Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: “Алгоритмы и структуры данных”

**Тема: “ Исследование и оценка алгоритмов поиска”**

Выполнил:

Студент 1 курса

Группа ПО-7

Комиссаров А.Е.

Проверила:

Глущенко Т.А.

Брест 2021

**Цель:** разработка программ, реализующих различные алгоритмы поиска, и оценка их временной и пространственной сложности.

**Ход работы**

**Вариант 6**

Разработать алгоритм и программу интерполирующего поиска. В качестве исходных данных использовать массив целых чисел, который формируется с помощью датчика случайных чисел с диапазоном от 0 до 100. Аргумент поиска – число.

1. **Псевдокод:**

int interpolationSearch(a : int[], key : int) // a должен быть отсортирован

left = 0 // левая граница поиска (будем считать, что элементы массива нумеруются с нуля)

right = a.length - 1 // правая граница поиска

while a[left] < key and key < a[right]

mid = left + (key - a[left]) \* (right - left) / (a[right] - a[left]) // индекс элемента, с которым будем проводить сравнение

if a[mid] < key

left = mid + 1

else if a[mid] > key

right = mid - 1

else

return mid

if a[left] == key

return left

else if a[right] == key

return right

else

return -1 // если такого элементав массиве нет

1. **Словесное описание:**

В основе интерполяционного поиска лежит операция интерполирование. Интерполирование – нахождение промежуточных значений величины по имеющемуся дискретному набору известных значений. Интерполяционный поиск работает только с упорядоченными массивами; он похож на бинарный, в том смысле, что на каждом шаге вычисляется некоторая область поиска, которая, по мере выполнения алгоритма, сужается.

Но в отличие от двоичного, интерполяционный поиск не делит последовательность на две равные части, а вычисляет приблизительное расположение ключа (искомого элемента), ориентируясь на расстояние между искомым и текущим значением элемента.

Идея алгоритма напоминает хорошо знакомый старшим поколениям поиск телефонного номера в обычном справочнике: список имен абонентов упорядочен, поэтому не составит труда найти нужный телефонный номер, так как, если мы, например, ищем абонента с фамилией, начинающейся на букву «Ю», то для дальнейшего поиска разумно будет перейти в конец справочника.

1. **Текст программы**

#include <iostream>

#include <algorithm>

#include <ctime>

using namespace std;

void fill(int\* massiv, int n) {

for (int i = 0; i < n; i++) { massiv[i] = rand() % 10; }

}

void output(int massiv[], int size) {

for (int i = 0; i < size; ++i) { cout << massiv[i] << " "; }cout << endl;

}

int interpolation\_search(int massiv[], int value, int size) {

int low = 0, high = size - 1, mid;

while (massiv[low] <= value && massiv[high] >= value) {

mid = low + ((value - massiv[low]) \* (high - low)) / (massiv[high] - massiv[low]);

if (massiv[mid] < value) { low = mid + 1; }

else if (massiv[mid] > value) { low = mid - 1; }

else { return mid; }

}

if (massiv[low] == value) { return low; }

else { return -1; }

}

int main() {

setlocale(LC\_ALL, "Russian");

unsigned int start\_time = clock();

int n, value, pos;

cout << "Введите кол-во элементов:"; cin >> n;

int\* massiv = new int[n];

fill(massiv, n);

sort(massiv, massiv + n);

//cout << "Исходный массив: ";

//output(massiv, n);

cout << "Введите значение для поиска: ";

cin >> value;

unsigned int end\_time = clock();

pos = interpolation\_search(massiv, value, n);

unsigned int search\_time = end\_time - start\_time;

if (pos != -1) {

cout << "Позиция значения в массиве: " << pos + 1 << endl;

}

else {

cout << "Значение не найдено." << endl;

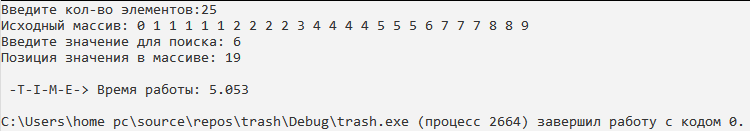
}

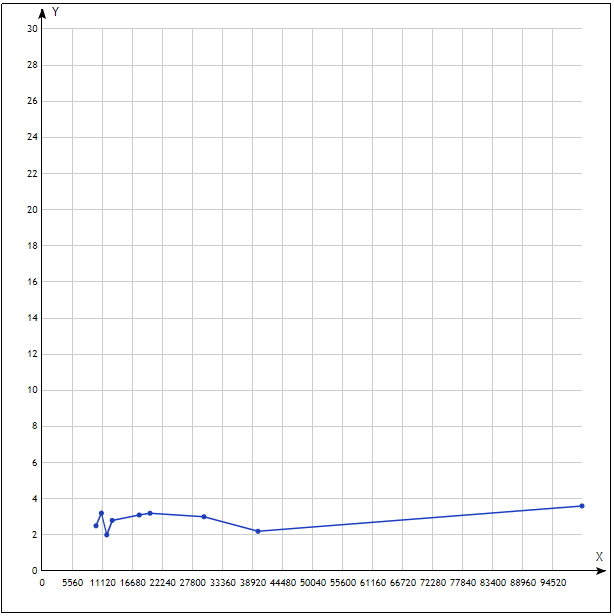
cout << "\n -T-I-M-E-> Время работы: " << search\_time / 1000.0 << endl;

return 0;

}

**Результат выполнения программы:**

****

****

**Вывод:** в ходе лабораторной работы разработал программу, реализующую интерполяционный алгоритм поиска, а также оценку его сложности.