|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
|  | | |
|  | | |
| Практическое задание №9 | | |
| 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 |
|  |  |
| Преподаватель: | Щукин Георгий Анатольевич |
|  |  |
|  | | |
| Новосибирск | | |
| 2021 | | |

**1.Результаты замеров:**

|  |  |  |
| --- | --- | --- |
| N | Жадный алгоритм | Наивная реализация |
| 10 | 1.2e-06 | 6.72e-05 |
| 15 | 1.7e-06 | 0.0020398 |
| 20 | 2.2e-06 | 0.0550147 |
| 25 | 2.2e-06 | 2.06682 |
| 30 | 2.7e-06 | 68.7308 |
| 100 | 9.2e-06 | - |
| 1 000 | 9.01e-05 | - |
| 10 000 | 0.0008388 | - |
| 100 000 | 0.00501 | - |
| 1 000 000 | 0.0534804 | - |
| 10 000 000 | 0.565111 | - |
| 100 000 000 | 5.72383 | - |

**2.Программа:**

**activities.cpp**

#include "activities.h"

#include <cmath>

#include <algorithm>

#include <numeric>

#include <random>

using namespace std;

vector<Activity> get\_max\_activities(const vector<Activity> &activities)

{

if (activities.empty()) return {};

vector<Activity>sortActivities = activities;

sort(sortActivities.begin(), sortActivities.end(), [](const Activity a, const Activity b)

{

return a.finish < b.finish;

});

Activity a1 = sortActivities[0];

Activity am = a1;

vector <Activity> ::iterator itA;

for (auto it = sortActivities.begin() + 1; it != sortActivities.end(); ++it)

{

if (a1.finish <= (\*it).start)

{

itA = it;

am = \*it;

break;

}

}

if (am == a1)

return { a1 };

vector<Activity> result = { a1, am };

for (auto it = itA; it != sortActivities.end(); ++it)

{

if (am.finish <= (\*it).start)

{

result.push\_back(\*it);

am = \*it;

}

}

return result;

}

vector<Activity> naive\_solution(const vector<Activity> &activities)

{

vector<int> size(activities.size());

vector<Activity> finding;

vector<Activity> result;

int min = 0;

if (activities.empty()) return {};

int i = 0;

bool flag = true;

while (i < activities.size())

{

size[i]++;

for (int j = i; j--;)

size[j] = size[j + 1] + 1;

if (size[0] > activities.size())

i++;

else

{

finding.clear();

i = 0;

flag = true;

for (int j = size.size(); j--;)

if (size[j] > 0) finding.push\_back(activities[size[j] - 1]);

for (int i = 0; i < finding.size() - 1 && flag; i++)

for (int j = i + 1; j < finding.size(); j++)

if (finding[i].finish > finding[j].start && finding[j].finish > finding[i].start)

{

flag = false;

break;

}

if (finding.size() > min && flag)

{

result = finding;

min = finding.size();

}

}

}

return result;

}

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

{

default\_random\_engine generator;

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

vector<Activity> result;

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

result.push\_back({ (int)distribution(generator), (int)distribution(generator) });

}

return result;

}

**main.cpp**

#define CATCH\_CONFIG\_RUNNER

#include "catch.hpp"

#include <vector>

#include <iostream>

#include "activities.h"

using namespace std;

int main(int argc, char\* argv[])

{

int result = Catch::Session().run(argc, argv);

for (int i = 10; i <= 100000000; i \*= 10)

{

cout << endl << "i: " << i;

vector<Activity> activities = random\_sequence(i, 1000);

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

vector<Activity> res = get\_max\_activities(activities);

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

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

cout << endl << "Greedy algorithm: " << seconds;

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

res = naive\_solution(activities);

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

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

cout << endl << "Naive solution: " << seconds;

}

return result;

}