DENIS RAMSKIY
GROUP II-11

Class of working with gene

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Threading;

namespace GenAlg
{
    class Gen
    {
        List<int> chromosome;
        int fitness;
        int used;

        public Gen()
        {
            used = 0;
            chromosome = new List<int>();
            for (int i = 0; i < 1000; i++)
            {
                Random rnd;
                if(i%2==0)
                    rnd = new Random(DateTime.Now.Millisecond+i*i*i);
                else
                    rnd = new Random(DateTime.Now.Millisecond-i*i*i);
                int dice = rnd.Next(0, 2);
                chromosome.Add(dice);
                if (dice == 1)
                    fitness++;
            }

        }
        public Gen(List<int> crom)
        {
            used = 0;
            this.chromosome = new List<int>();
            for (int i = 0; i < crom.Count; i++)
            {
                if (crom[i] == 1)
                    this.fitness++;
                this.chromosome.Add(crom[i]);
            }
        }
        public Gen(Gen arg)
        {
            this.chromosome = new List<int>();
            for (int i = 0; i < arg.chromosome.Count; i++)
                this.chromosome.Add(arg.chromosome[i]);
            this.fitness = arg.fitness;
            this.used = arg.used;
        }

        public static Gen operator + (Gen arg_1,Gen arg_2)
        {
            for (int i = 0; i < arg_1.chromosome.Count; i++)
                arg_1.chromosome[i] = arg_2.chromosome[i];
```

```
            arg_1.fitness = arg_2.fitness;
            return arg_1;
        }

        public int get_fitness()
        {
            return fitness;
        }
        public int get_gen(int i)
        {
            return chromosome[i];
        }

        public int get_used()
        {
            return used;
        }
        public void inkr_used()
        {
             used++;
        }
    }
}
```

Class of working with generation

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace GenAlg
{
    class Generation
    {
        List<Gen> parens_generation;
        int general_fitness;

        public Generation()
        {
            parens_generation = new List<Gen>();
            general_fitness = 0;
            for (int i = 0; i < 100; i++)
            {
                Gen temp = new Gen();
                parens_generation.Add(temp);
                general_fitness += temp.get_fitness();
            }
            sort();
        }
        public void sort()
        {
            for (int i = 0; i < parens_generation.Count; i++)
                for (int j = 0; j < parens_generation.Count - 1; j++)
                    if (parens_generation[j].get_fitness() < parens_generation[j +
1].get_fitness())
                    {
                        Gen temp = new Gen(parens_generation[j]);
                        parens_generation[j] += parens_generation[j + 1];
```

```
                    parens_generation[j + 1] += temp;
                }
        }
        public Gen get_gen(int i)
        {
            return parens_generation[i];
        }
        public int get_general_fitness()
        {
            return general_fitness;
        }

        public List<Gen> get_parents_tranc()
        {
            List<Gen> parents = new List<Gen>();
            int rnd_tik = 17;
            while (parents.Count < 2)
            {
                for (int i = 0; i < 100; i++)
                {
                    Random rnd = new Random(DateTime.Now.Millisecond + rnd_tik * rnd_tik);
                    if (rnd.NextDouble() < (Double)parens_generation[i].get_fitness() /
general_fitness)
                    {
                        parents.Add(parens_generation[i]);
                        parens_generation[i].inkr_used();
                        break;
                    }
                    rnd_tik += 7*i;
                }
            }
            return parents;
        }
        public List<Gen> get_parents_rul()
        {
            List<Gen> parents = new List<Gen>();
            int rnd_tik = 17;
            while (parents.Count < 2)
            {
                Random rnd = new Random(DateTime.Now.Millisecond + rnd_tik * rnd_tik);
                int a=rnd.Next() % 50;
                parents.Add(parens_generation[a]);
                parens_generation[a].inkr_used();
                rnd = new Random(DateTime.Now.Millisecond - rnd_tik * rnd_tik);
                a = rnd.Next() % 50;
                parents.Add(parens_generation[a]);
                parens_generation[a].inkr_used();
                rnd_tik += 19;
            }
            return parents;
        }
        public Gen get_child(List<Gen> perants)
        {
            int rnd_tik = 17;
            List<int> cild = new List<int>();
            for (int i = 0; i < 1000; i++)
            {
                Random rnd = new Random(DateTime.Now.Millisecond - rnd_tik * rnd_tik );
                if (rnd.Next(0, 2) == 1)
                    cild.Add(perants[0].get_gen(i));
                else
```

```
                    cild.Add(perants[1].get_gen(i));
                rnd_tik *= i;
            }
            Gen child = new Gen(cild);
            return child;
        }
        public void new_generation()
        {
            File.WriteAllText("carent_fitness.txt", "");
            File.WriteAllText("used.txt", "");
            for (int i = 0; i < 100; i++)
            {
                File.AppendAllText("carent_fitness.txt",
this.get_gen(i).get_fitness().ToString() + "\r\n");
                File.AppendAllText("used.txt", this.get_gen(i).get_used() + "\r\n");
            }


            List<Gen> new_generation = new List<Gen>();
            int new_fitnes = 0;
            for (int i = 0; i < 100; i++)
            {
                Gen temp = new Gen(get_child(get_parents_tranc()));
                new_generation.Add(temp);
                new_fitnes += temp.get_fitness();
            }
            this.general_fitness = new_fitnes;
            parens_generation = new_generation;
            sort();
        }
    }
}
```

Main program class

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace GenAlg
{
    class Program
    {
        static void Main(string[] args)
        {
            int tik = 0;
            Generation work = new Generation();

            File.WriteAllText("best_fitness.txt", "");
            File.WriteAllText("generation_fitness.txt", "");
            while (work.get_gen(0).get_fitness() < 800)
            {
                if (tik == 1 || tik == 20)
                    write(work);
                File.AppendAllText("best_fitness.txt",
work.get_gen(0).get_fitness().ToString()+"\r\n");
                File.AppendAllText("generation_fitness.txt",
work.get_general_fitness().ToString() + "\r\n");
```

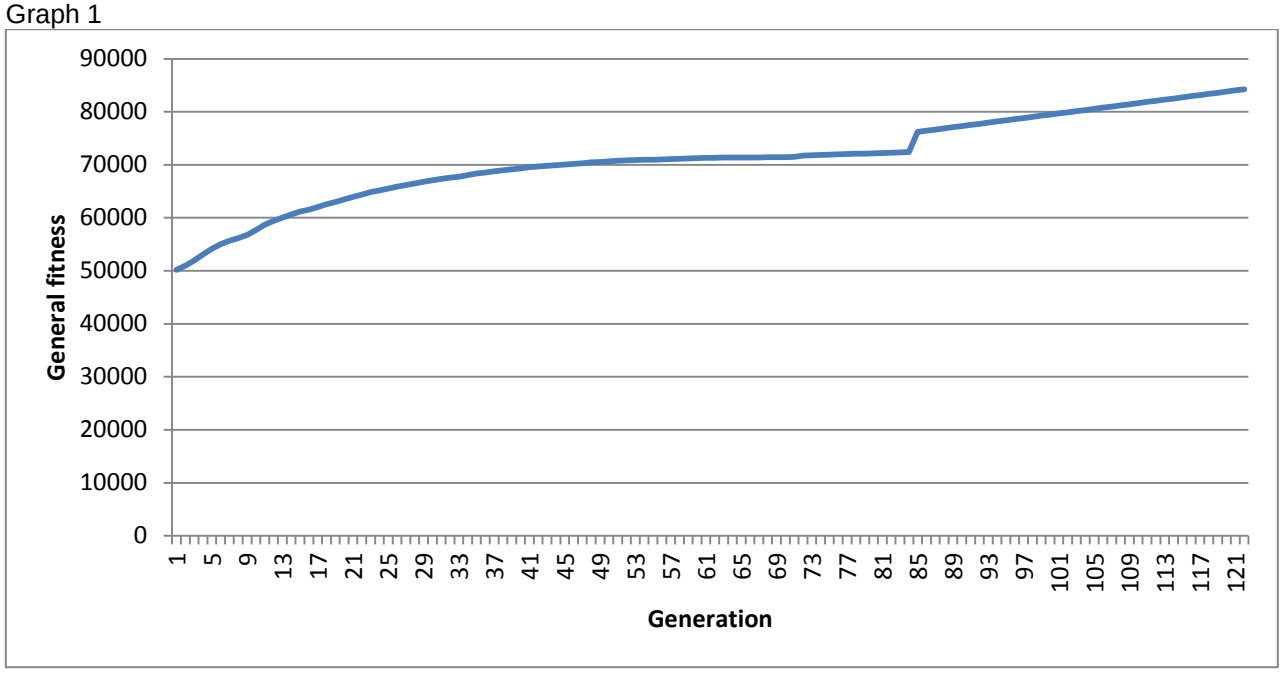
```
                Console.WriteLine(work.get_general_fitness());
                work.new_generation();
                tik++;
            }

        }

        static public void write(Generation work)
        {
            File.WriteAllText("carent_fitness.txt", "");
            File.WriteAllText("used.txt", "");
            for (int i = 0; i < 100; i++)
            {
                File.AppendAllText("carent_fitness.txt",
work.get_gen(i).get_fitness().ToString() + "\r\n");
                File.AppendAllText("used.txt", work.get_gen(i).get_used() + "\r\n");
            }
        }
    }
}
```

truncate-method

Graph 1



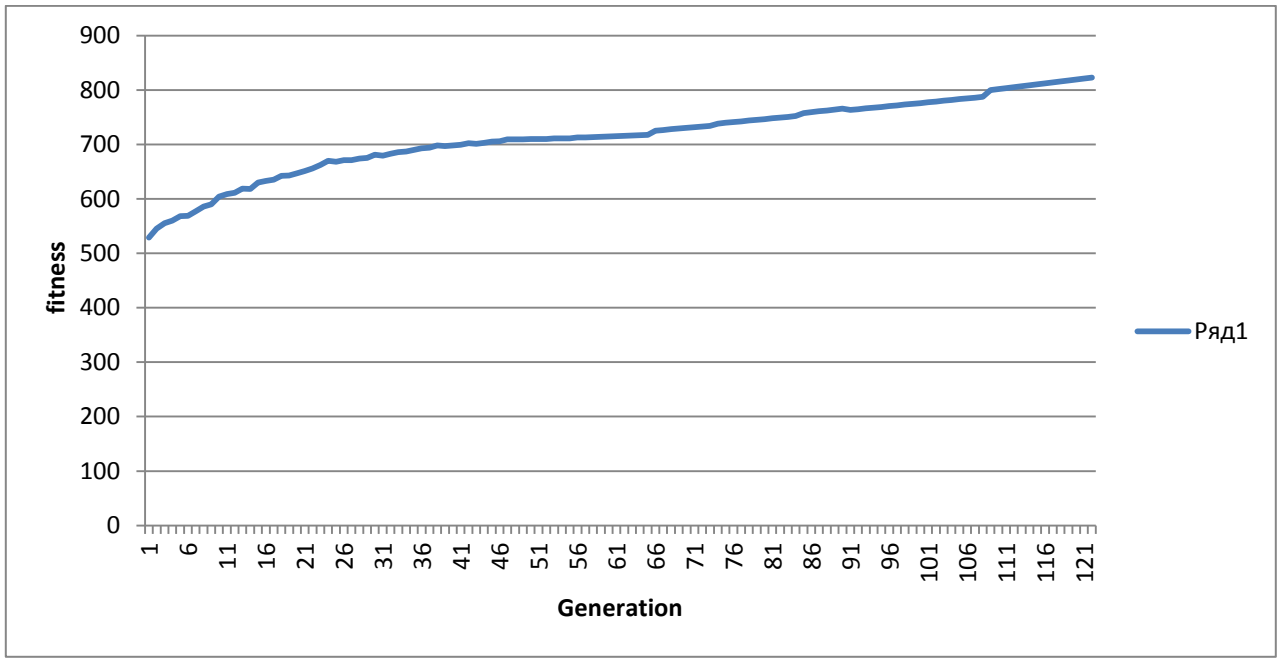


Table of generation condtion

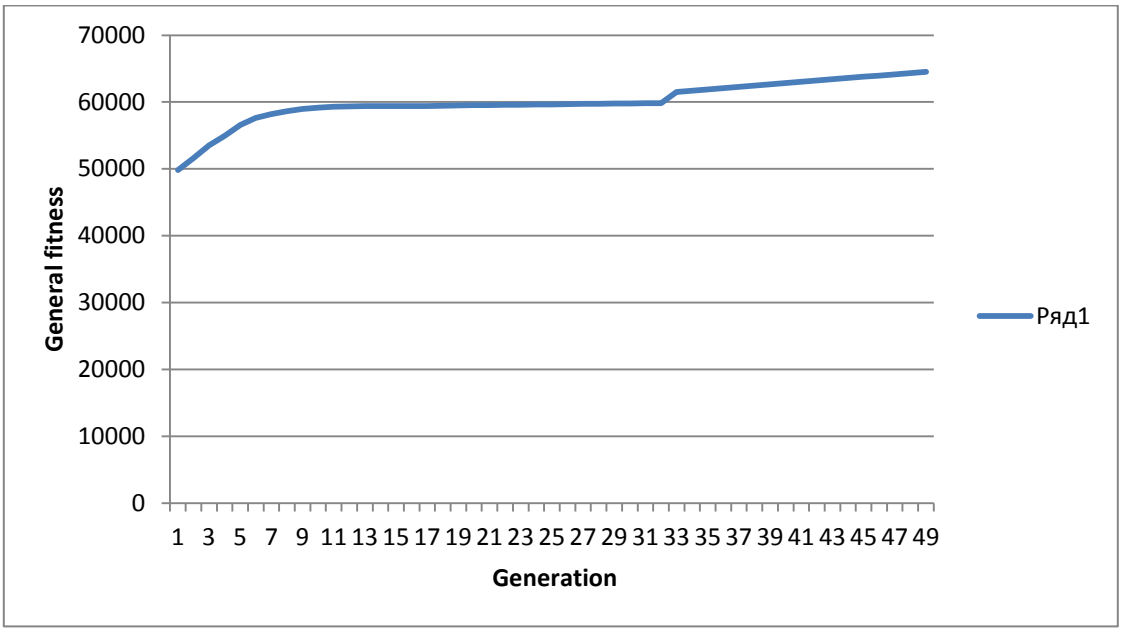
fitness	amaund of use	fitness	amaund of use	fitness	amaund of use
545	1	651	4	870	1
544	2	650	5	869	2
538	3	649	3	868	3
536	2	648	1	867	2
535	5	648	3	867	5
534	2	647	3	866	2
534	2	647	4	866	2
534	2	647	4	866	2
533	2	647	1	866	2
533	1	646	1	865	4
531	1	646	1	865	5
530	0	646	1	865	3
530	0	646	2	865	1
528	1	645	3	864	3
527	2	645	2	864	3
525	1	645	5	864	4
525	3	644	2	863	4
525	4	644	2	863	1
525	0	644	2	863	1
524	4	643	2	862	1
523	2	643	2	862	1
522	2	643	1	862	2
522	1	643	0	862	3
521	0	643	0	862	2
519	1	643	2	862	5
518	0	643	2	862	0
518	1	643	1	862	1
517	1	643	5	862	1
517	3	643	0	862	3
517	3	642	0	861	3
516	3	642	4	861	3
516	1	642	2	861	1
516	2	642	2	861	2
516	2	642	1	861	2
516	4	642	0	861	4
515	5	642	1	861	0
514	1	642	1	861	1
513	1	642	1	861	1
513	1	641	2	860	1
512	2	641	2	860	2
511	1	641	2	860	1

511	6	641	2	860	0
511	5	641	1	860	0
511	1	641	0	860	1
510	1	641	0	860	1
510	1	641	2	860	1
509	2	641	2	860	1
509	2	641	2	860	2
508	4	640	4	859	1
508	2	640	2	859	2
508	1	640	3	859	1
508	0	640	0	859	0
507	0	640	0	859	0
507	0	640	0	859	0
507	0	640	0	859	0
507	0	640	0	859	0
507	0	640	0	859	0
506	0	639	0	858	0
506	0	639	0	858	0
506	0	639	0	858	0
506	0	639	0	858	0
505	0	639	0	858	0
505	0	638	0	857	0
505	0	638	0	857	0
505	0	638	0	857	0
504	0	638	0	857	0
504	0	638	0	857	0
503	0	638	0	857	0
503	0	637	0	856	0
502	0	637	0	856	0
502	0	637	0	856	0
501	0	637	0	856	0
501	0	636	0	855	0
501	0	636	0	855	0
499	0	636	0	855	0
499	0	636	0	855	0
499	0	636	0	855	0
498	0	636	0	855	0
498	0	636	0	855	0
497	0	635	0	854	0
496	0	635	0	854	0
496	0	635	0	854	0
496	0	635	0	854	0
495	0	635	0	854	0

495	0	634	0	853	0
494	0	634	0	853	0
493	0	634	0	853	0
493	0	633	0	852	0
493	0	633	0	852	0
493	0	633	0	852	0
493	0	632	0	851	0
491	0	631	0	850	0
488	0	631	0	850	0
488	0	631	0	850	0
488	0	630	0	849	0
485	0	629	0	848	0
482	0	629	0	848	0
481	0	629	0	848	0
476	0	627	0	846	0
475	0	624	0	843	0

roulette-method

Graph 2



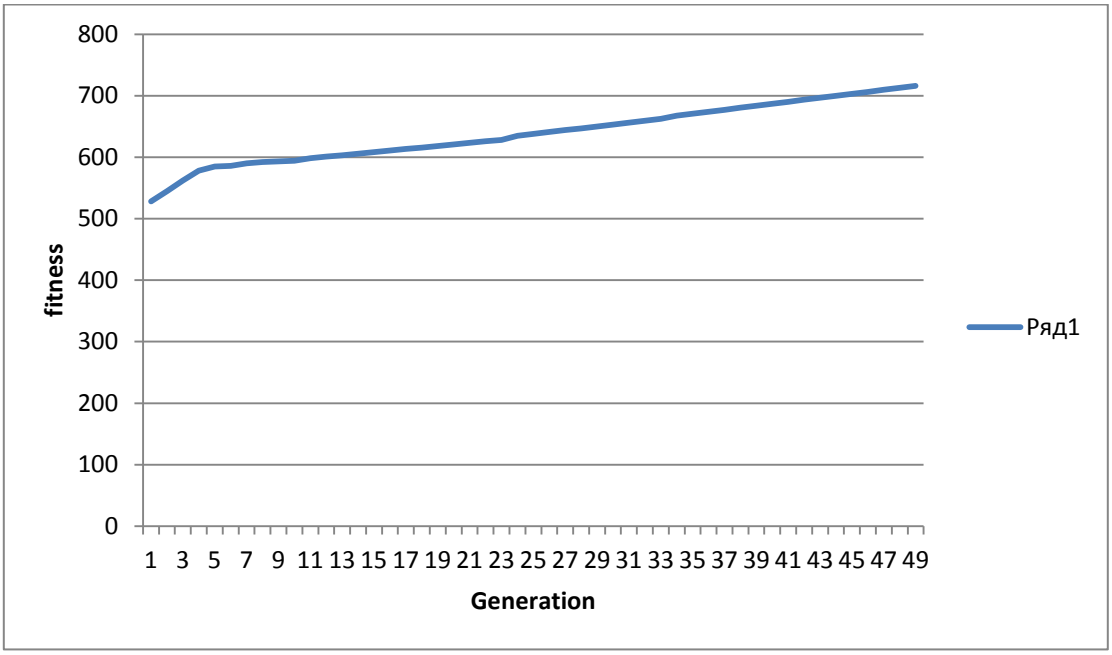


Table of generation condtion

fitness	Propability	amaund of use	fitness	Propability	amaund of use	fitness	Propability	amaund of use
545	0,01094114	7	632	0,010608	8	735	0,01139	15
544	0,01092106	6	631	0,010591	12	734	0,011374	12
543	0,01090099	3	630	0,010574	7	733	0,011359	10
543	0,01090099	5	630	0,010574	4	733	0,011359	7
542	0,01088091	5	629	0,010558	7	732	0,011343	9
542	0,01088091	4	629	0,010558	6	732	0,011343	7
540	0,01084076	5	627	0,010524	3	730	0,011312	6
540	0,01084076	5	627	0,010524	5	730	0,011312	5
539	0,01082069	5	626	0,010507	5	729	0,011297	5
537	0,01078053	4	624	0,010474	4	727	0,011266	5
536	0,01076046	5	623	0,010457	5	726	0,01125	0
535	0,01074038	4	622	0,01044	5	725	0,011235	0
530	0,01064001	2	617	0,010356	5	720	0,011157	1
529	0,01061993	2	616	0,010339	4	719	0,011142	1
529	0,01061993	2	616	0,010339	5	719	0,011142	4
529	0,01061993	2	616	0,010339	4	719	0,011142	1
527	0,01057978	0	614	0,010306	2	717	0,011111	1
527	0,01057978	2	614	0,010306	2	717	0,011111	2
527	0,01057978	2	614	0,010306	2	717	0,011111	2
526	0,0105597	1	613	0,010289	2	716	0,011095	2
526	0,0105597	2	613	0,010289	0	716	0,011095	2
525	0,01053963	2	612	0,010272	2	715	0,01108	0
525	0,01053963	2	612	0,010272	1	715	0,01108	2
525	0,01053963	1	612	0,010272	0	715	0,01108	1
525	0,01053963	1	612	0,010272	0	715	0,01108	0
523	0,01049948	1	610	0,010239	0	713	0,011049	0
523	0,01049948	1	610	0,010239	0	713	0,011049	0
522	0,0104794	1	609	0,010222	0	712	0,011033	0
521	0,01045933	1	608	0,010205	0	711	0,011018	0
521	0,01045933	1	608	0,010205	0	711	0,011018	0
521	0,01045933	1	608	0,010205	0	711	0,011018	0
521	0,01045933	0	608	0,010205	0	711	0,011018	0
520	0,01043925	0	607	0,010188	0	710	0,011002	0
520	0,01043925	1	607	0,010188	0	710	0,011002	0
520	0,01043925	2	607	0,010188	0	710	0,011002	0
520	0,01043925	1	607	0,010188	0	710	0,011002	0
520	0,01043925	0	607	0,010188	0	710	0,011002	0
519	0,01041918	0	606	0,010172	0	709	0,010987	0
519	0,01041918	0	606	0,010172	0	709	0,010987	0
519	0,01041918	0	606	0,010172	0	709	0,010987	0
519	0,01041918	0	606	0,010172	0	709	0,010987	0
519	0,01041918	0	606	0,010172	0	709	0,010987	0
518	0,0103991	0	605	0,010155	0	708	0,010971	0
518	0,0103991	0	605	0,010155	0	708	0,010971	0
518	0,0103991	0	605	0,010155	0	708	0,010971	0

518	0,0103991	0	605	0,010155	0	708	0,010971	0
517	0,01037903	1	604	0,010138	0	707	0,010956	0
516	0,01035895	1	603	0,010121	0	706	0,01094	0
515	0,01033887	1	602	0,010104	0	705	0,010925	0
515	0,01033887	1	602	0,010104	0	705	0,010925	0
514	0,0103188	0	601	0,010088	0	704	0,010909	0
514	0,0103188	0	601	0,010088	0	704	0,010909	0
514	0,0103188	0	601	0,010088	0	704	0,010909	0
513	0,01029872	0	600	0,010071	0	703	0,010894	0
513	0,01029872	0	600	0,010071	0	703	0,010894	0
513	0,01029872	0	600	0,010071	0	703	0,010894	0
513	0,01029872	0	600	0,010071	0	703	0,010894	0
513	0,01029872	0	600	0,010071	0	703	0,010894	0
512	0,01027865	0	599	0,010054	0	702	0,010878	0
512	0,01027865	0	599	0,010054	0	702	0,010878	0
512	0,01027865	0	599	0,010054	0	702	0,010878	0
511	0,01025857	0	598	0,010037	0	701	0,010863	0
511	0,01025857	0	598	0,010037	0	701	0,010863	0
510	0,0102385	1	597	0,01002	0	700	0,010847	0
510	0,0102385	0	597	0,01002	0	700	0,010847	0
510	0,0102385	1	597	0,01002	0	700	0,010847	0
510	0,0102385	1	597	0,01002	0	700	0,010847	0
509	0,01021842	1	596	0,010004	0	699	0,010832	0
509	0,01021842	1	596	0,010004	0	699	0,010832	0
509	0,01021842	1	596	0,010004	0	699	0,010832	0
508	0,01019835	1	595	0,009987	0	698	0,010816	0
508	0,01019835	0	595	0,009987	0	698	0,010816	0
508	0,01019835	0	595	0,009987	0	698	0,010816	0
507	0,01017827	0	594	0,00997	0	697	0,010801	0
507	0,01017827	0	594	0,00997	0	697	0,010801	0
506	0,01015819	0	593	0,009953	0	696	0,010785	0
506	0,01015819	0	593	0,009953	0	696	0,010785	0
505	0,01013812	0	592	0,009937	0	695	0,01077	0
505	0,01013812	0	592	0,009937	0	695	0,01077	0
505	0,01013812	0	592	0,009937	0	695	0,01077	0
504	0,01011804	0	591	0,00992	0	694	0,010754	0
504	0,01011804	0	591	0,00992	0	694	0,010754	0
503	0,01009797	0	590	0,009903	0	693	0,010739	0
503	0,01009797	0	590	0,009903	0	693	0,010739	0
502	0,01007789	0	589	0,009886	0	692	0,010723	0
502	0,01007789	0	589	0,009886	0	692	0,010723	0
502	0,01007789	0	589	0,009886	0	692	0,010723	0
501	0,01005782	0	588	0,009869	0	691	0,010708	0

500	0,01003774	0	587	0,009853	0	690	0,010692	0
500	0,01003774	0	587	0,009853	0	690	0,010692	0
498	0,00999759	0	585	0,009819	0	688	0,010661	0
498	0,00999759	0	585	0,009819	0	688	0,010661	0
498	0,00999759	0	585	0,009819	0	688	0,010661	0
495	0,00993736	0	582	0,009769	0	685	0,010615	0
495	0,00993736	0	582	0,009769	0	685	0,010615	0
494	0,00991729	0	581	0,009752	0	684	0,010599	0
494	0,00991729	0	581	0,009752	0	684	0,010599	0
489	0,00981691	0	576	0,009668	0	679	0,010522	0
489	0,00981691	0	576	0,009668	0	679	0,010522	0
488	0,00979684	0	575	0,009651	0	678	0,010506	0