Бюджетное учреждение высшего образования   
Ханты-Мансийского автономного округа   
«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**Отчет**

по лабораторной работе № 3

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы 609-21,

Шумилов И. Д.

Принял: старший преподаватель кафедры АиКС

Назаров Е.В.

Сургут

2024 г.

**Цель работы:** изучить базовые алгоритмы поиска, исследовать их свойства, закрепить навыки структурного программирования.

**Общее задание на лабораторную работу:**

1. Разработать функции, реализующие три алгоритма поиска (прыжками обязательно, еще два – по выбору).
2. Исследовать алгоритмы поиска: построить и сравнить зависимости среднего количества сравнений в случаях успешного и неуспешного поиска для заданных алгоритмов.
3. Найти зависимость оптимальной величины «прыжка» от размера последовательности.
4. Составить отчет, в котором привести графики полученных зависимостей, анализ свойств алгоритмов и выводы по работе.

**Индивидуальное задание на лабораторную работу:**

1. Поиск прыжками (одноуровневая и двухуровневая)
2. Поиск Фибоначчи
3. Интерполяционный поиск

*Листинг 1. Разработанные функции поиска:*

*// Одноуровневый поиск прыжками*

*int oneLvlJumpSearch(int arr[], int n, int x, int step) {*

*int prev = 0;*

*int fixStep = step;*

*while (compN++, arr[std::min(step, n) - 1] < x) {*

*prev = step;*

*step += fixStep;*

*if (prev >= n)*

*return -1;*

*}*

*while (compN++, arr[prev] < x) {*

*prev++;*

*if (prev == std::min(step, n))*

*return -1;*

*}*

*if (compN++, arr[prev] == x)*

*return prev;*

*return -1;*

*}*

*// Двухуровневый поиск прыжками*

*int twoLvlJumpSearch(int arr[], int n, int x) {*

*int step1 = (int)sqrt(n), step2 = (int)sqrt(step1);*

*int prev1 = 0, prev2 = step1;*

*while (compN++, arr[std::min(prev2, n) - 1] < x) {*

*prev1 = prev2;*

*prev2 += step1;*

*if (prev1 >= n)*

*return -1;*

*}*

*prev2 = prev1 + step2;*

*int end = prev1 + step1;*

*while (compN++, arr[std::min(prev2, end) - 1] < x) {*

*prev1 = prev2;*

*prev2 += step2;*

*if (prev1 >= end)*

*return -1;*

*}*

*while (compN++, arr[prev1] < x) {*

*prev1++;*

*if (prev1 == std::min(prev2, end))*

*return -1;*

*}*

*if (compN++, arr[prev1] == x)*

*return prev1;*

*return -1;*

*}*

*// Поиск Фибоначчи*

*int fibonacciSearch(int arr[], int n, int x) {*

*int f2 = 0;*

*int f1 = 1;*

*int f = f2 + f1;*

*while (f < n) {*

*f2 = f1;*

*f1 = f;*

*f = f2 + f1;*

*}*

*int offset = -1;*

*while (f > 1) {*

*int i = std::min(offset + f2, n - 1);*

*if (compN++, arr[i] < x) {*

*f = f1;*

*f1 = f2;*

*f2 = f - f1;*

*offset = i;*

*} else if (compN++, arr[i] > x) {*

*f = f2;*

*f1 = f1 - f2;*

*f2 = f - f1;*

*} else*

*return i;*

*}*

*if (compN++, f1 && arr[offset + 1] == x)*

*return offset + 1;*

*return -1;*

*}*

*// Интерполяционный поиск*

*int interpolationSearch(int arr[], int n, int x) {*

*int pos, low = 0, high = n - 1;*

*while (low <= high && x >= arr[low] && x <= arr[high] && high != low) {*

*pos = low + (((double)(high - low) / (arr[high] - arr[low])) \* (x - arr[low]));*

*if (compN++, arr[pos] == x)*

*return pos;*

*else {*

*if (compN++, arr[pos] < x)*

*low = pos + 1;*

*else*

*high = pos - 1;*

*}*

*}*

*return -1;*

*}*

**Пояснения к программе**

Помимо необходимых функций поиска была разработана функция заполнения массива случайными четными числами (чтобы в будущем было проще делать неуспешные поиски, просто пытаясь найти нечетное число) и была взята поразрядная сортировка, разработанная в предыдущей лабораторной работе, для сортировки сгенерированного случайного массива.

**Исследование алгоритмов**

Были проведены замеры среднего количества операций для всех видов поиска при успешном и неудачном поисках для разных размеров последовательностей.

Таблица 1 – Зависимости среднего количества сравнений в случаях успешного и неуспешного поиска

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Размер массива | Среднее количество сравнений алгоритма поиска (S(n)) | | | | | | | |
| Прыжками (одноуровневая) | | Прыжками (двухуровневая) | | Фибоначчи | | Интерполяционная | |
| Успех | Неудача | Успех | Неудача | Успех | Неудача | Успех | Неудача |
| 10000 | 99 | 94 | 61 | 60 | 17 | 20 | 5 | 6 |
| 20000 | 149 | 143 | 85 | 80 | 19 | 22 | 5 | 7 |
| 30000 | 176 | 167 | 101 | 101 | 19 | 23 | 5 | 7 |
| 40000 | 197 | 190 | 115 | 113 | 19 | 23 | 5 | 7 |
| 50000 | 229 | 226 | 131 | 126 | 19 | 24 | 5 | 7 |
| 60000 | 246 | 241 | 140 | 136 | 19 | 24 | 4 | 7 |
| 70000 | 258 | 250 | 158 | 148 | 19 | 24 | 4 | 7 |
| 80000 | 298 | 285 | 163 | 155 | 20 | 25 | 4 | 7 |
| 90000 | 303 | 298 | 169 | 165 | 20 | 25 | 4 | 8 |
| 100000 | 313 | 308 | 174 | 170 | 20 | 25 | 4 | 8 |

Рисунок 1 – График зависимости количества операций от размера в поисках прыжками

­

Рисунок 2 – График зависимости количества операций от размера в поисках Фибоначчи и интерполяционном

По графикам видно, что при поиске прыжками количество операций сравнения при успешном и неуспешном поиске отличается очень незначительно. Однако при поиске Фибоначчи или интерполяционном поиске при успешном поиске количество операций значительно меньше чем при неуспешном. Также следует отметить, что при поисках прыжками зависимость S(n) приближена к графику S(), при этом двухуровневый поиск значительно быстрее одноуровневого, при поиске Фибоначчи график зависимости S(n) близится к графику S(log n), а при интерполяционном к графику S(log log n).

**Определение зависимости оптимальной величины прыжка от размера**

Рисунок 3 – График зависимости размера последовательности от величины прыжка

Экспериментальным путем были получены данные о количестве операций. Проведя анализ графика зависимости на рис. 3 можно прийти к выводу, что оптимальная величина прыжка близится к значению, значения меньше этого занимают значительно больше операций, значения больше постепенно начинают требовать все больше и больше операций.

**Сравнительная оценка реализованных алгоритмов**

Таблица 2 – Сравнительная оценка реализованных алгоритмов

|  |  |
| --- | --- |
| Алгоритм поиска | Средняя сложность |
| Поиск прыжками (любой) | S() |
| Поиск Фибоначчи | S(log n) |
| Интерполяционный поиск | S(log log n) |

**Вывод:** были изучены алгоритмы одноуровневого и двухуровневого поиска прыжками, поиска Фибоначчи, интерполяционного поиска и были исследованы их свойства.