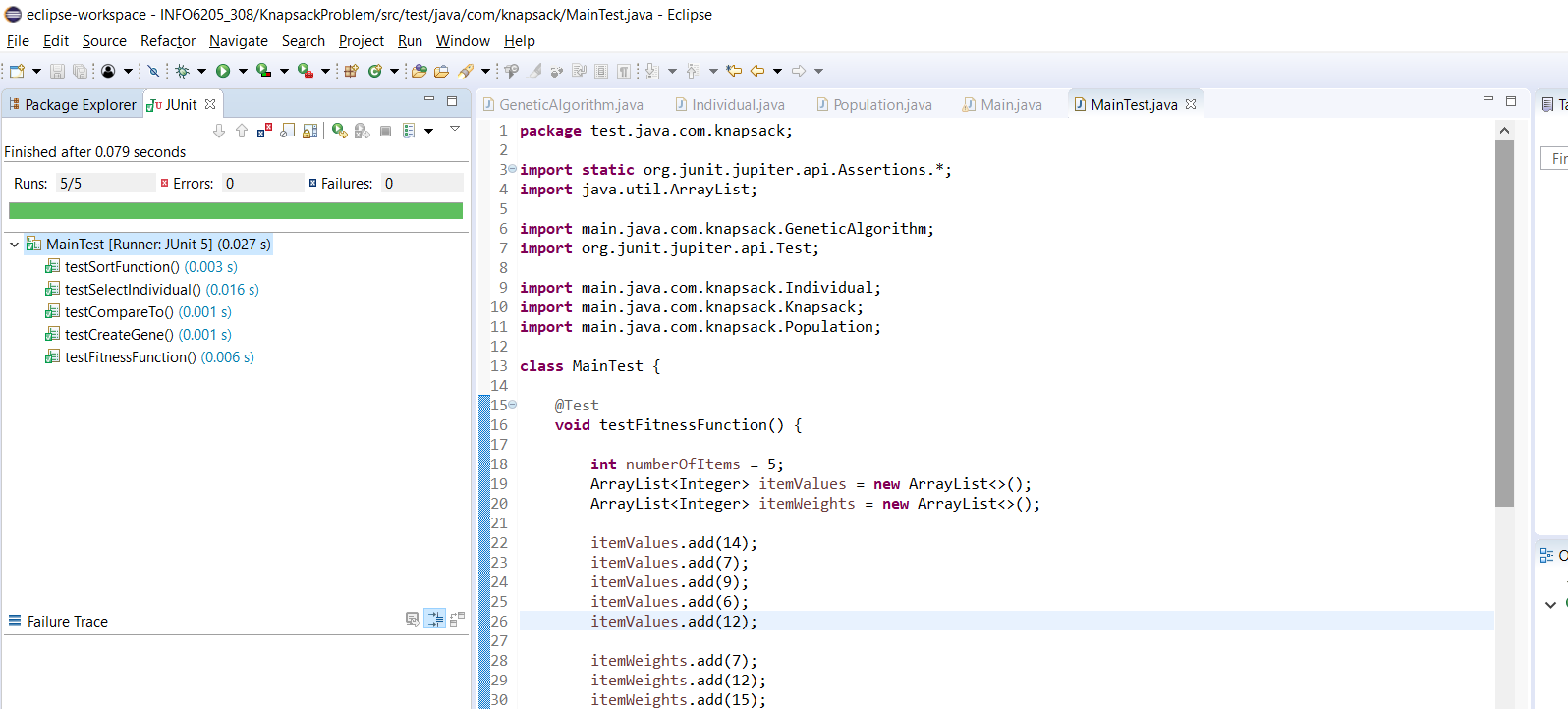
The fitness score and gene is displayed of all the remaining candidates. The highest fitness value of the remaining population is then stored and displayed separately. If the highest fitness score is not changed for 10 generations the algorithm terminates or else it executes according to the user input.

**Solution:**

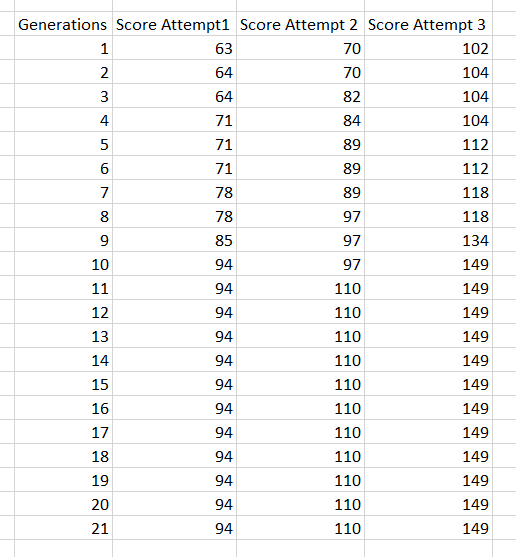
****

**Test Case**

Here is a screenshot of successful execution of unit tests.



Graph showing results for sample data



X-axis: Number of generations

Y-axis: Best Fitness Score