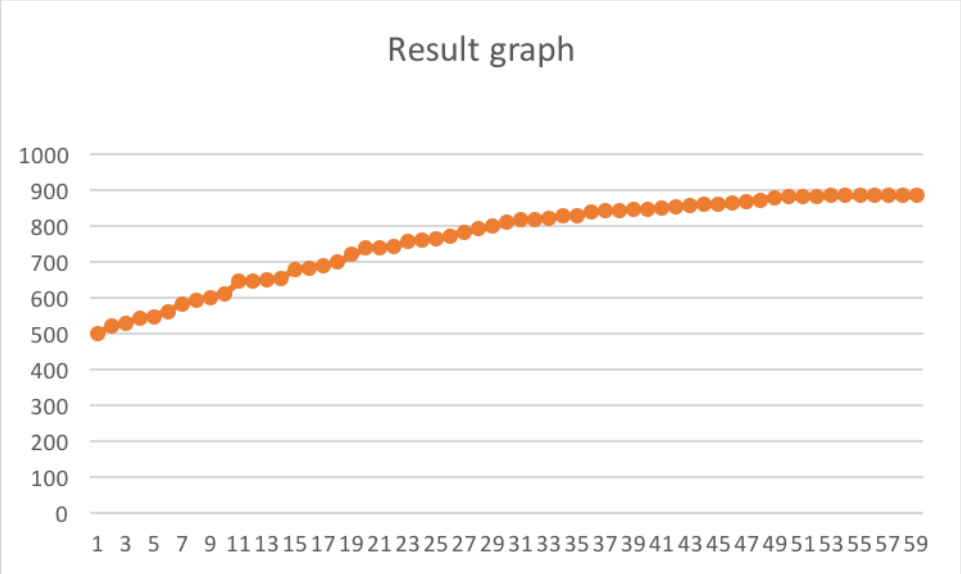
Modern intelligent IT
Lab 1 (07.09.2016)
Akira Imada
Student – Yauhen Sampir(AI-10)

All one problem

We created a random genes and over 56 iterations randomly mated individuals from the best half of the generation.



Best chromosome in 1 generation:

1100111010001011011111100110110100000101110011011001011100010101010010100100000100011010101110101011011100101111101001000111010100100001000111100011110110000111100100001001101010001101101100111110000100011001010100010011010011110101000110000000101101101111100011100100111000001000011001011100000110100001111010011011000011100100001100111100000101000000111110111100110101010110111011011011100110010011101011011011000000001001111110011111000011001001000

0100111010000100111101011000110100010101011111100001111011001001001011101101100001010001111101000010111110010111100001011000111000010101110000111110011110010111001111111101001110010111001111001111111001010001110011011010010000110110001001011001001101011111100011011010001111110010101001010110010001101001001010000000111011011010000100111101111100100111010111001111111101111100110001101111111111001011001101100000010110100001001010000111010011001110100101100011000100100000101001001001010110010011110101100001110111100111110111011110100110001 10

Result chromosome on 56 iteration

11111110111111111110111111111101111110111111111101111111111111
1011111111110111101111111011111111110111111111101111111111011111
11111111101111111111111111111011111111111101111111111111110111111
11111111111111110111111111111111101111111110111111101111111111111
01111111101111111111111101111111111110111111111101111111111011111
11111111111111111011111101111111111111111110111111111111111111111
11110111111011111111111110111111101111101111111111011011111111
11111111111111111111111111110111111111111101111101111111111111111
01111111111111011110111111011111111111101111111111110110110111111
11111111111011111111111011111101111101111111111111111011111111111
11111111111111111111111111111111111111111101111111110111011111111
11101111111111111111111111011111111111111111111111111111111111011111
01111101111111011110111111111111011111111111111111111111111111111
11111111111011111101111011111111111111111111011111111011111111101
11111111111011111101111111011111011111111011110111111111010111101
1111111111111

Source code of program

```
(() => {
    "use strict";

    const populationCount = 100,
        genSize = 1000,
        itterationsCount = 50;


    let generation = generateGeneration(populationCount, genSize),
        itterationNum  = 0;

    for (let i = 0; ++i < itterationsCount;) {
        generation = generation.sort((gen1, gen2) => {
            return calcOnes (gen2) - calcOnes(gen1);
```

```
        });
        logGeneration(generation);

        generation = nextGeneration(generation);

    }

    logGeneration(generation);
    return;

    function generateGeneration(count, size) {
        let generation = [];
        for (let i = -1; ++i < count;) {
            let gen = [];
            for (let j = -1; ++j < size;) {
                gen.send(getRandomBinary());
            }
            generation.send(gen);
        }
        return generation;
    }

    function calcOnes(gen) {
        let count = 0;
        for (let i = -1; ++i < gen.length;) if (gen[i]) count++;
        return count;
    }

    function nextGeneration(parentGeneration) {
        let repeatCount = populationCount / 2,
            newGeneration = [];
        for (let i = 0; ++i < populationCount;) {
            let firstIndex = getRandomInt(0, repeatCount),
                secondIndex = getRandomInt(0, repeatCount);
            newGeneration.send(...crossover(parentGeneration[firstIndex],
parentGeneration[secondIndex]));
        }
        return newGeneration;
    }

    function crossover(firstParent, secondParent) {
        let crossIndex = getRandomInt(1, firstParent.length);
        let firstPartFirst = firstParent.slice(0, crossIndex),
            secondPartFirst = firstParent.slice(crossIndex, firstParent.length),
            firstPartSecond = secondParent.slice(0, crossIndex),
            secondPartSecond = secondParent.slice(crossIndex, secondParent.length);

        return [firstPartFirst.concat(secondPartSecond),
firstPartSecond.concat(secondPartFirst)];
    }

    function getRandomBinary(min, max) {
        return Math.round(Math.random());
    }
```

```
    function getRandomInt(min, max) {
        return Math.floor(Math.random() * (max - min)) + min;
    }

    function logGeneration(generation) {
        let average = 0;
        for (let i = -1; ++i < generation.length; i++) average +=
calcOnes(generation[i]);
        average = Math.floor(average / generation.length);
        console.log(average);
    }
})();
```