Лабораторная работа 1

Артамоновой Анастасии ПИН-24

**Конспект**

Методы сортировки

1. Сортировка в линейных структурах
   1. Сортировка выбором

Сортировка выбором состоит в том, что сначала в неупорядоченном списке выбирается и отделяется от остальных наименьший элемент. После этого исходный список оказывается измененным. Измененный список принимается за исходный и процесс продолжается до тех пор, пока все элементы не будут выбраны. Очевидно, что выбранные элементы образуют упорядоченный список.

* 1. Сортировка вставкой

Метод сортировки вставкой предусматривает поочередный выбор из неупорядоченной последовательности элементов каждого элемента, сравнение его с предыдущим, уже упорядоченным, и перемещение на соответствующее место.

* 1. Сортировка бинарной вставкой

При использовании метода сортировки вставками ключ i-го элемента сравнивается приблизительно с i/2 ключами уже отсортированных элементов. Для того чтобы ускорить этот процесс, Джон Мочлив предложил использовать метод бинарных вставок, т.е. сравнивать ключ i-го элемента (Ki) с ключом середины уже отсортированной последовательности (Ki /2). Затем в зависимости от того, ключ Ki меньше или больше Ki /2, будем рассматривать следующие элементы: [K1; Ki /2–1] или [Ki /2 + 1; Ki]. Для решения проблемы нечетного числа элементов множества номер центрального элемента вычисляется по формуле: mid: = (low + high) div 2.

* 1. Сортировка обменом

В сортировке обменом элементы списка последовательно сравниваются между собой и меняются местами в том случае, если предшествующий элемент больше последующего.

* 1. Сортировка методом Шелла

В методе Шелла сравниваются не соседние элементы, а элементы, расположенные на расстоянии d (где d - шаг между сравниваемыми элементами): d = [n/2]. После каждого просмотра шаг d уменьшается вдвое. На последнем просмотре он сокращается до d = 1.

* 1. Сортировка Хоара

В методе быстрой сортировки фиксируется базовый ключ, относительно которого все элементы с большим весом перемещаются вправо, а с меньшим - влево. В качестве базового элемента обычно выбирается любой крайний элемент исходного множества, который постоянно сравнивается с противоположно стоящими элементами. При этом весь список элементов делится относительно базового ключа на две части. Для каждой части процесс повторяется.

1. Сортировка в нелинейных структурах
   1. Турнирная сортировка

В методе быстрой сортировки фиксируется базовый ключ, относительно которого все элементы с большим весом перемещаются вправо, а с меньшим - влево. В качестве базового элемента обычно выбирается любой крайний элемент исходного множества, который постоянно сравнивается с противоположно стоящими элементами. При этом весь список элементов делится относительно базового ключа на две части. Для каждой части процесс повторяется.

* 1. Пирамидальная сортировка

Данный тип сортировки заключается в построении пирамидального дерева. В ходе преобразования элементы триад сравниваются дважды, при этом элемент с большим весом перемещается вверх, а с меньшим – вниз.

* 1. Топологическая сортировка

Топологическая сортировка - это одна из известных задач в программировании, в которой широко применяются списки. Под топологической сортировкой понимается сортировка элементов, для которых определен частичный порядок, т.е. упорядочение задано не на всех, а только на некоторых парах элементов.

Методы поиска

1. Основанные на сравнение ключей
   1. Последовательный поиск

В последовательном поиске исходное множество не упорядоченное, т.е. имеется произвольный набор ключей {k1, k2, k3,..., kn}. Метод заключается в том, что отыскиваемый ключ K последовательно сравнивается со всеми элементами множества. При этом поиск заканчивается досрочно, если ключ найден.

1.2. Бинарный поиск

В бинарном поиске исходное множество должно быть упорядоченным по возрастанию. Иными словами, каждый последующий ключ больше предыдущего: {k1 ≤ k2 ≤ k3 ≤ k4 ,..., kn–1 ≤ kn}. Отыскиваемый ключ сравнивается с центральным элементом множества. Если он меньше центрального, то поиск продолжается в левом подмножестве, в противном случае - в правом. Центральный элемент находится по формуле: Nэл = [n/2] + 1, где квадратные скобки обозначают, что от деления берется только целая часть (дробная часть не учитывается). В методе бинарного поиска анализируются только центральные элементы.

1.3. Фибоначчиев поиск

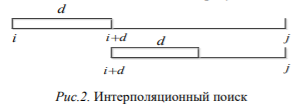
В этом поиске анализируются элементы, находящиеся в позициях, равных числам Фибоначчи. Числа Фибоначчи получаются по следующему правилу: каждое последующее число равно сумме двух предыдущих чисел.

1.4. Интерполяционный поиск

Исходное множество должно быть упорядоченно по возрастанию весов. Первоначальное сравнение осуществляется на расстоянии шага d, который определяется по формуле:



, где j - номер последнего рассматриваемого элемента; i - номер первого рассматриваемого элемента; K - отыскиваемый ключ; Ki K j , - значения ключей в i-й и j-й позициях; [ ] - целая часть числа. Идея метода заключается в следующем: шаг d меняется после каждого этапа по формуле, приведенной выше. Алгоритм заканчивает работу при d = 0. В этом случае осуществляется анализ соседних элементов, после чего делается окончательный вывод о результатах поиска (рис.2).



1.5. Поиск по бинарному дереву

Использование структуры бинарного дерева позволяет быстро вставлять и удалять записи и проводить эффективный поиск по таблице. Такая гибкость достигается добавлением в каждую запись двух полей для хранения ссылок.

1. Основанные на числовых свойствах ключей

2.1. Поиск по бору

Бор имеет вид m-арного дерева. Каждый узел уровня h представляет множество всех ключей, начинающихся с определенной последовательности из h литер. Узел определяет m-путевое разветвление в зависимости от (h + 1)-й литеры.

В первом узле записывается первая буква (цифра) ключа. Во втором узле к ней добавляется еще один символ и т.д. Если слово, начинающееся с определенной буквы (цифры), единственное, то оно сразу записывается в первом узле.

2.2. Поиск хешированием

В основе поиска лежит переход от исходного множества к множеству хеш-функций h(k). Хеш-функция, основанная на делении, имеет следующий вид: h(k) = k mod (m), где k - ключ; mod - целочисленный остаток от деления; m - целое число.

**Задание 1.1.**

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Стандартный обмен | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 9 | 1161 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Простой выбор | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 5 | 402 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Простая вставка | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 2 | 165 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Бинарная вставка | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 2 | 126 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Интерполяционная вставка | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 2 | 376 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Слияние | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 0 | 3 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Челночная | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 9 | 1051 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Шелла | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 0 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Хоара | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 0 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Пирамидальная | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 0 | 2 |

**Задание 1.2**

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Стандартный обмен | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 24 | 1804 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Простой выбор | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 5 | 340 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Простая вставка | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 2 | 204 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Бинарная вставка | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 1 | 91 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Слияние | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 1 | 19 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Челночная | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 6 | 742 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Шелла | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 0 | 9 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Хоара | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 0 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| Метод сортировки | Пирамидальная | | |
| Количество элементов исходного массива N | 100-300 | 1000-3000 | 10 000-30 000 |
| Время сортировки t, мc | 0 | 0 | 5 |

**Задание 1.3**

#include <iostream>

#include <time.h>

using namespace std;

void Shaker\_sort(int\* mas, int size)

{

bool sort\_or\_not = false;

int right = size - 1;

int left = 0;

while (sort\_or\_not == false) {

bool sort\_or\_not = true;

for (int i = left; i < right; i++) {

if (mas[i] > mas[i + 1]) {

swap(mas[i], mas[i + 1]);

sort\_or\_not = false;

}

}

if (sort\_or\_not) break;

right--;

for (int i = right; i > left; i--) {

if (mas[i] < mas[i - 1]) {

swap(mas[i], mas[i - 1]);

sort\_or\_not = false;

}

}

left++;

}

}int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

int N = 0;

cout << "Введите размер массива: ";

cin >> N;

int\* Array = new int[N];

cout << "Массив до сортировки: "<<endl;

for (int i = 0; i < N; i++)

{

Array[i] = rand() % 1000 - 500;

cout << Array[i] << " | ";

}

cout << endl;

Shaker\_sort(Array, N);

cout << "Массив после сортировки: " << endl;

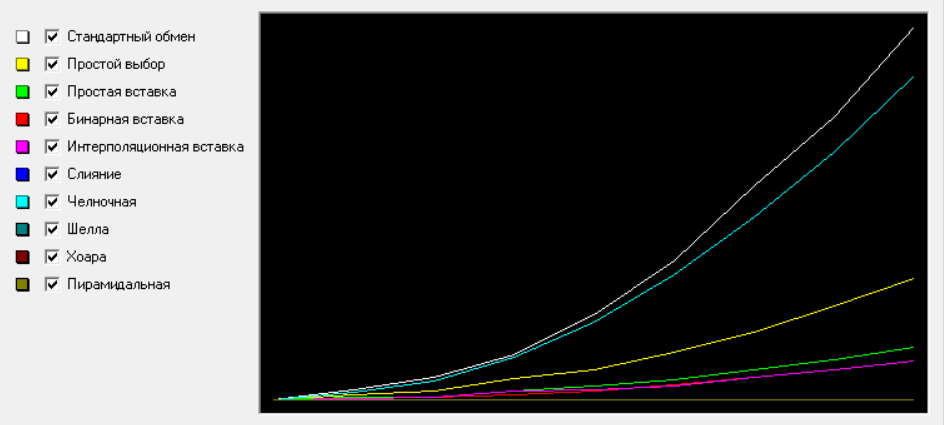
for (int i = 0; i < N; i++)

cout << Array[i] << " | ";

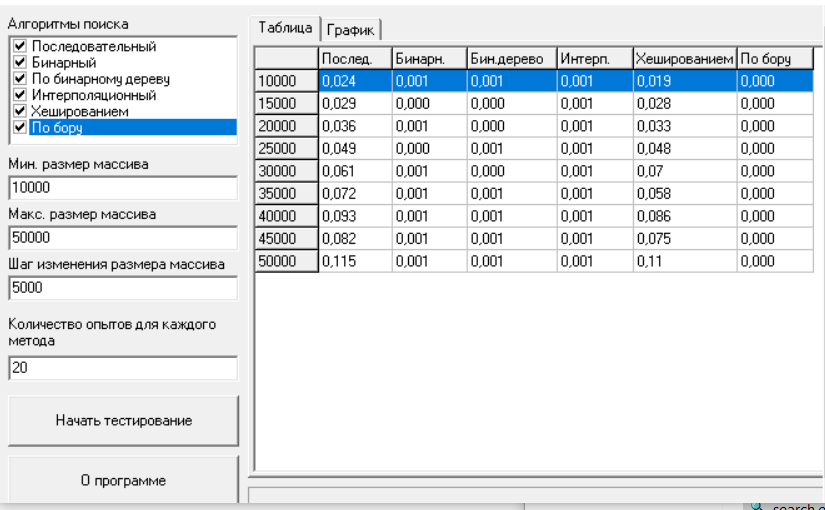
return 0;

}

**Задание 1.4**



**Задание 2.1**

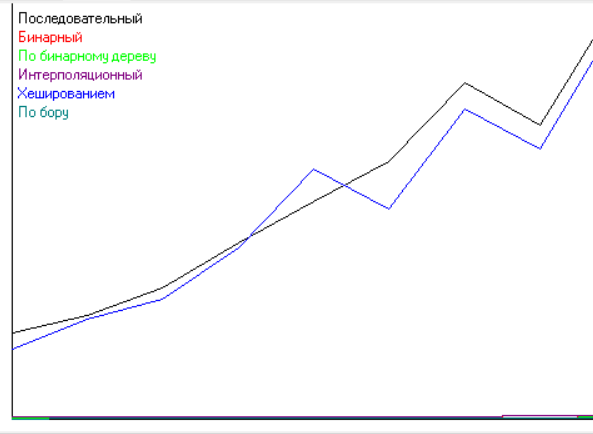


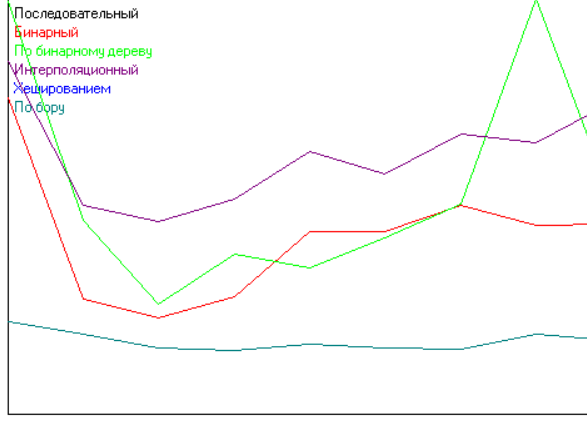
Последовательность – O(n).

Хеширование – O(n) – Худший; O(1) – средний расклад.

По бору- O(n).

Бинарный – log\_2 (n)





**Задание 2.4**

Алгоритм Рабина

#include <iostream>

#include <string>

#include <vector>

#include <string.h>

using namespace std;

int Rabin(string S1, string S2)

{

int index = -1;

for (int i = 0; i < S2.length() - S1.length() + 1; i++) {

for (int j = 0; j < S1.length(); j++)

if (S2[i + j] != S1[j]) break;

index = i;

}

return index;

}

int main()

{

setlocale(LC\_ALL, "ru");

string s1;

string s2;

cout << "Введите искомую строку: ";

cin >> s1;

cout << "Введите строку, в которой нужно найти первую: ";

cin >> s2;

int index = Rabin(s1, s2);

if (index == -1)

cout << "Строка не найдена" << endl;

else

cout << "Строка начинается с индекса " << index << endl;

return 0;

}

**Задание 2.5**

#include <iostream>

#include <time.h>

#include <ctime>

using namespace std;

void Shaker\_sort(int\* mas, int size)

{

bool sort\_or\_not = false;

int right = size - 1;

int left = 0;

while (sort\_or\_not == false) {

bool sort\_or\_not = true;

for (int i = left; i < right; i++) {

if (mas[i] > mas[i + 1]) {

swap(mas[i], mas[i + 1]);

sort\_or\_not = false;

}

}

if (sort\_or\_not) break;

right--;

for (int i = right; i > left; i--) {

if (mas[i] < mas[i - 1]) {

swap(mas[i], mas[i - 1]);

sort\_or\_not = false;

}

}

left++;

}

}

int fibonacci(int number) {

if (number == 0)

return 0; // базовый случай (условие завершения)

if (number == 1)

return 1; // базовый случай (условие завершения)

return fibonacci(number - 1) + fibonacci(number - 2);

}

void fibonacci\_find(int\* mass, int size, int poisk)

{

if (size > 1)

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

if (poisk <= mass[fibonacci(i)]) {

if (poisk == mass[fibonacci(i)]) {

cout << "namber is located";

break;

}

int k = 0;

for (int j = fibonacci(i - 1); j < fibonacci(i); j++) {

mass[k] = mass[j];

k++;

}

fibonacci\_find(mass, k, poisk);

}

}

else

if (poisk == mass[0])

cout << "namber is located";

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

int N = 0;

cout << "Введите размер массива: ";

cin >> N;

int\* Array = new int[N];

cout << "Массив до сортировки: " << endl;

for (int i = 0; i < N; i++)

{

Array[i] = rand() % 1000 - 500;

cout << Array[i] << " | ";

}

cout << endl;

unsigned int start\_time = clock();

Shaker\_sort(Array, N);

unsigned int end\_time = clock();

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

cout << "Массив после сортировки: " << endl;

for (int i = 0; i < N; i++)

cout << Array[i] << " | ";

cout << endl;

int find = 0;

cout << "Введите ключ: ";

cin >> find;

cout << "Время работы сортировки: " << search\_time << "мс" << endl;

fibonacci\_find(Array, N, find);

return 0;

}