**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

Московский институт электроники и математики Национального

исследовательского университета "Высшая школа экономики"

Департамент прикладной математики

**ОТЧЕТ**

**По лабораторной работе №1**

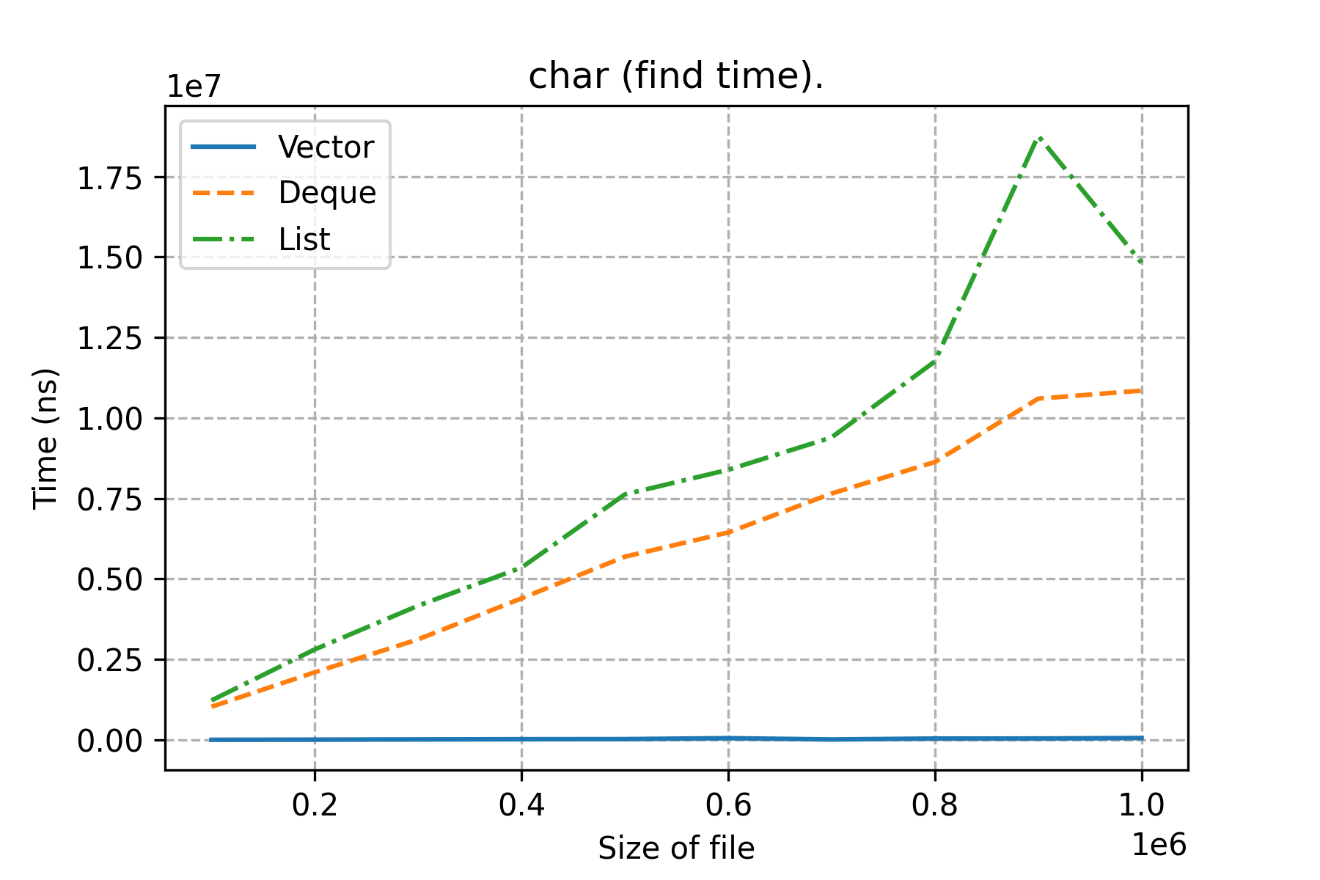
**По курсу «Алгоритмизация и программирование»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | ФИО студента | | Номер группы | Дата |
| Колодин Матвей Алексеевич | БПМ213 | 21.10.22 |
|  |
|  |
|  |

**Москва – 2022 г.**

**Результаты**

1. Давайте сравним алгоритм find на следующих контейнерах: vector, deque и list. Данные будут типа char, а файлы, в которых хранятся данные, разных размеров.

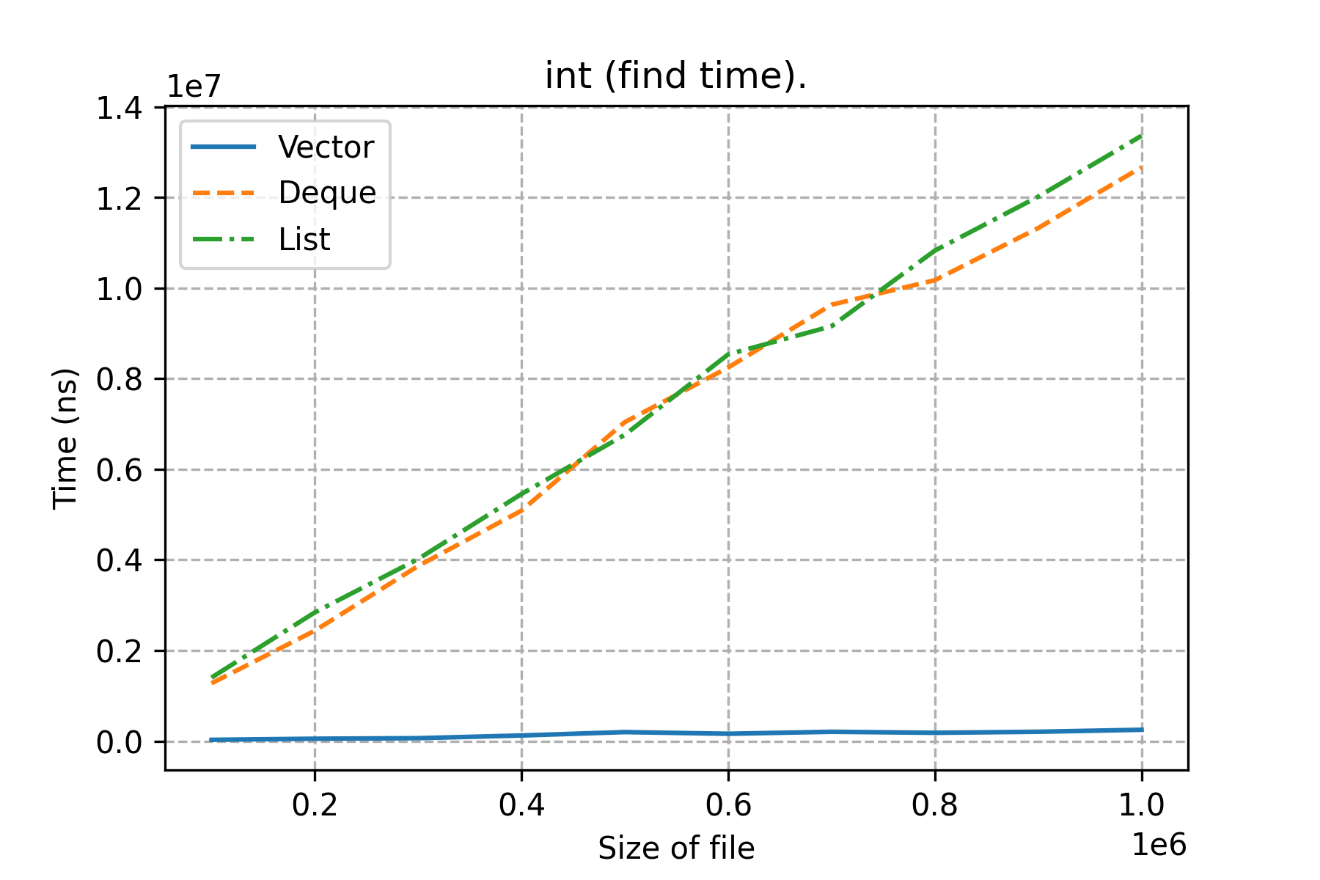


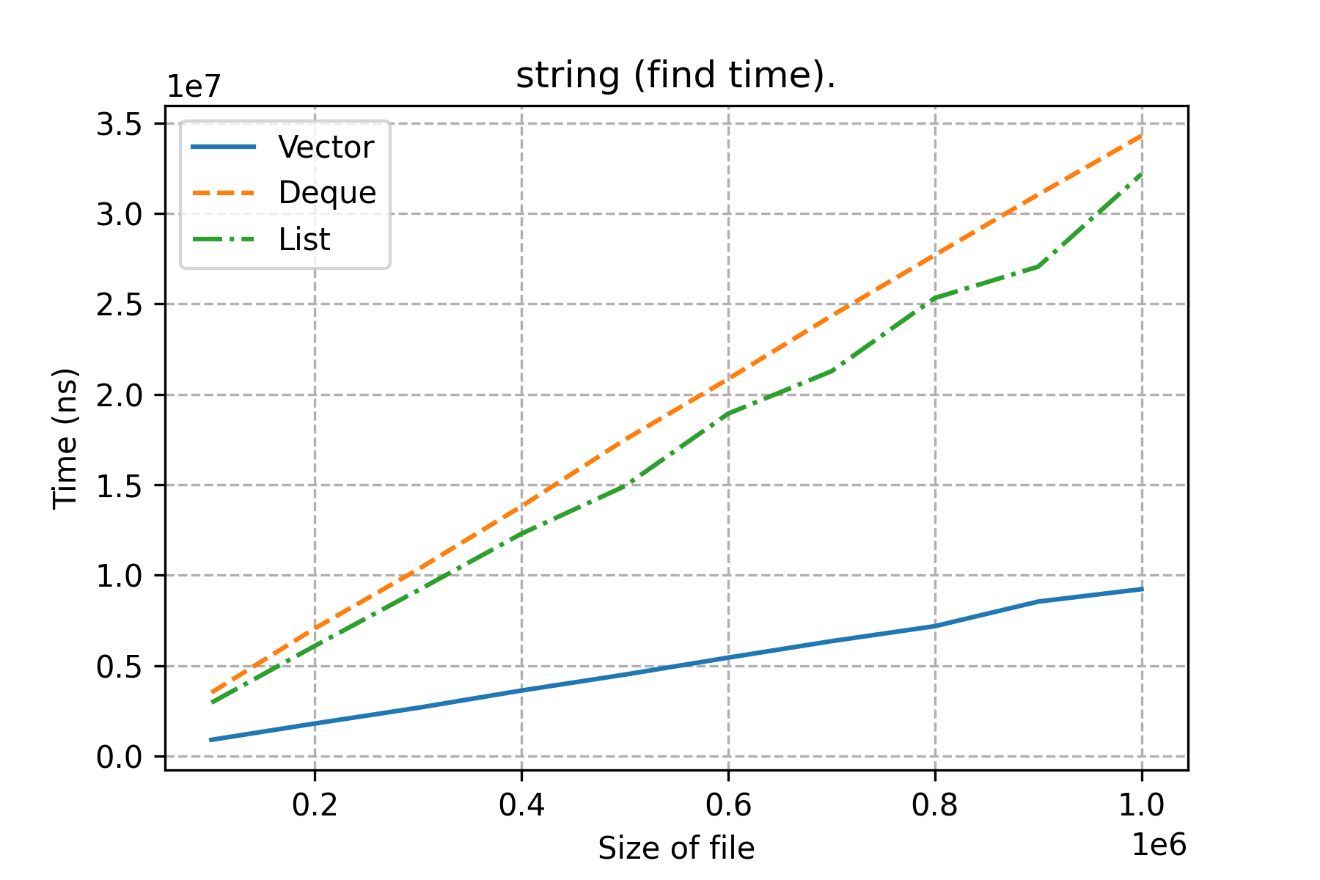
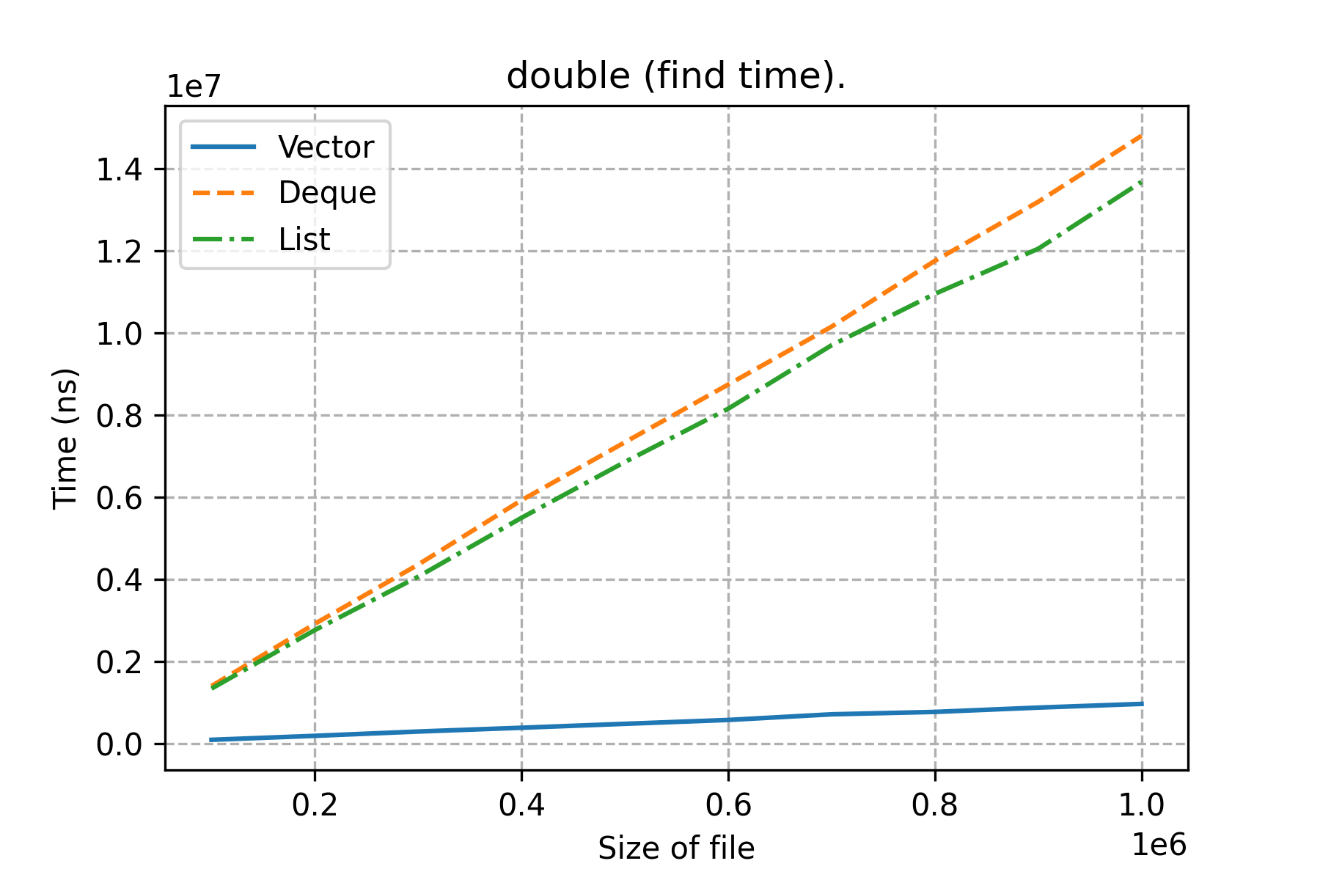
Можно видеть, что наилучшее время у vector, а худшее у list.

Объясняется это их хранением в памяти и тем, как ходят итераторы по их элементам.

Есть некоторая аномалия с листом (возможно какое-то провисание операционной системы)

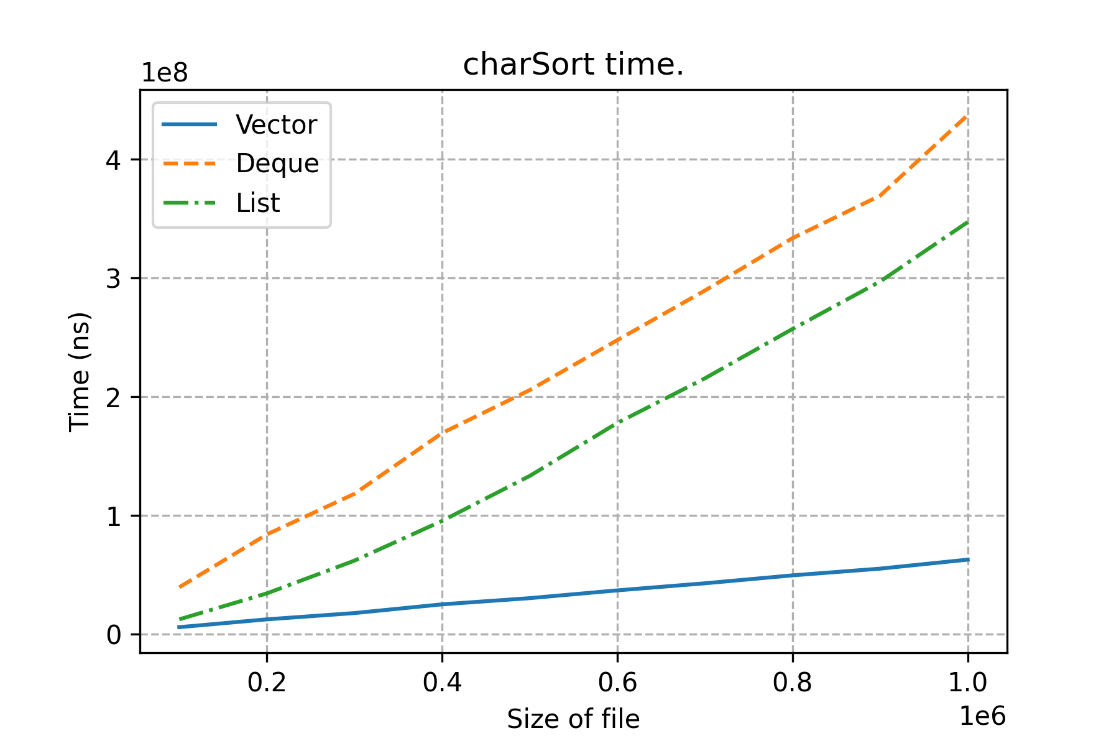
1. Для int мы видим, что Deque и List очень близки по результату, а вот Vector сильно лучше их.

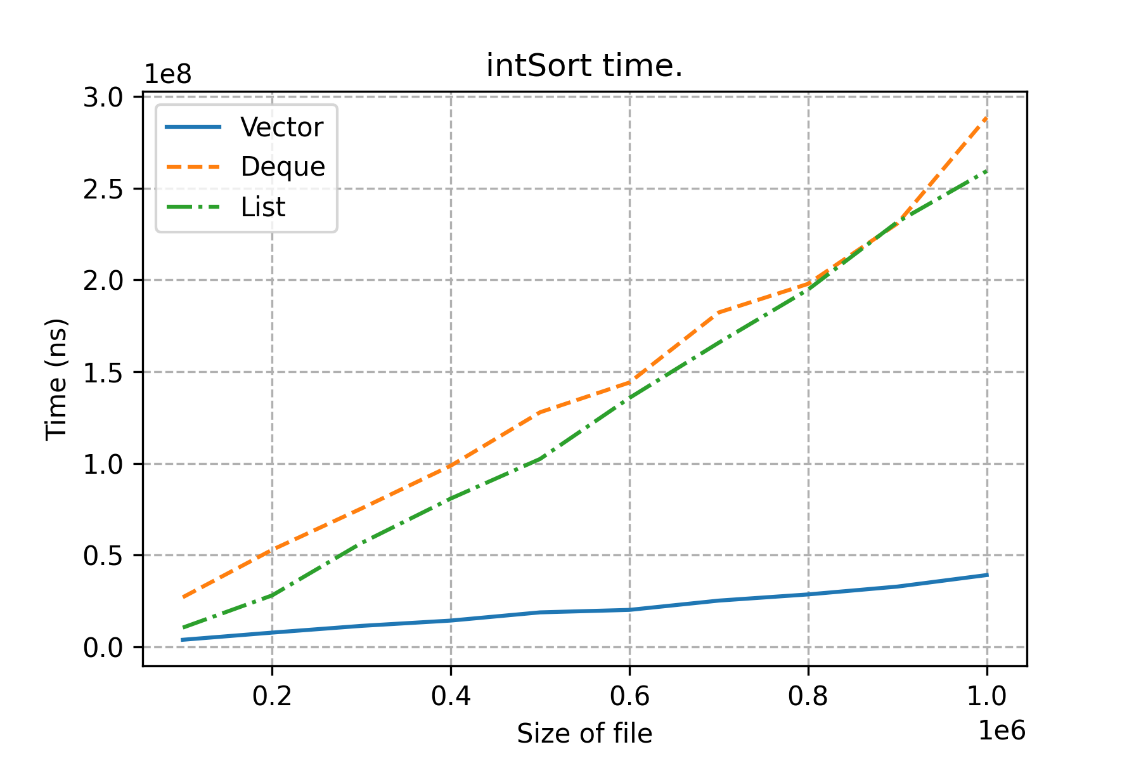


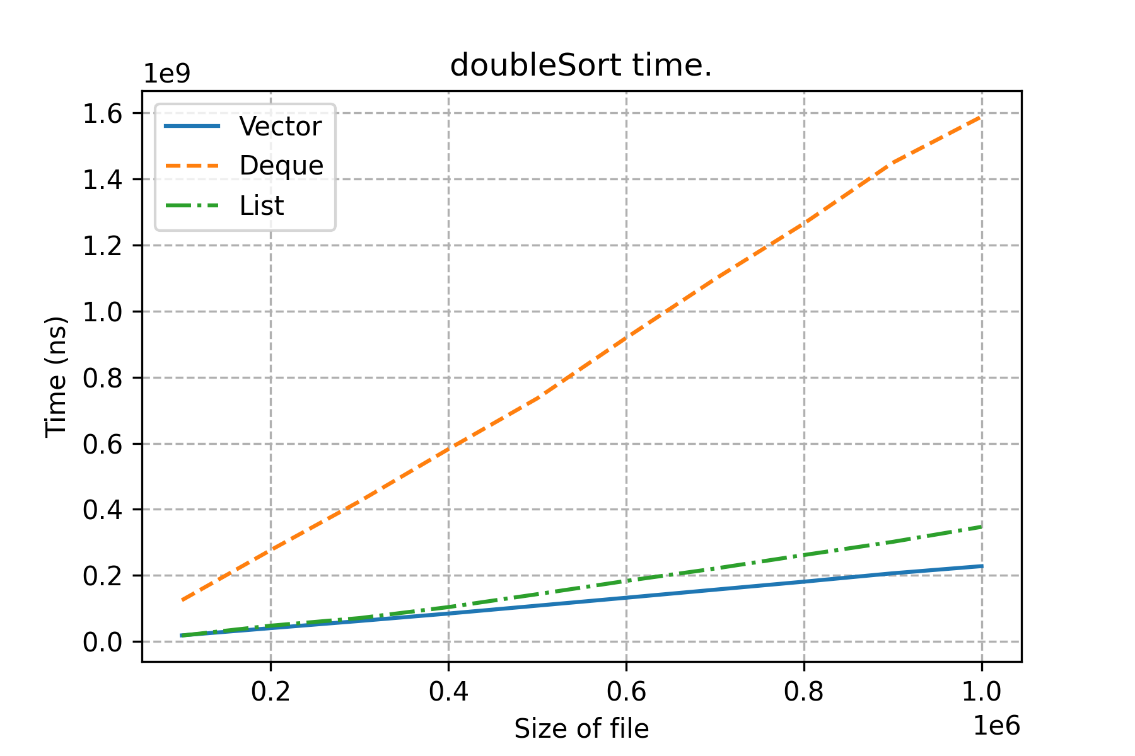


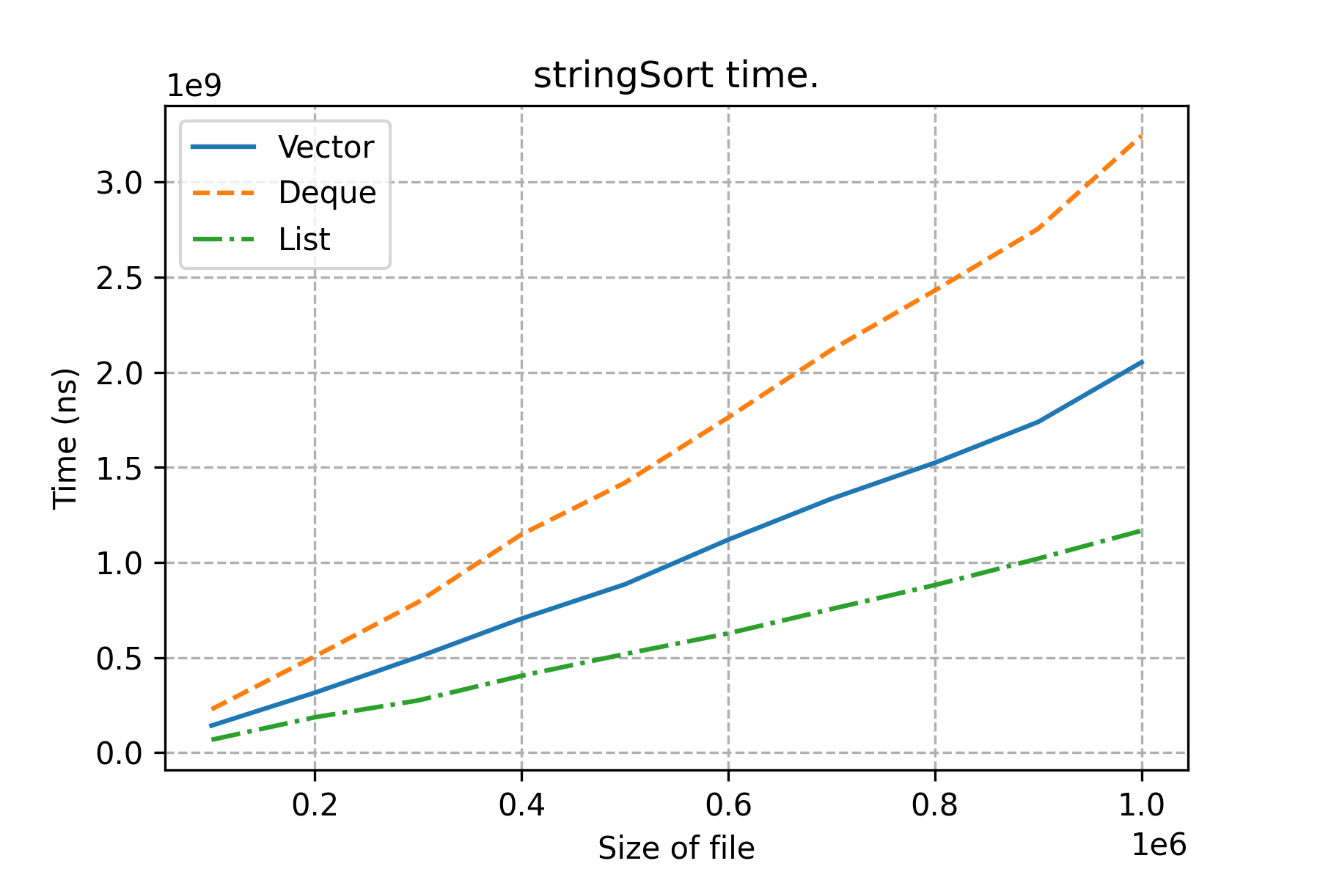
Для double ситуация схожая, а вот для string, Vector начинает тратить больше времени.

1. Переходим к sort – алгоритму.





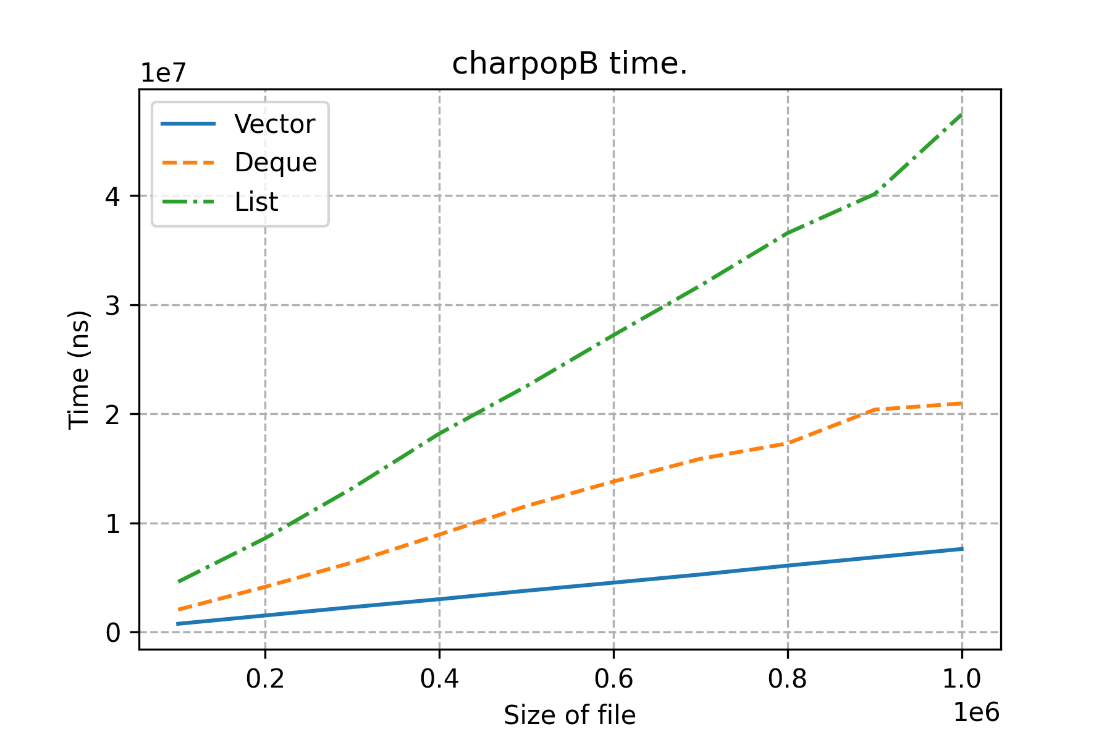


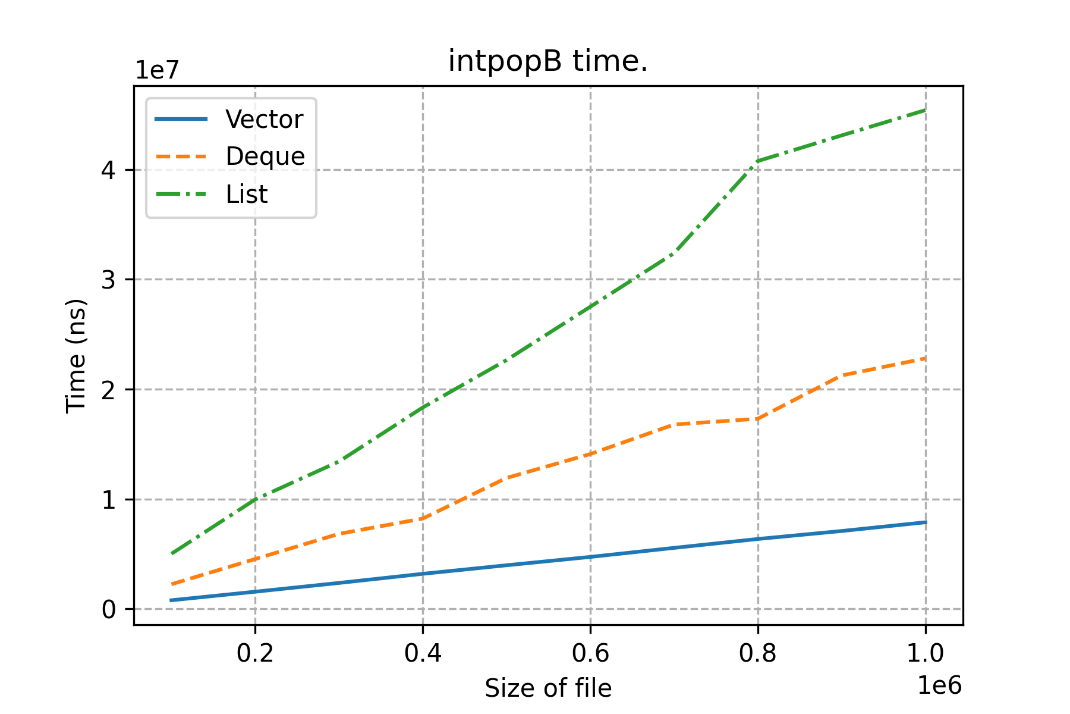


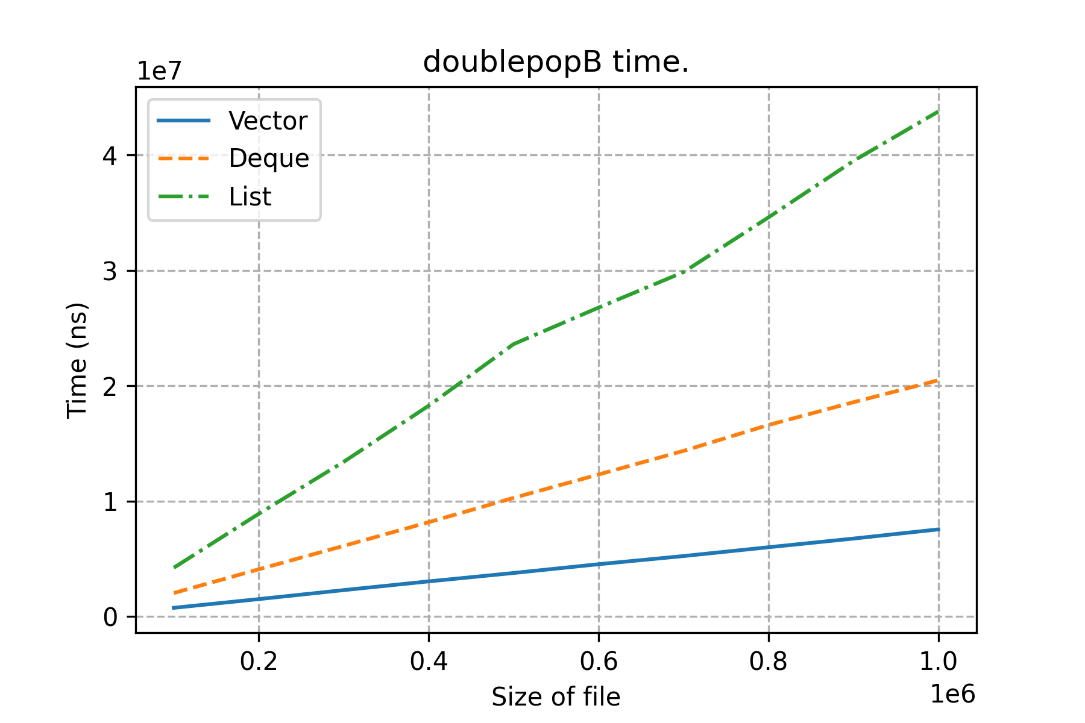
Ситуация, в целом, для char и int схожая, а вот в double к победителям по скорости присоединяется List, а для string и вовсе становится самым быстрым методом.

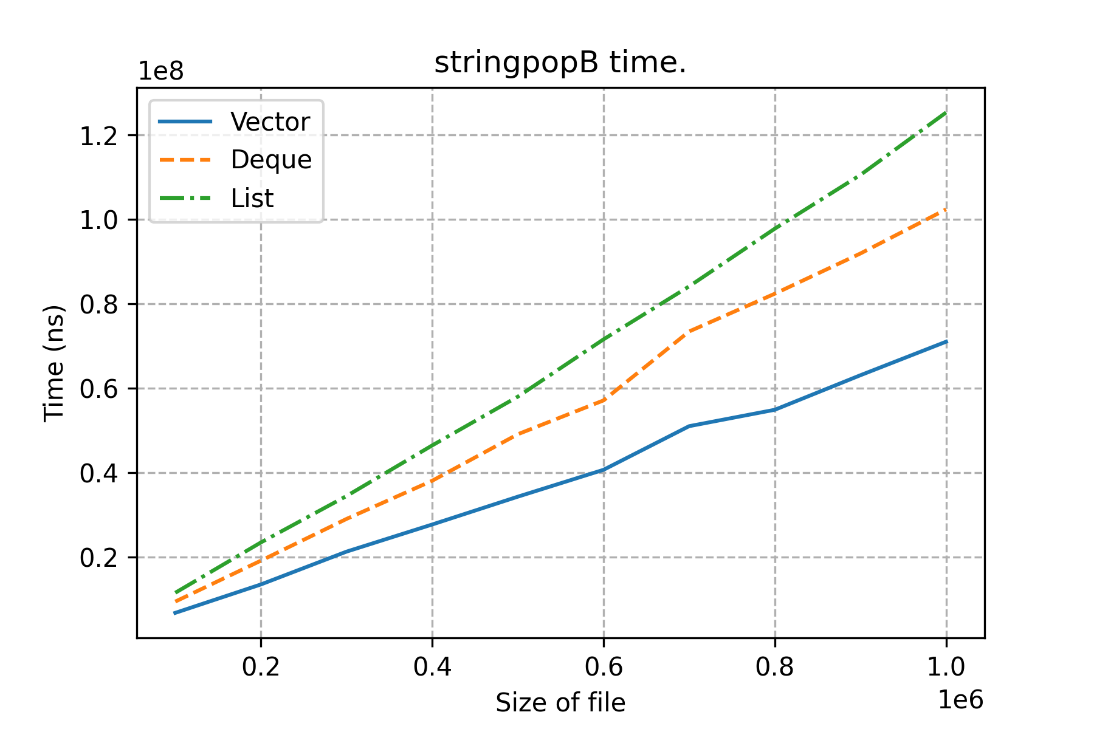
**У List свой метод сортировки, поскольку там не random-access итератор.**

1. Переходим к выкидыванию элементов! Pop\_back



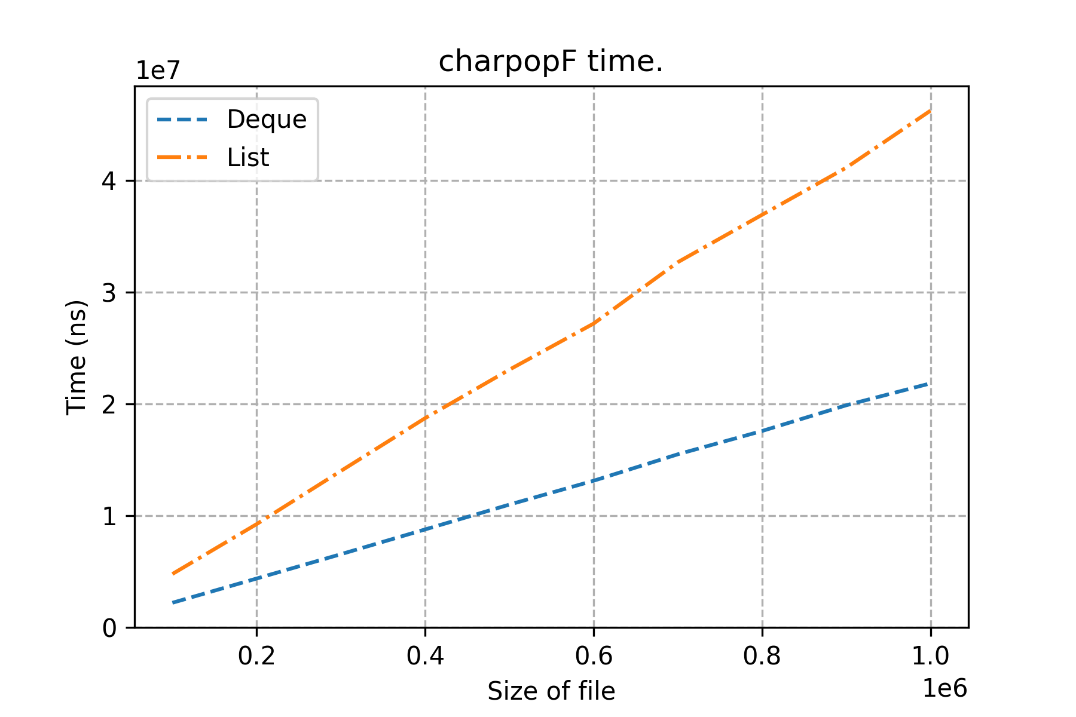


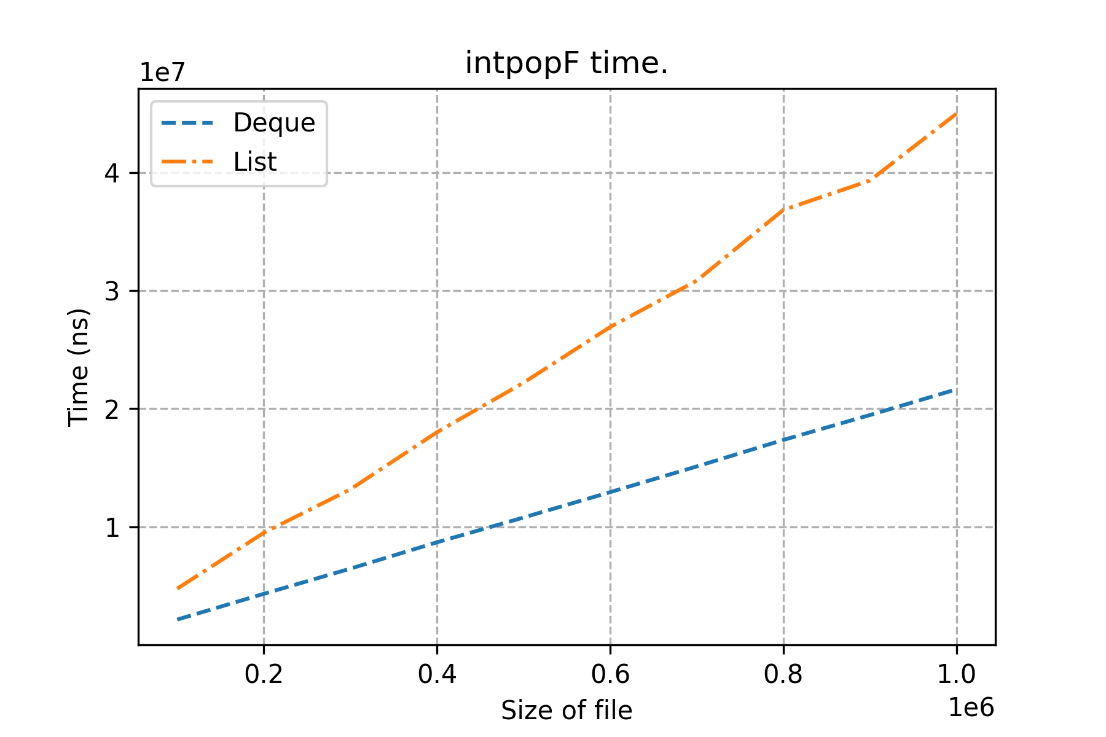


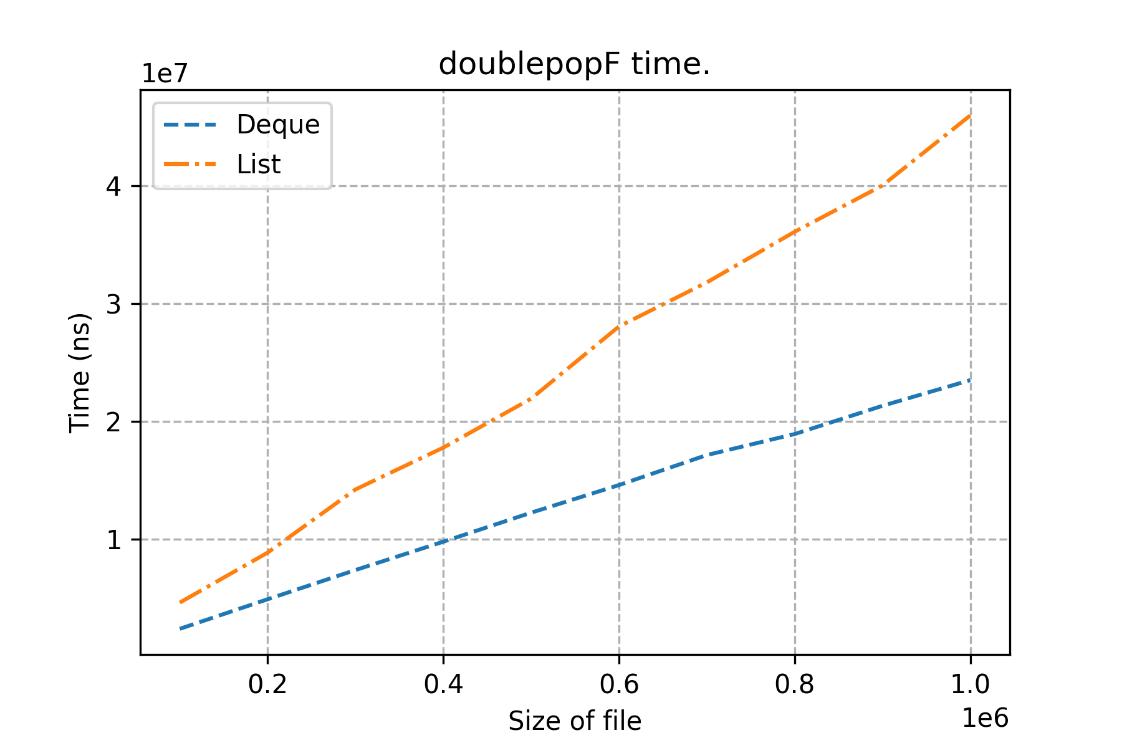


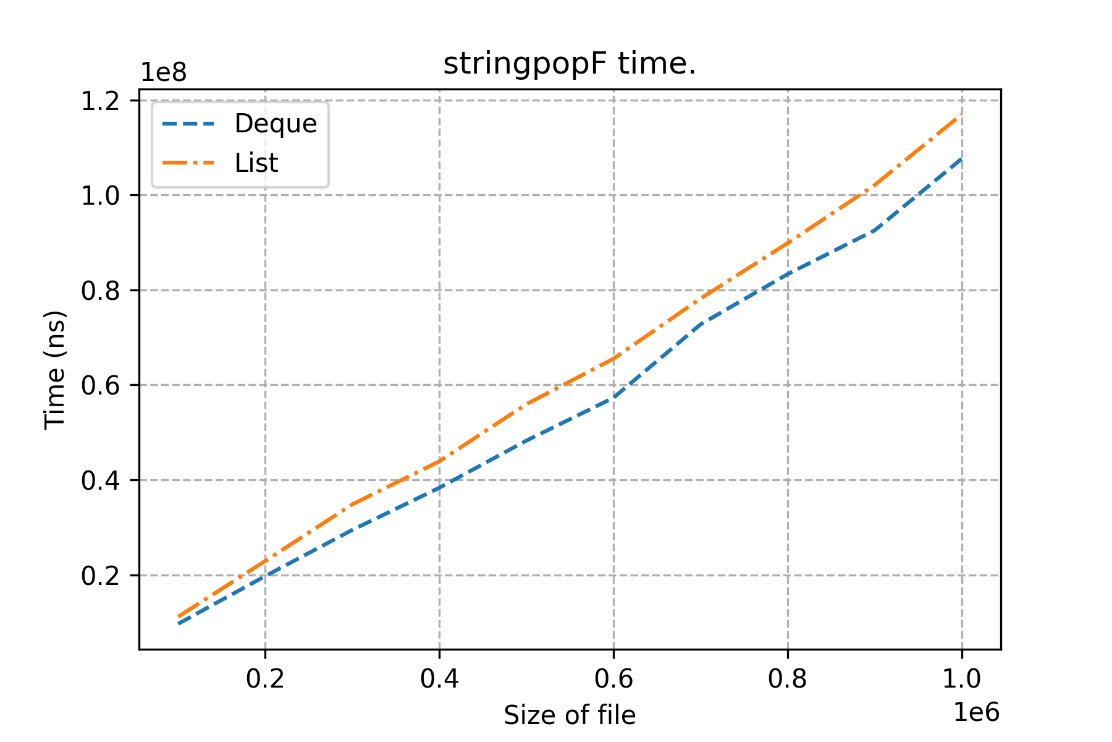
Тут для всех ситуаций тройка по скорости работы определяется однозначно.

1. Теперь pop\_front!





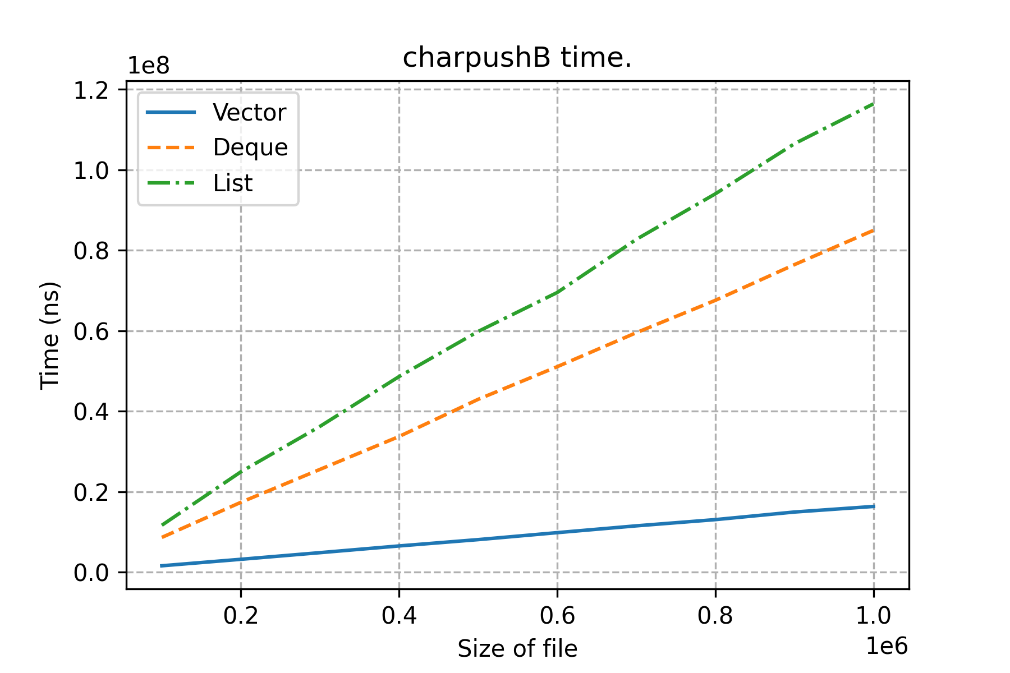


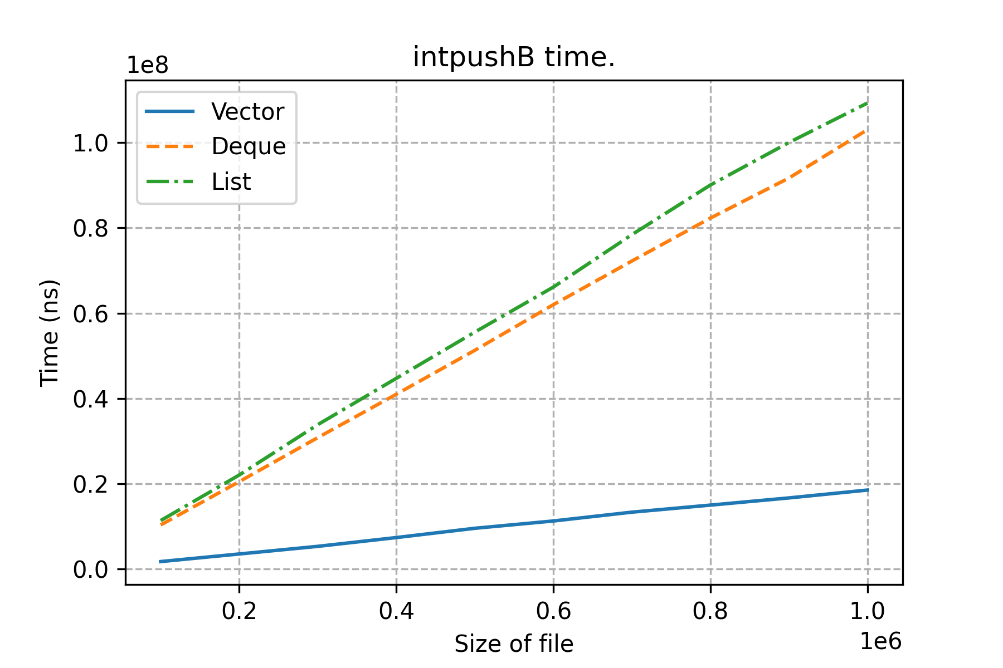


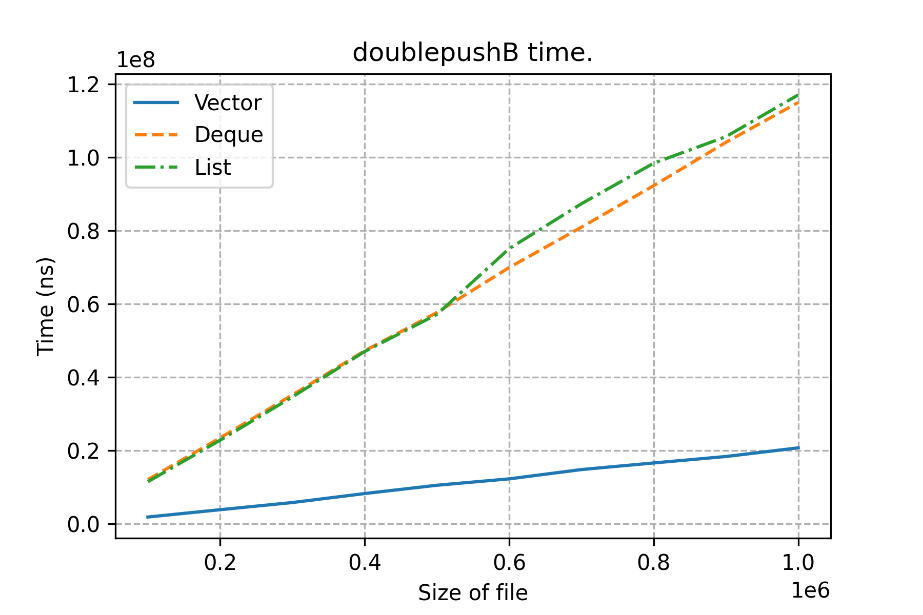
Контейнера vector нет, поскольку в нем не реализован метод pop\_front (но костыльный вариант есть в коде). Работает долго(по расчетам около 6 часов уйдет)., смысла смотреть нет

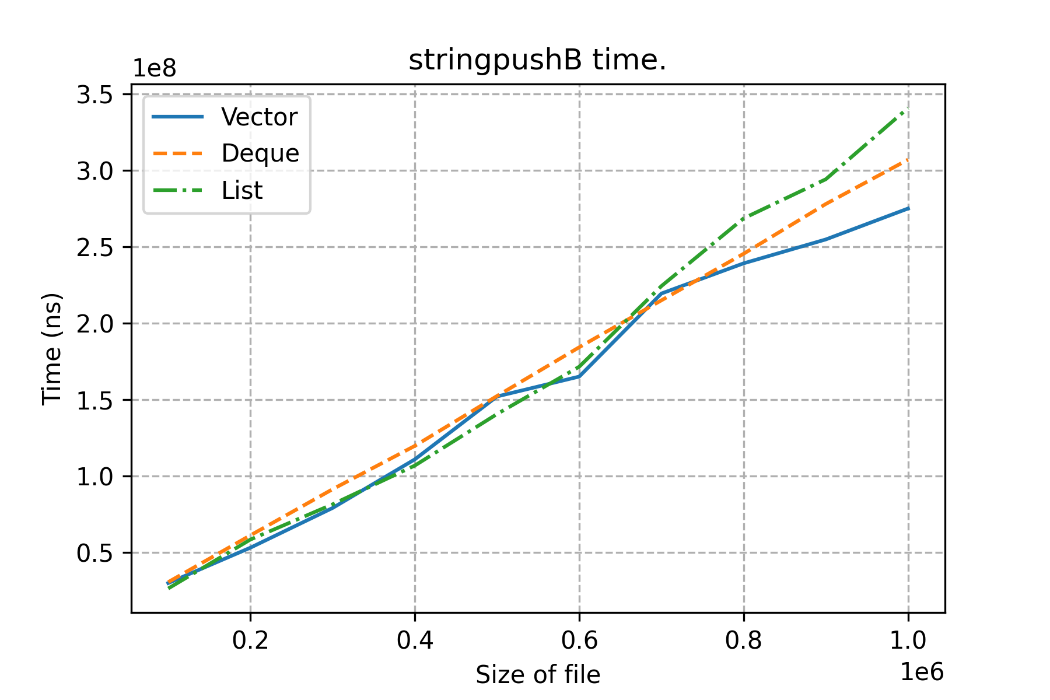
Тут по скорости все также однозначно!

1. Переходим к последним функциям – push! На очереди pushB!



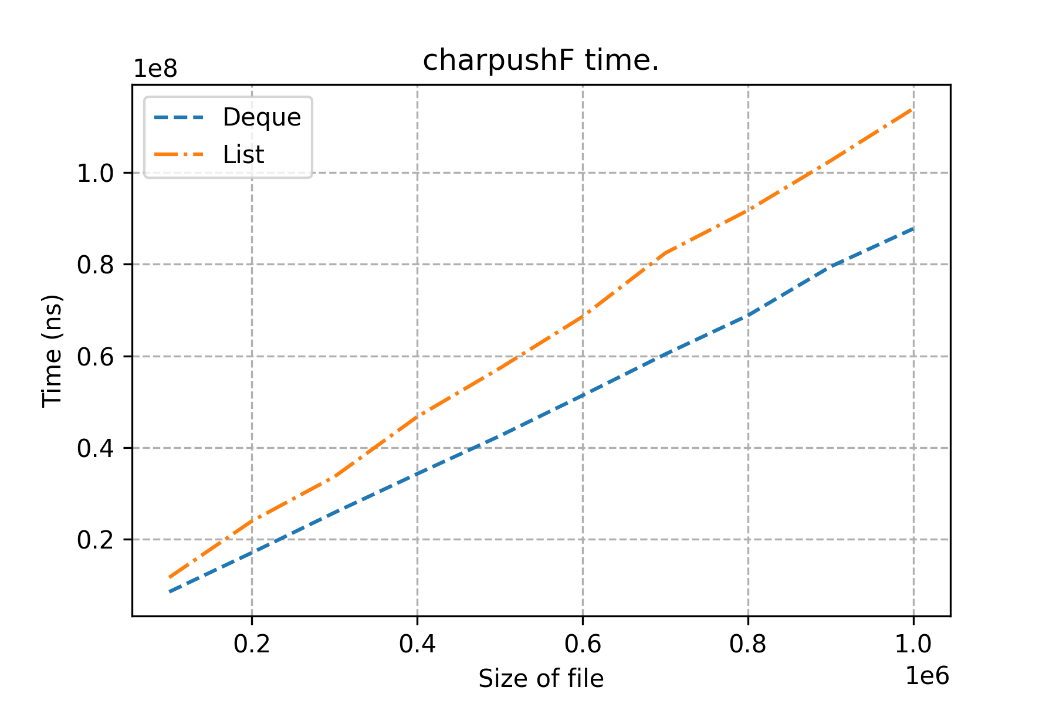


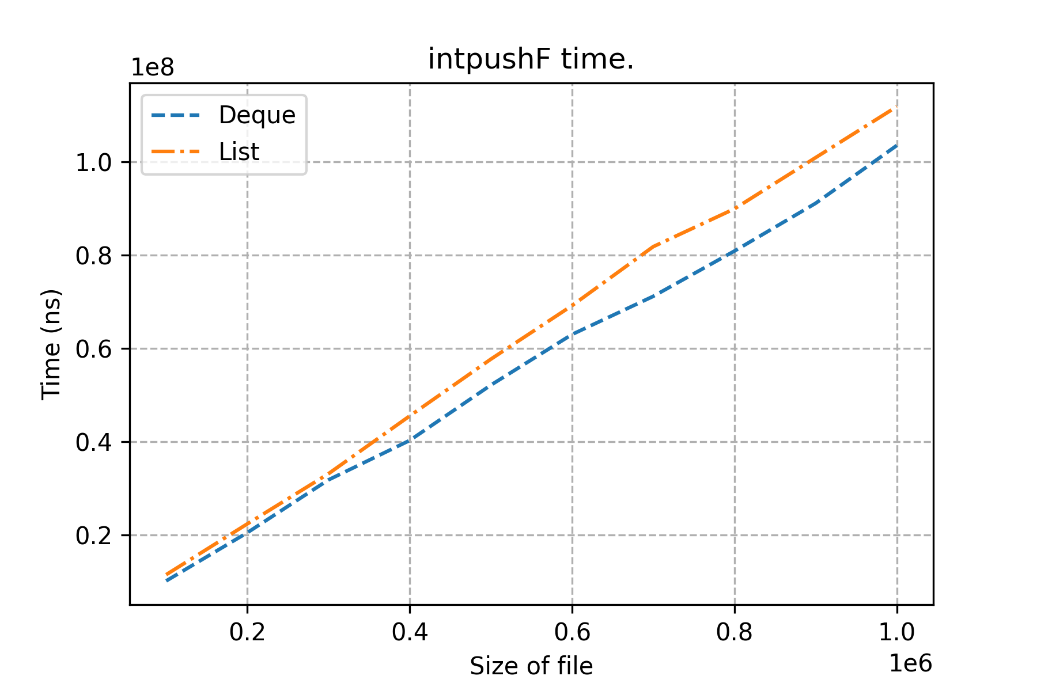


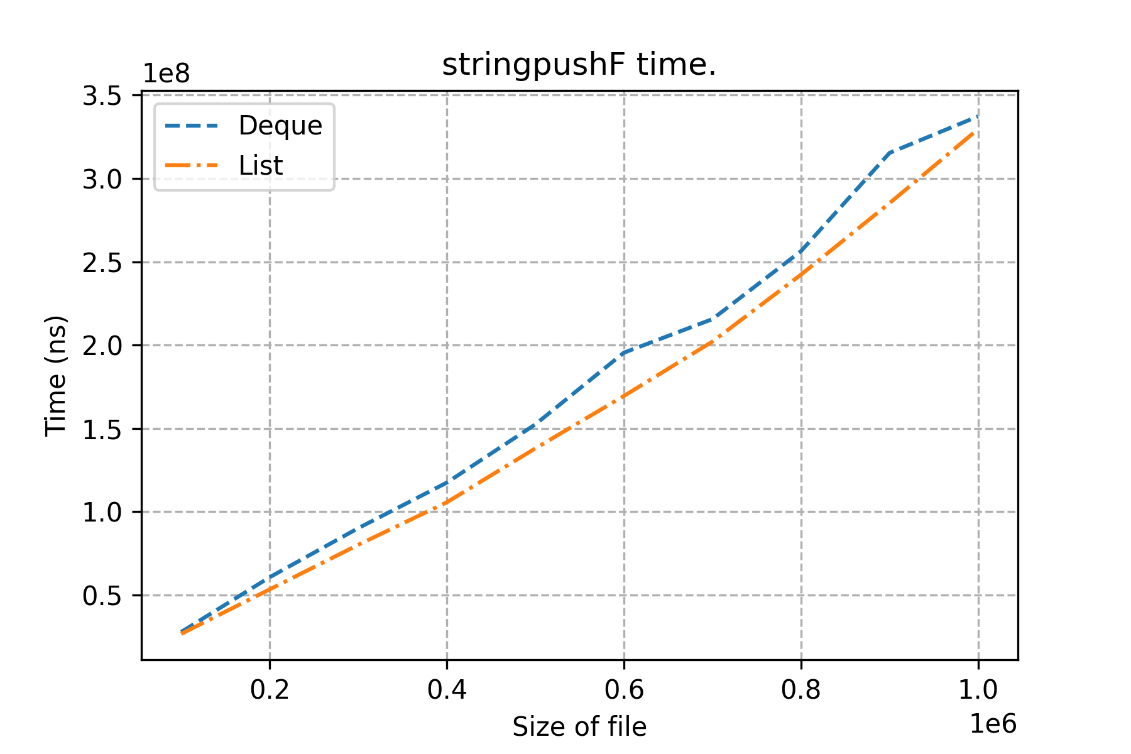
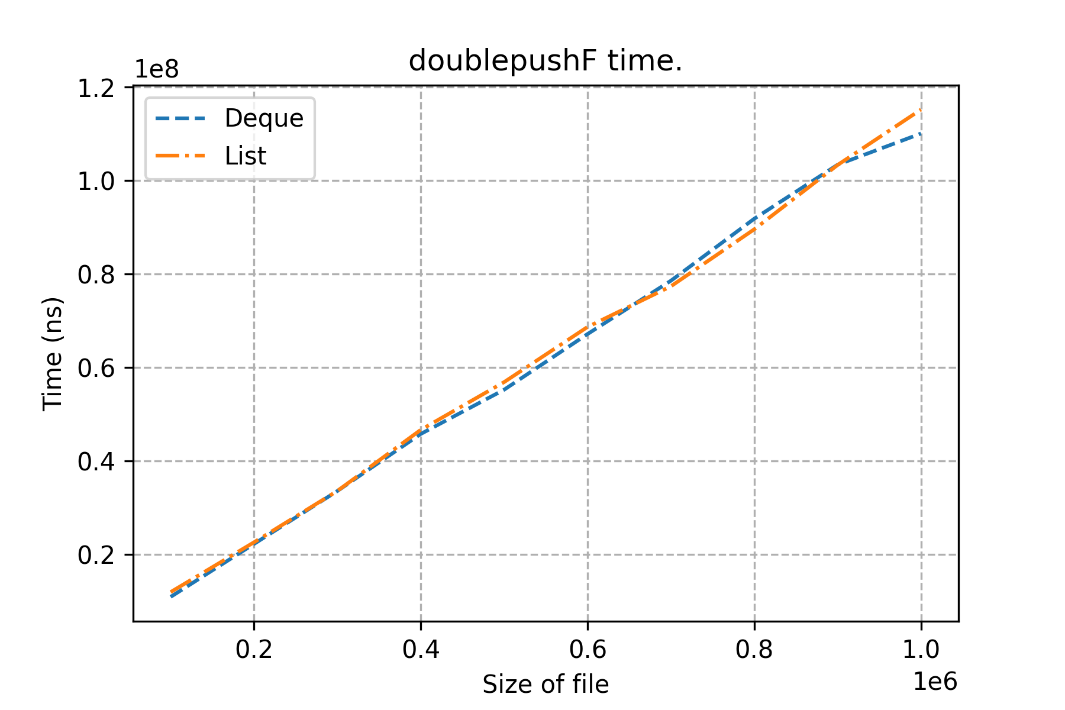


В целом, ситуации схожи! Только, в очередной раз, выделился случай со string – данными.

1. И последнее – pushF!







Результаты очень схожи! На string чуть лучше Deque, а на char и int – List.

Vector – контейнер опять отсутствует по причине того, что метод не реализован, а костыль работает долго

**Код**

Main.cpp

#include "TimeLogger.h"

#include "Load.h"

#include "Experiments.h"

int main() {

experiments::ex\_all();

}

TimeLogger.h

#pragma once

#include <iostream>

#include <fstream>

#include <chrono>

#include <string>

namespace time\_log {

class TimeLogger {

public:

TimeLogger(const std::string& path\_to\_out\_file);

using clock = std::chrono::high\_resolution\_clock;

using time\_point = std::chrono::time\_point<clock>;

void reset\_start();

void log\_duration(int cont\_size, const char\* type, const char\* opt\_message);

void log\_duration(int cont\_size, const char\* type);

private:

time\_point m\_start;

std::ofstream m\_fout;

};

}

#pragma once

#include <vector>

#include <list>

#include <deque>

#include <string>

#include <iostream>

#include <fstream>

namespace data {

std::vector<char> load\_vector\_char(const std::string& path\_file);

std::vector<int> load\_vector\_int(const std::string& path\_file);

std::vector<double> load\_vector\_double(const std::string& path\_file);

std::vector<std::string> load\_vector\_string(const std::string& path\_file);

std::list<char> load\_list\_char(const std::string& path\_file);

std::list<int> load\_list\_int(const std::string& path\_file);

std::list<double> load\_list\_double(const std::string& path\_file);

std::list<std::string> load\_list\_string(const std::string& path\_file);

std::deque<char> load\_deque\_char(const std::string& path\_file);

std::deque<int> load\_deque\_int(const std::string& path\_file);

std::deque<double> load\_deque\_double(const std::string& path\_file);

std::deque<std::string> load\_deque\_string(const std::string& path\_file);

}

Load.h

#pragma once

#include <vector>

#include <list>

#include <deque>

#include <string>

#include <iostream>

#include <fstream>

namespace data {

std::vector<char> load\_vector\_char(const std::string& path\_file);

std::vector<int> load\_vector\_int(const std::string& path\_file);

std::vector<double> load\_vector\_double(const std::string& path\_file);

std::vector<std::string> load\_vector\_string(const std::string& path\_file);

std::list<char> load\_list\_char(const std::string& path\_file);

std::list<int> load\_list\_int(const std::string& path\_file);

std::list<double> load\_list\_double(const std::string& path\_file);

std::list<std::string> load\_list\_string(const std::string& path\_file);

std::deque<char> load\_deque\_char(const std::string& path\_file);

std::deque<int> load\_deque\_int(const std::string& path\_file);

std::deque<double> load\_deque\_double(const std::string& path\_file);

std::deque<std::string> load\_deque\_string(const std::string& path\_file);

}

Experiments.h

#pragma once

namespace experiments {

void ex\_all();

void ex\_find\_test();

void ex\_sort\_test();

void ex\_pop\_bf\_test();

void ex\_push\_bf\_test();

void find\_test\_char();

void find\_test\_int();

void find\_test\_double();

void find\_test\_string();

void sort\_test\_char();

void sort\_test\_int();

void sort\_test\_double();

void sort\_test\_string();

void popB\_test\_char();

void popB\_test\_int();

void popB\_test\_double();

void popB\_test\_string();

void popF\_test\_char();

void popF\_test\_int();

void popF\_test\_double();

void popF\_test\_string();

void pushB\_test\_char();

void pushB\_test\_int();

void pushB\_test\_double();

void pushB\_test\_string();

void pushF\_test\_char();

void pushF\_test\_int();

void pushF\_test\_double();

void pushF\_test\_string();

}

TimeLogger.cpp

#include "TimeLogger.h"

time\_log::TimeLogger::TimeLogger(const std::string& path\_to\_out\_file)

: m\_start(clock::now()), m\_fout(path\_to\_out\_file)

{

if (!m\_fout.is\_open()) {

std::cout << "The file can't be opened" << "\n";

std::cout << "Name of file is:" << path\_to\_out\_file << "\n";

return;

}

m\_fout << "sep=;\n";

m\_fout << "Duration (ns); Size; Type; Message\n";

}

void time\_log::TimeLogger::reset\_start() {

m\_start = clock::now();

}

void time\_log::TimeLogger::log\_duration(int cont\_size, const char\* type, const char\* opt\_message) {

std::chrono::nanoseconds duration = clock::now() - m\_start;

m\_fout << duration.count() << ';' << cont\_size << ';' << type << ';' << opt\_message << '\n';

}

void time\_log::TimeLogger::log\_duration(int cont\_size, const char\* type) {

std::chrono::nanoseconds duration = clock::now() - m\_start;

m\_fout << duration.count() << ';' << cont\_size << ';' << type << ';' << '\n';

}

Load.cpp

#include "Load.h"

// Сначала выпишем блок для векторов.

std::vector<char> data::load\_vector\_char(const std::string& path\_file) {

std::vector<char> vec;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened" << "\n";

std::cout << "Name of file is:" << path\_file << "\n";

return vec;

}

for (char el; !fin.eof();) {

if (fin >> el) {

vec.push\_back(el);

}

}

fin.close();

return vec;

}

std::vector<int> data::load\_vector\_int(const std::string& path\_file) {

std::vector<int> vec;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return vec;

}

for (int el; !fin.eof();) {

if (fin >> el) {

vec.push\_back(el);

}

}

fin.close();

return vec;

}

std::vector<double> data::load\_vector\_double(const std::string& path\_file) {

std::vector<double> vec;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return vec;

}

for (double el; !fin.eof();) {

if (fin >> el) {

vec.push\_back(el);

}

}

fin.close();

return vec;

}

std::vector<std::string> data::load\_vector\_string(const std::string& path\_file) {

std::vector<std::string> vec;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return vec;

}

for (std::string el; !fin.eof();) {

if (fin >> el) {

vec.push\_back(el);

}

}

fin.close();

return vec;

}

// Теперь определим функции для листа.

std::list<char> data::load\_list\_char(const std::string& path\_file) {

std::list<char> lst;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return lst;

}

for (char el; !fin.eof();) {

if (fin >> el) {

lst.push\_back(el);

}

}

fin.close();

return lst;

}

std::list<int> data::load\_list\_int(const std::string& path\_file) {

std::list<int> lst;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return lst;

}

for (int el; !fin.eof();) {

if (fin >> el) {

lst.push\_back(el);

}

}

fin.close();

return lst;

}

std::list<double> data::load\_list\_double(const std::string& path\_file) {

std::list<double> lst;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return lst;

}

for (double el; !fin.eof();) {

if (fin >> el) {

lst.push\_back(el);

}

}

fin.close();

return lst;

}

std::list<std::string> data::load\_list\_string(const std::string& path\_file) {

std::list<std::string> lst;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return lst;

}

for (std::string el; !fin.eof();) {

if (fin >> el) {

lst.push\_back(el);

}

}

fin.close();

return lst;

}

// И, наконец, напишем реализацию для deque.

std::deque<char> data::load\_deque\_char(const std::string& path\_file) {

std::deque<char> deq;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return deq;

}

for (char el; !fin.eof();) {

if (fin >> el) {

deq.push\_back(el);

}

}

fin.close();

return deq;

}

std::deque<int> data::load\_deque\_int(const std::string& path\_file) {

std::deque<int> deq;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return deq;

}

for (int el; !fin.eof();) {

if (fin >> el) {

deq.push\_back(el);

}

}

fin.close();

return deq;

}

std::deque<double> data::load\_deque\_double(const std::string& path\_file) {

std::deque<double> deq;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return deq;

}

for (double el; !fin.eof();) {

if (fin >> el) {

deq.push\_back(el);

}

}

fin.close();

return deq;

}

std::deque<std::string> data::load\_deque\_string(const std::string& path\_file) {

std::deque<std::string> deq;

std::ifstream fin(path\_file);

if (!fin.is\_open()) {

std::cout << "The file can't be opened";

std::cout << "Name of file is:" << path\_file << "\n";

return deq;

}

for (std::string el; !fin.eof();) {

if (fin >> el) {

deq.push\_back(el);

}

}

fin.close();

return deq;

}

Experiments.cpp

#include "Experiments.h"

#include "Load.h"

#include "TimeLogger.h"

#include <string>

#include <algorithm>

#include <iterator>

std::string data\_path = "C:\\Users\\ender\\Desktop\\LP1\\Generator\\Data\\";

std::string reults\_path = "C:\\Users\\ender\\Desktop\\LP1\\Results\\";

void experiments::ex\_all() {

ex\_find\_test();

ex\_sort\_test();

ex\_pop\_bf\_test();

ex\_push\_bf\_test();

}

void experiments::ex\_find\_test() {

find\_test\_char();

find\_test\_int();

find\_test\_double();

find\_test\_string();

}

void experiments::ex\_sort\_test() {

sort\_test\_char();

sort\_test\_int();

sort\_test\_double();

sort\_test\_string();

}

void experiments::ex\_pop\_bf\_test() {

popB\_test\_char();

popB\_test\_int();

popB\_test\_double();

popB\_test\_string();

popF\_test\_char();

popF\_test\_int();

popF\_test\_double();

popF\_test\_string();

}

void experiments::ex\_push\_bf\_test() {

pushB\_test\_char();

pushB\_test\_int();

pushB\_test\_double();

pushB\_test\_string();

pushF\_test\_char();

pushF\_test\_int();

pushF\_test\_double();

pushF\_test\_string();

}

// 1 блок - поиск.

void experiments::find\_test\_char() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "find\_test\_char.csv");

std::string TD = "char\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<char> arr = data::load\_vector\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<char> arr = data::load\_deque\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<char> arr = data::load\_list\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "List");

}

}

void experiments::find\_test\_int() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "find\_test\_int.csv");

std::string TD = "int\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<int> arr = data::load\_vector\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<int> arr = data::load\_deque\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<int> arr = data::load\_list\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "List");

}

}

void experiments::find\_test\_double() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "find\_test\_double.csv");

std::string TD = "double\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<double> arr = data::load\_vector\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<double> arr = data::load\_deque\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<double> arr = data::load\_list\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), '/');

timer.log\_duration(i, "List");

}

}

void experiments::find\_test\_string() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "find\_test\_string.csv");

std::string TD = "string\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<std::string> arr = data::load\_vector\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), "/");

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<std::string> arr = data::load\_deque\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), "/");

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<std::string> arr = data::load\_list\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::find(arr.begin(), arr.end(), "/");

timer.log\_duration(i, "List");

}

}

// 2 блок - сортировка.

void experiments::sort\_test\_char() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "sort\_test\_char.csv");

std::string TD = "char\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<char> arr = data::load\_vector\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<char> arr = data::load\_deque\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<char> arr = data::load\_list\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

arr.sort();

timer.log\_duration(i, "List");

}

}

void experiments::sort\_test\_int() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "sort\_test\_int.csv");

std::string TD = "int\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<int> arr = data::load\_vector\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<int> arr = data::load\_deque\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<int> arr = data::load\_list\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

arr.sort();

timer.log\_duration(i, "List");

}

}

void experiments::sort\_test\_double() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "sort\_test\_double.csv");

std::string TD = "double\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<double> arr = data::load\_vector\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<double> arr = data::load\_deque\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<double> arr = data::load\_list\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

arr.sort();

timer.log\_duration(i, "List");

}

}

void experiments::sort\_test\_string() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "sort\_test\_string.csv");

std::string TD = "string\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<std::string> arr = data::load\_vector\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<std::string> arr = data::load\_deque\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

std::sort(arr.begin(), arr.end());

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<std::string> arr = data::load\_list\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

arr.sort();

timer.log\_duration(i, "List");

}

}

// 3.1 блок - popB.

void experiments::popB\_test\_char() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popB\_test\_char.csv");

std::string TD = "char\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<char> arr = data::load\_vector\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<char> arr = data::load\_deque\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<char> arr = data::load\_list\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "List");

}

}

void experiments::popB\_test\_int() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popB\_test\_int.csv");

std::string TD = "int\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<int> arr = data::load\_vector\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<int> arr = data::load\_deque\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<int> arr = data::load\_list\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "List");

}

}

void experiments::popB\_test\_double() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popB\_test\_double.csv");

std::string TD = "double\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<double> arr = data::load\_vector\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<double> arr = data::load\_deque\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<double> arr = data::load\_list\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "List");

}

}

void experiments::popB\_test\_string() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popB\_test\_string.csv");

std::string TD = "string\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<std::string> arr = data::load\_vector\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<std::string> arr = data::load\_deque\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<std::string> arr = data::load\_list\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_back();

timer.log\_duration(i, "List");

}

}

// 3.2 блок - popF;

void experiments::popF\_test\_char() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popF\_test\_char.csv");

std::string TD = "char\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<char> arr = data::load\_vector\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.erase(arr.begin());

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<char> arr = data::load\_deque\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<char> arr = data::load\_list\_char(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "List");

}

}

void experiments::popF\_test\_int() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popF\_test\_int.csv");

std::string TD = "int\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<int> arr = data::load\_vector\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.erase(arr.begin());

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<int> arr = data::load\_deque\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<int> arr = data::load\_list\_int(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "List");

}

}

void experiments::popF\_test\_double() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popF\_test\_double.csv");

std::string TD = "double\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<double> arr = data::load\_vector\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.erase(arr.begin());

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<double> arr = data::load\_deque\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<double> arr = data::load\_list\_double(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "List");

}

}

void experiments::popF\_test\_string() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "popF\_test\_string.csv");

std::string TD = "string\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<std::string> arr = data::load\_vector\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.erase(arr.begin());

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<std::string> arr = data::load\_deque\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<std::string> arr = data::load\_list\_string(data\_path + TD + data\_size + ".txt");

timer.reset\_start();

while (arr.size() > 0)

arr.pop\_front();

timer.log\_duration(i, "List");

}

}

// 4.1 блок - pushB.

void experiments::pushB\_test\_char() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushB\_test\_char.csv");

std::string TD = "char\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<char> arr = data::load\_vector\_char(data\_path + TD + data\_size + ".txt");

std::vector<char> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<char> arr = data::load\_deque\_char(data\_path + TD + data\_size + ".txt");

std::deque<char> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<char> arr = data::load\_list\_char(data\_path + TD + data\_size + ".txt");

std::list<char> tmp;

std::list<char>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_back(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}

void experiments::pushB\_test\_int() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushB\_test\_int.csv");

std::string TD = "int\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<int> arr = data::load\_vector\_int(data\_path + TD + data\_size + ".txt");

std::vector<int> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<int> arr = data::load\_deque\_int(data\_path + TD + data\_size + ".txt");

std::deque<int> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<int> arr = data::load\_list\_int(data\_path + TD + data\_size + ".txt");

std::list<int> tmp;

std::list<int>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_back(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}

void experiments::pushB\_test\_double() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushB\_test\_double.csv");

std::string TD = "double\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<double> arr = data::load\_vector\_double(data\_path + TD + data\_size + ".txt");

std::vector<double> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<double> arr = data::load\_deque\_double(data\_path + TD + data\_size + ".txt");

std::deque<double> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<double> arr = data::load\_list\_double(data\_path + TD + data\_size + ".txt");

std::list<double> tmp;

std::list<double>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_back(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}

void experiments::pushB\_test\_string() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushB\_test\_string.csv");

std::string TD = "string\_";

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<std::string> arr = data::load\_vector\_string(data\_path + TD + data\_size + ".txt");

std::vector<std::string> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Vector");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<std::string> arr = data::load\_deque\_string(data\_path + TD + data\_size + ".txt");

std::deque<std::string> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_back(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<std::string> arr = data::load\_list\_string(data\_path + TD + data\_size + ".txt");

std::list<std::string> tmp;

std::list<std::string>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_back(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}

// 4.2 блок - pushF.

void experiments::pushF\_test\_char() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushF\_test\_char.csv");

std::string TD = "char\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<char> arr = data::load\_vector\_char(data\_path + TD + data\_size + ".txt");

std::vector<char> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.insert(tmp.begin(), arr[i]);

}

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<char> arr = data::load\_deque\_char(data\_path + TD + data\_size + ".txt");

std::deque<char> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_front(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<char> arr = data::load\_list\_char(data\_path + TD + data\_size + ".txt");

std::list<char> tmp;

std::list<char>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_front(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}

void experiments::pushF\_test\_int() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushF\_test\_int.csv");

std::string TD = "int\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<int> arr = data::load\_vector\_int(data\_path + TD + data\_size + ".txt");

std::vector<int> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.insert(tmp.begin(), arr[i]);

}

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<int> arr = data::load\_deque\_int(data\_path + TD + data\_size + ".txt");

std::deque<int> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_front(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<int> arr = data::load\_list\_int(data\_path + TD + data\_size + ".txt");

std::list<int> tmp;

std::list<int>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_front(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}

void experiments::pushF\_test\_double() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushF\_test\_double.csv");

std::string TD = "double\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<double> arr = data::load\_vector\_double(data\_path + TD + data\_size + ".txt");

std::vector<double> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.insert(tmp.begin(), arr[i]);

}

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<double> arr = data::load\_deque\_double(data\_path + TD + data\_size + ".txt");

std::deque<double> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_front(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<double> arr = data::load\_list\_double(data\_path + TD + data\_size + ".txt");

std::list<double> tmp;

std::list<double>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_front(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}

void experiments::pushF\_test\_string() {

time\_log::TimeLogger timer = time\_log::TimeLogger(reults\_path + "pushF\_test\_string.csv");

std::string TD = "string\_";

/\* Долго работает(!!!). Мне передали, что можно написать, но не использовать.

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::vector<std::string> arr = data::load\_vector\_string(data\_path + TD + data\_size + ".txt");

std::vector<std::string> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.insert(tmp.begin(), arr[i]);

}

timer.log\_duration(i, "Vector");

}

\*/

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::deque<std::string> arr = data::load\_deque\_string(data\_path + TD + data\_size + ".txt");

std::deque<std::string> tmp;

timer.reset\_start();

for (int i = 0; i < arr.size(); ++i) {

tmp.push\_front(arr[i]);

}

timer.log\_duration(i, "Deque");

}

for (int i = 100000; i <= 1000000; i += 100000) {

std::string data\_size = std::to\_string(i);

std::list<std::string> arr = data::load\_list\_string(data\_path + TD + data\_size + ".txt");

std::list<std::string> tmp;

std::list<std::string>::iterator it = arr.begin();

timer.reset\_start();

while (it != arr.end()) {

tmp.push\_front(\*it);

++it;

}

timer.log\_duration(i, "List");

}

}