МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**КАФЕДРА ИИТ**

ОТЧЁТ

по лабораторной работе №2

**«Алгоритмы сортировки»**

Выполнил:

Студент 1 курса

группы ПО-9

Харитонович Захар Сергеевич

Проверила:

Войцехович О. Ю.

Брест 2022

**Порядок выполнения работы**

*Вариант 23*

*Задание*

Реализовать 3 алгоритма сортировки:

сортировка подсчётом,

сортировка вставками,

пирамидальная сортировка;

и провести экспериментальное исследование их эффективности.

Необходимо измерить время работы каждого алгоритма при различном количестве элементов в массиве и заполнить таблицу для каждого алгоритма

По заполненной таблице построить для каждого алгоритма график зависимости времени его выполнения от числа элементов в массиве.

По результатам экспериментов определить какой алгоритм работает быстрее и почему.

**Реализация**

#include <iostream>

#include <ctime>

#include <random>

using namespace std;

int findMax(int \*, size\_t);

void sortCount(int \*, size\_t, int);

void sortInsert(int \*, size\_t);

void sortPyramid(int \*, size\_t);

void to\_heap(int \*, size\_t, size\_t);

int main() {

for (size\_t n = 50000; n <= 1000000; n += 50000) {

int \* arr1 = new int[n], \* arr2 = new int[n], \* arr3 = new int[n];

for (size\_t i = 0; i < n; i++) {

int temp = 3 \* rand();

arr1[i] = temp;

arr2[i] = temp;

arr3[i] = temp;

}

cout << "n = " << n << endl;

int k = findMax(arr1, n);

clock\_t time = clock();

sortCount(arr1, n, k);

time = clock() - time;

cout << "sortCount() run time is " << time << " ms" << endl;

time = clock();

sortInsert(arr2, n);

time = clock() - time;

cout << "sortInsert() run time is " << time << " ms" << endl;

time = clock();

sortPyramid(arr3, n);

time = clock() - time;

cout << "sortPyramid() run time is " << time << " ms" << endl;

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

delete[] arr1;

delete[] arr2;

delete[] arr3;

}

return 0;

}

int findMax(int \* arr, size\_t n) {

int maximum = 0;

for (size\_t i = 0; i < n; i++) {

maximum = max(maximum, arr[i]);

}

return maximum;

}

void sortCount(int \* arr, size\_t n, int maximum) {

int \* count = new int[maximum + 1];

for(size\_t i = 0; i < maximum; i++) {

count[i] = 0;

}

for (size\_t i = 0; i < n; i++) {

count[arr[i]]++;

}

int b = 0;

for (size\_t i = 0; i <= maximum; i++) {

for(size\_t j = 0; j < count[i]; j++) {

arr[b++] = i;

}

}

}

void sortInsert(int \* arr, size\_t n) {

for (int step = 1; step < n; step++) {

int key = arr[step];

int j;

for (j = step - 1; key < arr[j] && j >= 0; j--) {

arr[j+1] = arr[j];

}

arr[j+1] = key;

}

}

void sortPyramid(int \* arr, size\_t n) {

for (int i = n / 2 - 1; i >= 0; i--) {

to\_heap(arr, n, i);

}

for (int i = n - 1; i >= 0; i--) {

swap(arr[0], arr[i]);

to\_heap(arr, i, 0);

}

}

void to\_heap(int \* arr, size\_t n, size\_t root) {

size\_t largest = root;

size\_t left = root \* 2 + 1;

size\_t right = root \* 2 + 2;

if (left < n && arr[left] > arr[largest]) {

largest = left;

}

if (right < n && arr[right] > arr[largest]) {

largest = right;

}

if (largest != root) {

swap(arr[root], arr[largest]);

to\_heap(arr, n, largest);

}

}

*Таблица 1*

|  |  |  |  |
| --- | --- | --- | --- |
| n | Подсчётом, мс | Вставками, мс | Пирамидальная, мс |
| 50000 | 8 | 1424 | 20 |
| 100000 | 0 | 5712 | 24 |
| 150000 | 8 | 11178 | 32 |
| 200000 | 2 | 19468 | 50 |
| 250000 | 0 | 30254 | 67 |
| 300000 | 0 |  | 82 |
| 350000 | 4 |  | 194 |
| 400000 | 0 |  | 220 |
| 450000 | 16 |  | 250 |
| 500000 | 0 |  | 288 |
| 550000 | 0 |  | 314 |
| 600000 | 0 |  | 359 |
| 650000 | 16 |  | 375 |
| 700000 | 0 |  | 437 |
| 750000 | 0 |  | 531 |
| 800000 | 15 |  | 592 |
| 850000 | 15 |  | 547 |
| 900000 | 16 |  | 547 |
| 950000 | 0 |  | 609 |
| 1000000 | 15 |  | 617 |

*График 1 – зависимости времени выполнения алгоритмов от объема исходных данных.*

**Вывод:** были реализованы алгоритмы сортировки подсчётом, вставками, пирамидальной сортировки. Проведена экспериментальная оценка алгоритмов сортировки при различных наборах данных. Наилучший результат показал алгоритм сортировки подсчётом, так как он имеет линейную сложность.