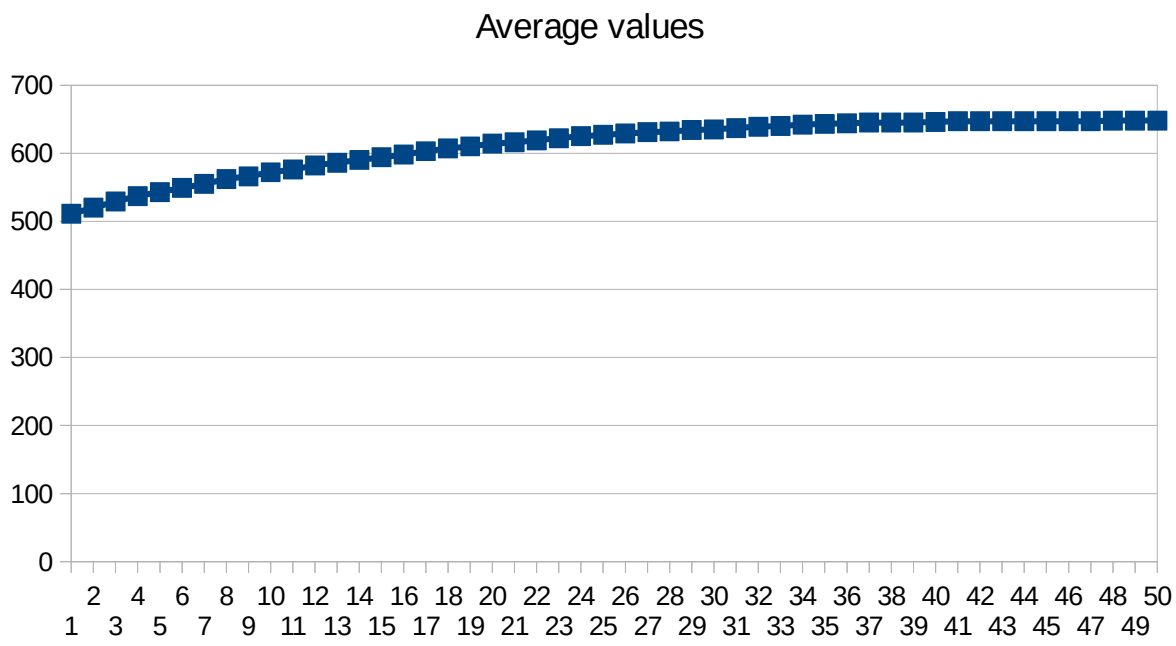
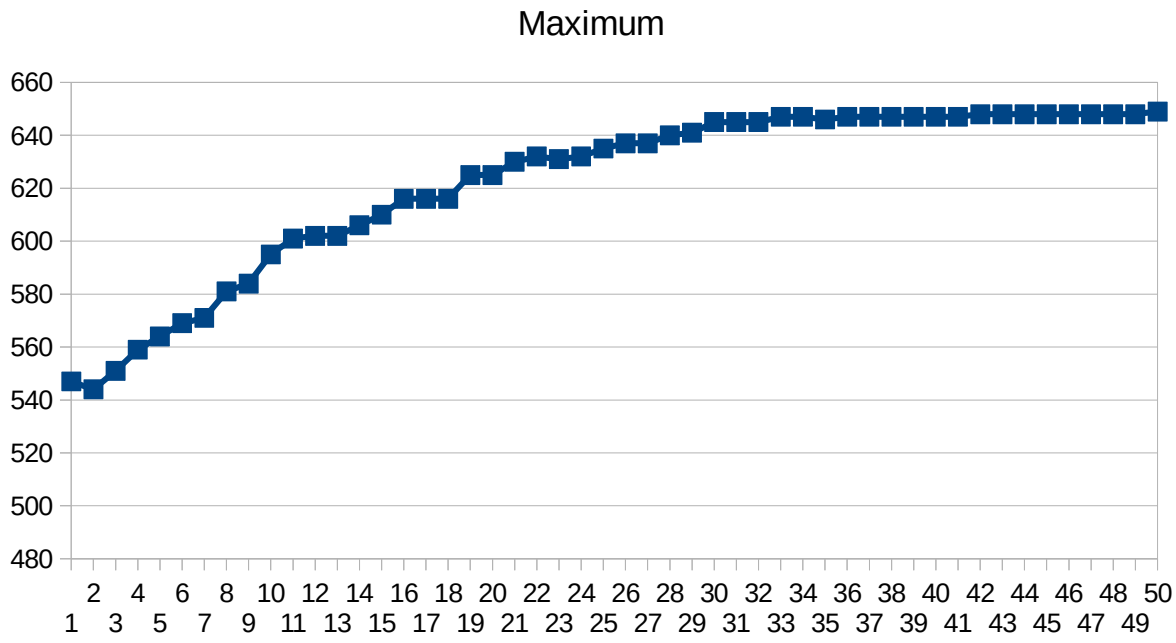
1. Diagram: Average value of one's in chromes



2. Diagram: Maximum value of one's in chromosome



We can see that after 50 iteration our algorythm create chromosomes with more that 60% good gens. When we run our program next time we can get another data, but in common range.

Source code:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import random
```

```
def create_population(gens, chromos):
    population = []
    for chrom in xrange(chromos):
        population.append([])
        for gen in xrange(gens):
            population[chrom].append(random.randint(0, 1))
    return population
def calc_ones(arr):
    count = 0
    for i in arr:
        if i == 1:
            count += 1
    return count
def find_best_chromos(population):
    list = []
    for chrom in population:
        list.append((calc_ones(chrom), chrom))
    buble_sort(list)
    r = []
    for item in list[len(population)/2:]:
        r.append(item[1])
    return r
def create_childs(pop):
    size = len(pop)
    par1, par2 = random.randint(0, size-1), random.randint(0, size-1)
    cut_point = random.randint(0, len(pop[0]))
    child1 = pop[par1][:cut_point] + pop[par2][cut_point:]
    child2 = pop[par2][:cut_point] + pop[par1][cut_point:]
    return child1, child2
def buble_sort(A):
    for i in range(len(A)):
        for k in range(len(A) - 1, i, -1):
            if A[k][0] < A[k - 1][0]:
                swap(A, k, k - 1)
def swap(A, x, y):
    tmp = A[x]
    A[x] = A[y]
    A[y] = tmp
def get_avg_count(population):
    chromos = len(population)
    average_list = []
    for chrom in xrange(chromos):
        cur_counter = calc_ones(population[chrom])
        average_list.append(cur_counter)
    return sum(average_list) / len(average_list)
def maximum(population):
    max = 0
    for chrom in population:
        counter = calc_ones(chrom)
        if max < counter:
            max = counter
    return max
def main():
    population = create_population(1000, 100)
    # print(population)
    for i in xrange(50):
        best_chromos = find_best_chromos(population)
        # print("best", best_chromos)
        population = create_new_generation(best_chromos)
```

```
        # print("new gen", population)
        avg = get_avg_count(population)
        max = maximum(population)
        print(str(i) + ";" + str(avg) + ";" + str(max))
def create_new_generation(best):
    new_generation = []
    for iter in xrange(len(best)):
        ch1, ch2 = create_childs(best)
        new_generation.append(ch1)
        new_generation.append(ch2)
    return new_generation
if __name__ ==
"__main__":
    main()
```

Result:

0	511	547
1	520	544
2	529	551
3	537	559
4	543	564
5	549	569
6	555	571
7	562	581
8	566	584
9	572	595
10	576	601
11	582	602
12	586	602
13	590	606
14	594	610
15	598	616
16	603	616
17	607	616
18	610	625
19	614	625
20	616	630
21	619	632
22	622	631
23	625	632
24	627	635
25	629	637
26	631	637
27	632	640
28	634	641
29	635	645
30	637	645
31	639	645
32	640	647
33	642	647
34	643	646
35	644	647
36	645	647
37	645	647
38	645	647
39	646	647
40	647	647
41	647	648
42	647	648
43	647	648
44	647	648
45	647	648
46	647	648
47	648	648
48	648	648
49	648	649