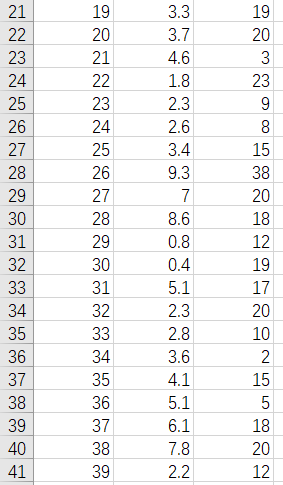
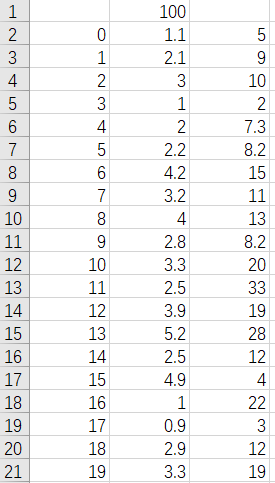
Our group aim to solve knapsack problem.

We put many goods with different volume and value into a bag with fixed size. Our program aims to get a good solution to the problem that, what’s the biggest value we can put in the bag, and what goods we choose.

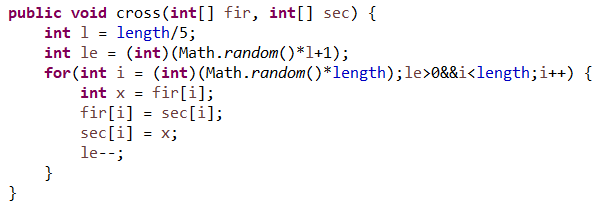
There is a csv file contains all information about goods. The number 100 in the first row shows the size of bag. First line shows the number of goods, the second line shows volume and the third shows their value.

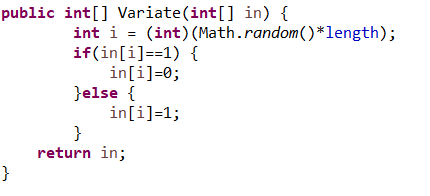


After we read all goods into a Goods list, it’s time to get the first generation. Every individual unit has it’s binary codes (gene) and it’s fitness. To get a new gene, we use Math.random() method to get a int array contains 0 and 1. 0 means not put in and 1 means put in. Fitness is simply the total value in the bag.

We use the Binary Search Tree as the data structure because we don’t need to sort every time when a new generation comes, also it’s easy to weed out bad individual use DeletMin(). In BST, we use gene as key, and fitness as value.

After we put first generation into BST, it’s time to cross their genes and variation. We set cross rate as 0.7 and variation rate as 0.1.

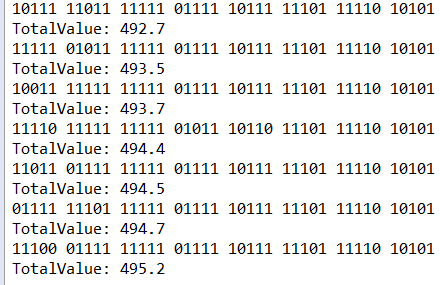




Then we use DeletMin() to reserve top 10% of the generation (delete the other 90%).

When these all done, we get new random genes and add them to genes’ group. The second generation appears and the loop begins.

After hundreds generations, we get some genes with high fitness.



In this picture shows some of the output. As we can see, No. 14,20,27,33,35,37 (starts whit No.0) is never selected. We look back to the csv file and find that their value per volume is low compared to others.