|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
|  | | |
|  | | |
| Практическое задание № 1 | | |
| по дисциплине «Методы построения и анализа алгоритмов» | | |
|  | | |
| **кафедра теоретической и прикладной информатики** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-03 |
| Бригада: | Место для ввода текста. |
| Студенты: | Сидоров Даниил, |
|  | Малыгин Сергей |
|  |  |
| Преподаватели: | Щукин Георгий Анатольевич |
|  |  |
|  | | |
| Новосибирск | | |
| 2021 | | |

**1.Результаты замеров:**

Оптимизация по скорости O2:

Заполнение

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | Vector(в конец) | Vector(в начало) | List(в конец) | List(в начало) | Set | unordered\_set |
| 104 | 0.0001455 sec | 0.0063598 sec | 0.0018451 sec | 0.0016044 sec | 0.0062183 sec | 0.0040329 sec |
| 105 | 0.0014509 sec | 0.942976 sec | 0.0108463 sec | 0.0166377 sec | 0.0595373 sec | 0.0403319 sec |
| 106 | 0.0070923 sec | 123.038 sec | 0.105709 sec | 0.159866 sec | 1.32033 sec | 0.325466 sec |
| 107 | 0.0840399 sec |  | 1.28219 sec | 1.54543 sec | 28.3918 sec | 7.37499 sec |
| 108 | 1.08807 sec |  | 14.2375 sec | 16.9303 sec |  |  |

Итерация

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | Vector(в конец) | Vector(в начало) | List(в конец) | List(в начало) | Set | unordered\_set |
| 104 | 0.000396 sec | 0.0004376 sec | 0.0008142 sec | 0.000662 sec | 0.0005038 sec | 0.0002698 sec |
| 105 | 0.0005398 sec | 0.0006325 sec | 0.0032569 sec | 0.0026417 sec | 0.0039288 sec | 0.0044832 sec |
| 106 | 0.0021575 sec | 0.0008868 sec | 0.020365 sec | 0.0228606 sec | 0.10725 sec | 0.0563819 sec |
| 107 | 0.0065134 sec |  | 0.274478 sec | 0.181822 sec | 1.79868 sec | 0.945171 sec |
| 108 | 0.13996 sec |  | 5.1677 sec | 2.93298 sec |  |  |

Поиск(4000,100,1000,1000)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | Vector(в конец) | Vector(в начало) | List(в конец) | List(в начало) | Set | unordered\_set |
| 104 | 0.0134318 sec | 0.0133646 sec | 0.0022534 sec | 0.0035933 sec | 0.0002751 sec | 7.16e-05 sec |
| 105 | 0.132124 sec | 0.176397 sec | 0.145446 sec | 0.154893 sec | 0.0002969 sec | 8.73e-05 sec |
| 106 | 2.45154 sec | 1.73827 sec | 1.53378 sec | 1.4668 sec | 0.0007326 sec | 7.97e-05 sec |
| 107 | 21.3603 sec |  | 15.3323 sec | 13.3198 sec | 0.0017447 sec | 0.0001404 sec |
| 108 | 206.111 sec |  | 170.678 sec | 150.303 sec |  |  |

2.**Программы:**

Вектор:

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <numeric>

#include <random>

#include <chrono>

using namespace std;

// Returns shuffled sequence of unique numbers of specified size, with values from start to start + size - 1.

vector<int> shuffled\_sequence(int size, int start = 0) {

vector<int> result(size);

iota(result.begin(), result.end(), start);

random\_shuffle(result.begin(), result.end());

return result;

}

// Returns sequence of random numbers of specified size, with values from 0 to max.

vector<int> random\_sequence(int size, int max) {

default\_random\_engine generator;

uniform\_int\_distribution<int> distribution(0, max);

vector<int> result;

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

result.push\_back(distribution(generator));

}

return result;

}

int main() {

const int size =10000000;

// Container to use.

vector<int> container;

// Insert elements into container.

const auto elems\_to\_add = shuffled\_sequence(size);

auto t1 = std::chrono::high\_resolution\_clock::now();

for (const auto &elem : elems\_to\_add) {

container.push\_back(elem);

}

auto t2 = std::chrono::high\_resolution\_clock::now();

auto seconds = std::chrono::duration<double>(t2 - t1).count();

std::cout << "Time: " << seconds << " sec." << std::endl;

// Iterate through elements.

long long sum = 0;

t1 = std::chrono::high\_resolution\_clock::now();

for (const auto &elem : container) {

sum += elem;

}

cout << "Sum is " << sum << endl;

t2 = std::chrono::high\_resolution\_clock::now();

seconds = std::chrono::duration<double>(t2 - t1).count();

std::cout << "Time: " << seconds << " sec." << std::endl;

// Perform search into container.

int hits = 0;

const auto elems\_to\_search = random\_sequence(4000, 2 \* size);

t1 = std::chrono::high\_resolution\_clock::now();

for (const auto &elem : elems\_to\_search) {

auto it = find(container.begin(), container.end(), elem);

if (it != container.end()) {

hits++;

}

}

cout << "Found " << hits << " elements" << endl;

t2 = std::chrono::high\_resolution\_clock::now();

seconds = std::chrono::duration<double>(t2 - t1).count();

std::cout << "Time: " << seconds << " sec." << std::endl;

}

**Списки:**

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <numeric>

#include <random>

#include <chrono>

#include <list>

using namespace std;

vector<int> random\_sequence(int size, int max) {

default\_random\_engine generator;

uniform\_int\_distribution<int> distribution(0, max);

vector<int> result;

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

//result.push\_back(distribution(generator));

result.insert(result.begin(), distribution(generator));

}

return result;

}

vector<int> shuffled\_sequence(int size, int start = 0) {

vector<int> result(size);

iota(result.begin(), result.end(), start);

random\_shuffle(result.begin(), result.end());

return result;

}

int main() {

std::list<int> List;

const int size = 100000000;

const auto elems\_to\_add = shuffled\_sequence(size);

auto t1 = std::chrono::high\_resolution\_clock::now();

for (const auto &elem : elems\_to\_add) {

// List.push\_back(elem);

List.insert(List.begin(), elem);

}

auto t2 = std::chrono::high\_resolution\_clock::now();

auto seconds = std::chrono::duration<double>(t2 - t1).count();

cout << "Time: " << seconds << " sec." << endl;

long long sum = 0;

t1 = std::chrono::high\_resolution\_clock::now();

for (const auto &elem : List) {

sum += elem;

}

cout << "Sum is " << sum << endl;

t2 = std::chrono::high\_resolution\_clock::now();

seconds = std::chrono::duration<double>(t2 - t1).count();

cout << "Time: " << seconds << " sec." << endl;

int hits = 0;

const auto elems\_to\_search = random\_sequence(100, 2 \* size);

t1 = std::chrono::high\_resolution\_clock::now();

for (const auto &elem : elems\_to\_search) {

auto it = find(List.begin(), List.end(), elem);

if (it != List.end()) {

hits++;

}

}

cout << "Found " << hits << " elements" << endl;

t2 = std::chrono::high\_resolution\_clock::now();

seconds = std::chrono::duration<double>(t2 - t1).count();

cout << "Time: " << seconds << " sec." << endl;

}

**Множества:**

#include<iostream>

#include<set>

#include<unordered\_set>

#include<chrono>

#include <random>

#include <vector>

#include <string>

#include <algorithm>

#include <numeric>

using namespace std;

using namespace chrono;

vector<int> shuffled\_sequence(int size, int start = 0)

{

vector<int> result(size);

iota(result.begin(), result.end(), start);

random\_shuffle(result.begin(), result.end());

return result;

}

vector<int> random\_sequence(int size, int max)

{

default\_random\_engine generator;

uniform\_int\_distribution<int> distribution(0, max);

vector<int> result;

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

result.push\_back(distribution(generator));

}

return result;

}

const int MIN = 10000;

const int MAX = 10000000;

int main()

{

set<int> SET;

cout << "SET" << endl;

for (int i(MIN); i <= MAX; i \*= 10)

{

cout << "i = " << i << endl;

cout << "[SET.insert]::Time = ";

auto elems\_to\_add = shuffled\_sequence(i);

auto t1 = high\_resolution\_clock::now();

for (auto& elem : elems\_to\_add)

{

SET.insert(elem);

}

auto t2 = high\_resolution\_clock::now();

elems\_to\_add.clear();

auto seconds = duration<double>(t2 - t1).count();

cout << seconds << endl << endl;

long long int sum(0);

t1 = high\_resolution\_clock::now();

for (auto& elem : SET)

{

sum = sum + elem;

}

t2 = high\_resolution\_clock::now();

seconds = duration<double>(t2 - t1).count();

cout << "[SET.sum]::Time = " << seconds << endl;

cout << "sum = " << sum << endl << endl;

int hits = 0;

auto elems\_to\_search = random\_sequence(1000, 2 \* i);

t1 = high\_resolution\_clock::now();

for (auto& elem : elems\_to\_search)

{

auto it = SET.find(elem);

if (it != SET.end())

{

hits++;

}

}

t2 = high\_resolution\_clock::now();

cout << "Found " << hits << " elements" << endl;

seconds = duration<double>(t2 - t1).count();

cout << "[SET.find]::Time = " << seconds << endl << endl << endl;

elems\_to\_search.clear();

SET.clear();

}

#include<iostream>

#include<set>

#include<unordered\_set>

#include<chrono>

#include <random>

#include <vector>

#include <string>

#include <algorithm>

#include <numeric>

using namespace std;

using namespace chrono;

vector<int> shuffled\_sequence(int size, int start = 0)

{

vector<int> result(size);

iota(result.begin(), result.end(), start);

random\_shuffle(result.begin(), result.end());

return result;

}

vector<int> random\_sequence(int size, int max)

{

default\_random\_engine generator;

uniform\_int\_distribution<int> distribution(0, max);

vector<int> result;

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

result.push\_back(distribution(generator));

}

return result;

}

const int MIN = 10000;

const int MAX = 10000000;

int main()

{

set<int> SET;

cout << "SET" << endl;

for (int i(MIN); i <= MAX; i \*= 10)

{

cout << "i = " << i << endl;

cout << "[SET.insert]::Time = ";

auto elems\_to\_add = shuffled\_sequence(i);

auto t1 = high\_resolution\_clock::now();

for (auto& elem : elems\_to\_add)

{

SET.insert(elem);

}

auto t2 = high\_resolution\_clock::now();

elems\_to\_add.clear();

auto seconds = duration<double>(t2 - t1).count();

cout << seconds << endl << endl;

long long int sum(0);

t1 = high\_resolution\_clock::now();

for (auto& elem : SET)

{

sum = sum + elem;

}

t2 = high\_resolution\_clock::now();

seconds = duration<double>(t2 - t1).count();

cout << "[SET.sum]::Time = " << seconds << endl;

cout << "sum = " << sum << endl << endl;

int hits = 0;

auto elems\_to\_search = random\_sequence(1000, 2 \* i);

t1 = high\_resolution\_clock::now();

for (auto& elem : elems\_to\_search)

{

auto it = SET.find(elem);

if (it != SET.end())

{

hits++;

}

}

t2 = high\_resolution\_clock::now();

cout << "Found " << hits << " elements" << endl;

seconds = duration<double>(t2 - t1).count();

cout << "[SET.find]::Time = " << seconds << endl << endl << endl;

elems\_to\_search.clear();

SET.clear();

}

cout << "///////////////////////////////////////////" << endl << endl << endl;

unordered\_set<int> U\_SET;

cout << "U\_SET" << endl;

for (int i(MIN); i <= MAX; i \*= 10)

{

cout << "i = " << i << endl;

cout << "[U\_SET.insert]::Time = ";

auto elems\_to\_add = shuffled\_sequence(i);

auto t1 = high\_resolution\_clock::now();

for (auto& elem : elems\_to\_add)

{

U\_SET.insert(elem);

}

auto t2 = high\_resolution\_clock::now();

elems\_to\_add.clear();

auto seconds = duration<double>(t2 - t1).count();

cout << seconds << endl << endl;

long long int sum(0);

t1 = high\_resolution\_clock::now();

for (auto& elem : U\_SET)

{

sum = sum + elem;

}

t2 = high\_resolution\_clock::now();

seconds = duration<double>(t2 - t1).count();

cout << "[U\_SET.sum]::Time = " << seconds << endl;

cout << "sum = " << sum << endl << endl;

int hits = 0;

auto elems\_to\_search = random\_sequence(1000, 2 \* i);

t1 = high\_resolution\_clock::now();

for (auto& elem : elems\_to\_search)

{

auto it = U\_SET.find(elem);

if (it != U\_SET.end())

{

hits++;

}

}

t2 = high\_resolution\_clock::now();

cout << "Found " << hits << " elements" << endl;

seconds = duration<double>(t2 - t1).count();

cout << "[ U\_SET.find]::Time = " << seconds << endl << endl << endl;

elems\_to\_search.clear();

U\_SET.clear();

}

return 0;

}