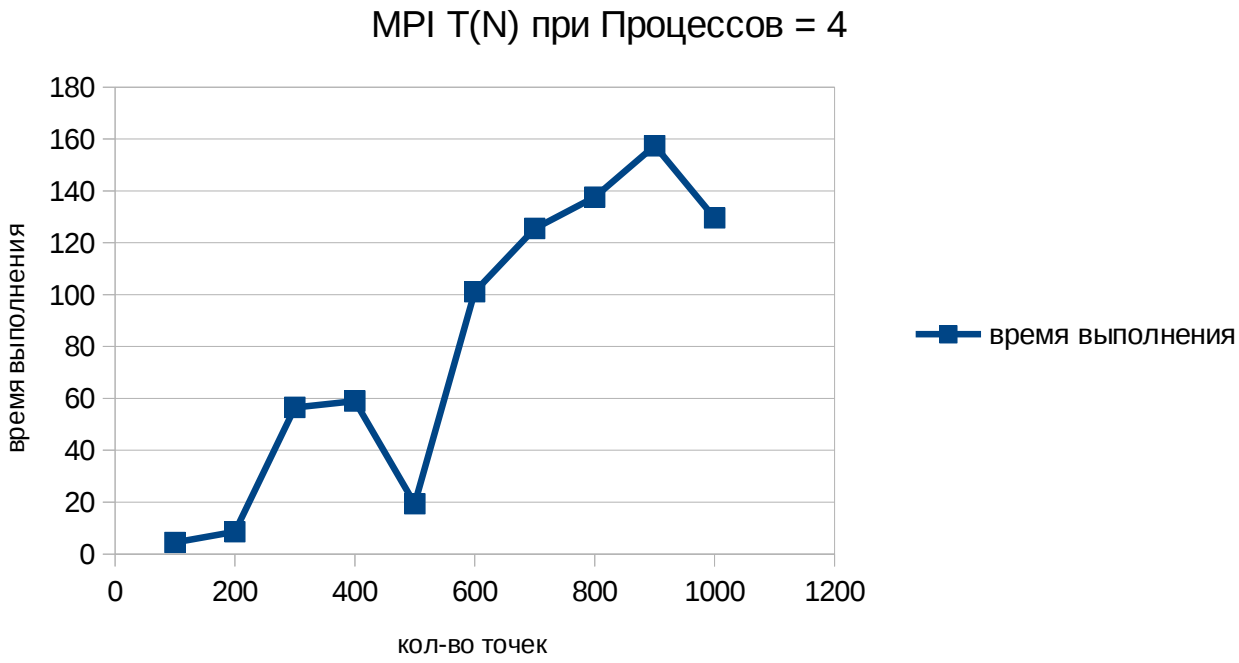
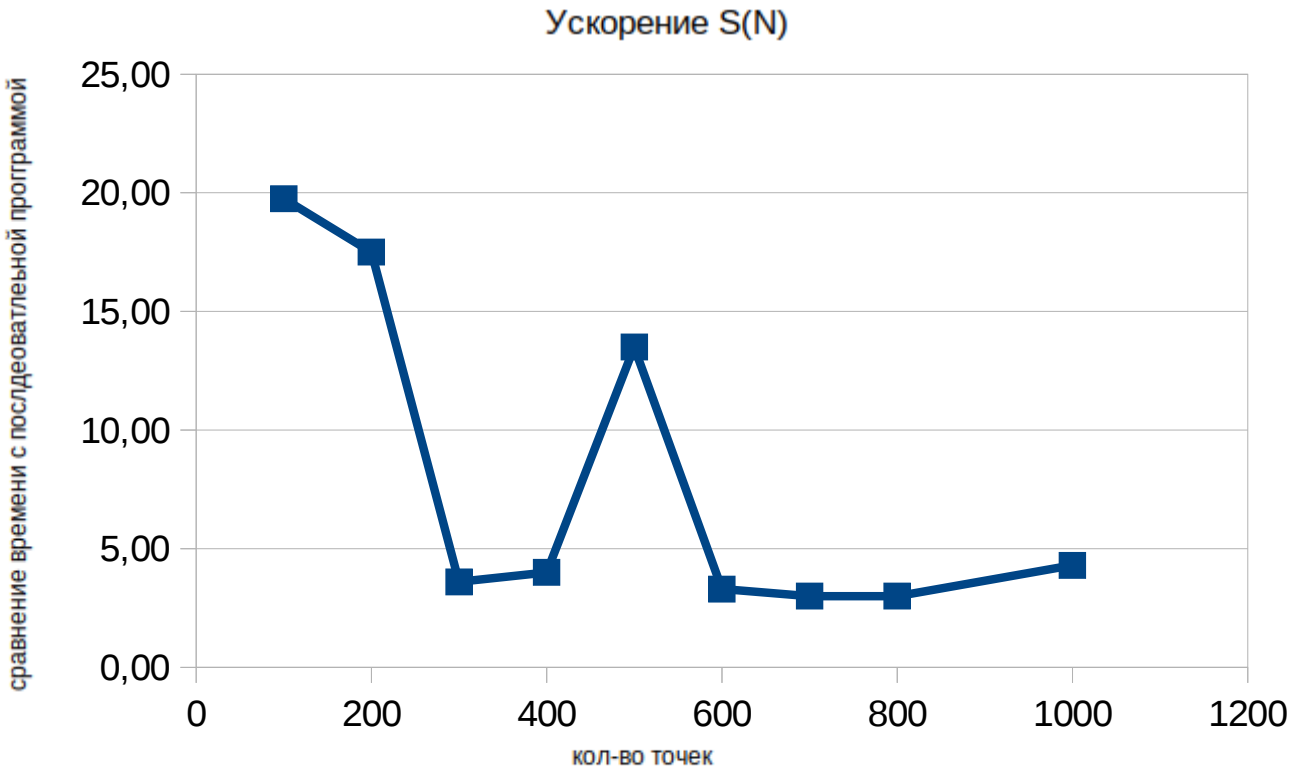
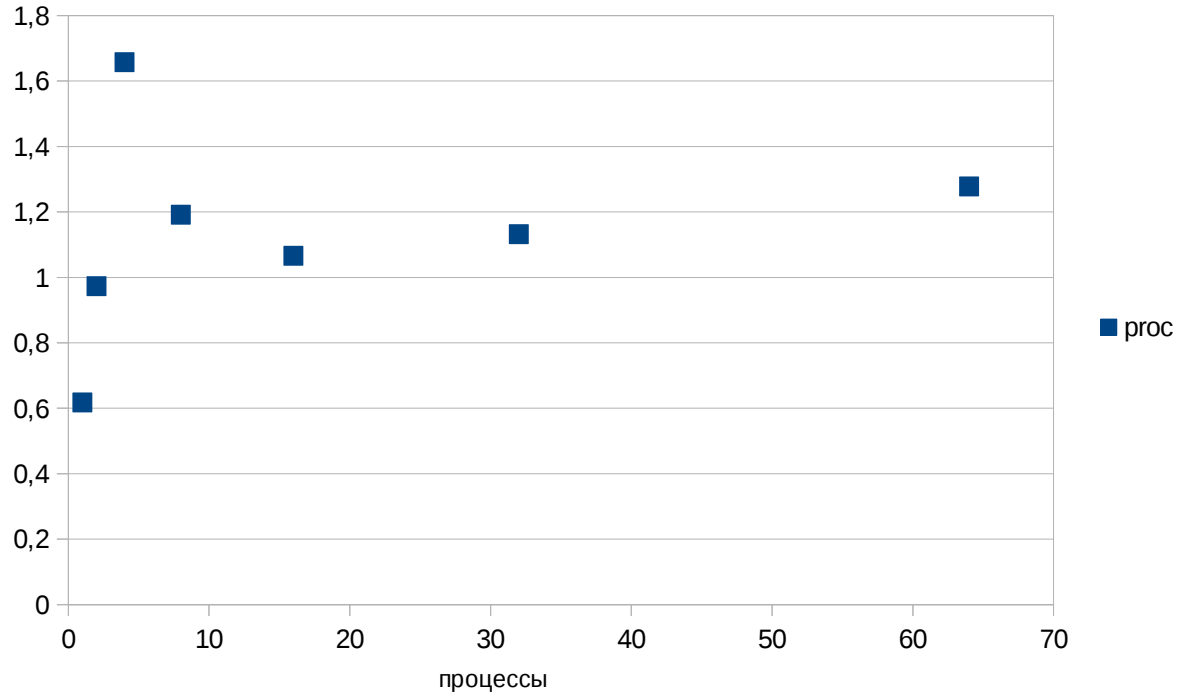
Женин Максим Николаевич, maximham@mail.ru
Задание 1 MPI: методы Монте-Карло

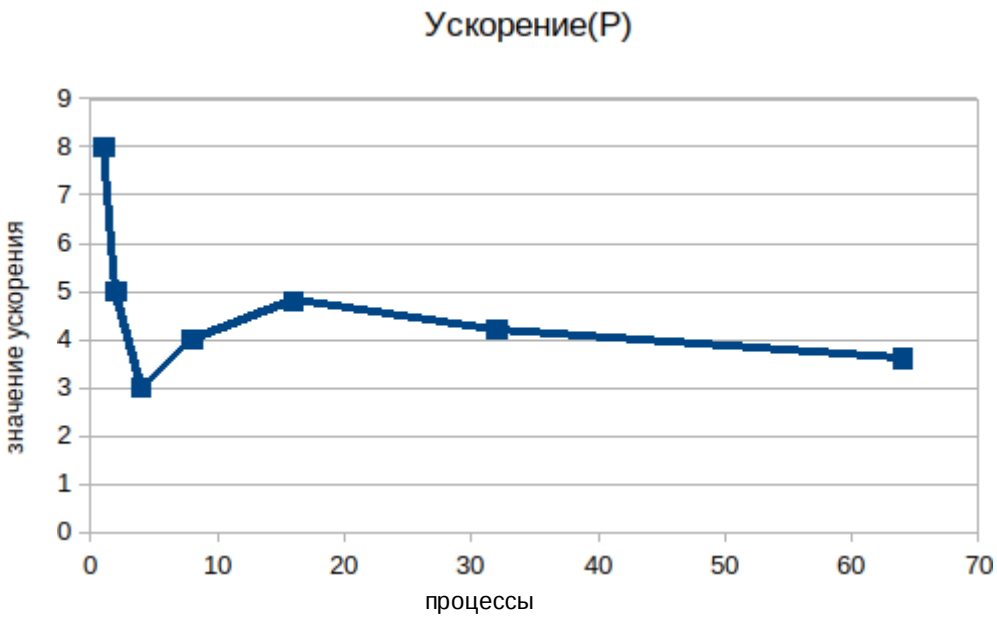
T(N): a = 4,b = 600, x = 50, p =0.5 , N = 100 от 1000, P = 4
Тут время выполнения умножил на 1000, чтобы нормально отображалось!



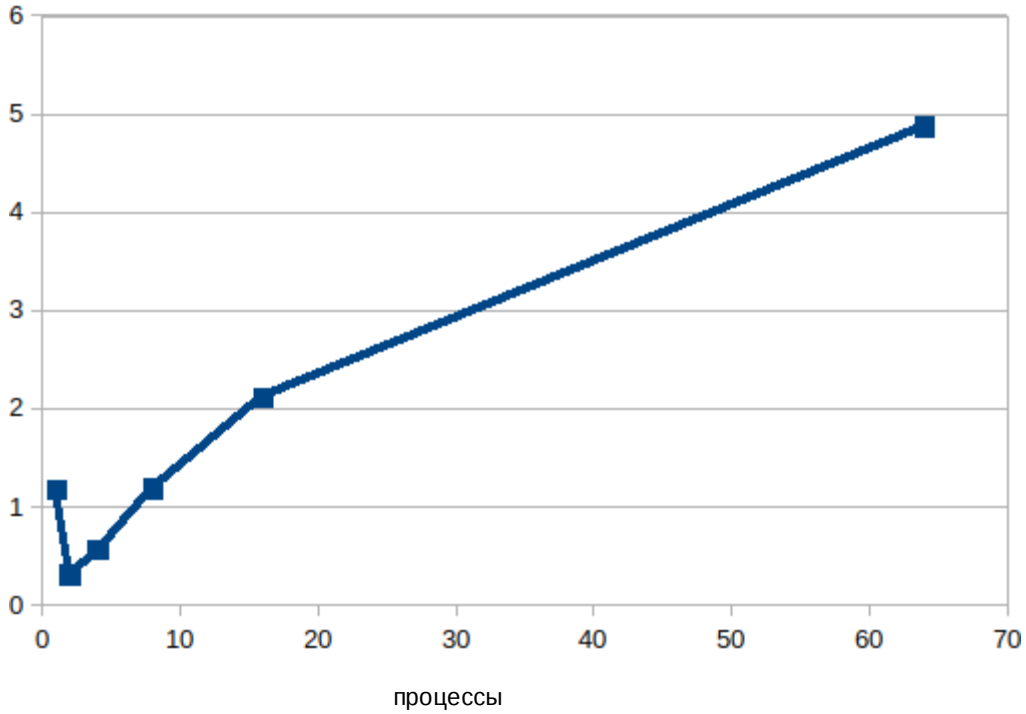


фиксированном значении N = 10000
Кол-во процессов от 1 — 64;
T(P)

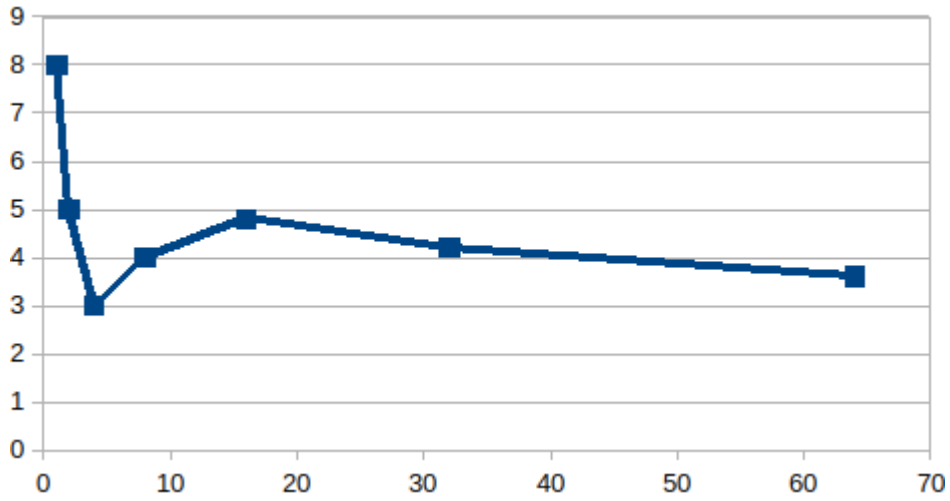




T(Proc * 1000)



S(Proc * 1000)



```
#include <iostream>
#include <mpi.h>
#include <fstream>
#include <cstdlib>
#include <cmath>
#include <ctime>

int Nprocs = 1;
int Rank;


using namespace std;


double frand(double a, double b)
{
	//double z =  a+(b-a)*(rand()/double(RAND_MAX));
	return a+(b-a)*(rand()/double(RAND_MAX));
}

int do_walk(int a, int b, int x, double p, double& t)
{
	int step = 0;
	while( x>a && x<b )
	{
		if( frand(0,1)<p )
			x += 1;
		else
			x -= 1;
		t += 1.0;
		step += 1;
	}
	return x;
}

void run_mc(int a, int b, int x, double p, int N)
{
	// srand(time(0));
	double t = 0.0;
	double w = 0.0;
	double ws = 0.0;
	double ts = 0.0;
	for( int i=0; i<N; i++ )
	{
		int out = do_walk(a, b, x, p, t);
```

```
            if( out == b )
                w += 1;
        }
        cout << "t == " << t << endl;
        cout << "w == " << w << endl;

MPI_Reduce(&w,&ws,1,MPI_DOUBLE_PRECISION,MPI_SUM,0,MPI_COMM_
WORLD);

MPI_Reduce(&t,&ts,1,MPI_DOUBLE_PRECISION,MPI_SUM,0,MPI_COMM_W
ORLD);
        if (!Rank) {
            ofstream f("output.txt");
            f << ws/N << " " << t/N << endl;
            f.close();
        }
}

int main(int argc, char** argv)
{
        if (MPI_Init(&argc, &argv) != MPI_SUCCESS) {
            fprintf(stderr, "failed to init MPI\n");
            exit(1);
        }
        if (MPI_Comm_size(MPI_COMM_WORLD, &Nprocs) != MPI_SUCCESS ||
MPI_Comm_rank(MPI_COMM_WORLD, &Rank) != MPI_SUCCESS) {
            fprintf(stderr, "failed to get communicator size or rank\n");
            exit(1);
        }
        srand(time(NULL)+ Rank);
        int a = atoi(argv[1]);
        int b = atoi(argv[2]);
        int x = atoi(argv[3]);
        double p = atof(argv[4]);
        int N = atoi(argv[5]);
        int M = N / Nprocs;
        int zer = N % Nprocs;
        double time = MPI_Wtime();
        double maxtime = 0;
        if (Rank == Nprocs - 1){
            run_mc(a, b, x, p, M + zer);
        }
        else
        {
            run_mc(a,b,x,p,M);
```

```
        }

        time = MPI_Wtime() - time;

MPI_Reduce(&time,&maxtime,1,MPI_DOUBLE_PRECISION,MPI_MAX,0,MPI_
COMM_WORLD);
        if (!Rank) {
                ofstream f1("stat.txt");
                f1 << maxtime << " " << a << " " << b << " " << x << " " << p << " " <<
N << " " << Nprocs << endl;
                f1.close();
        }
        MPI_Finalize();
        return 0;
}
```