БУ ВО Ханты-Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**ОТЧЁТ**

по лабораторной работе № 3

по дисциплине: Алгоритмы и структуры данных

тема: «Базовые алгоритмы поиска»

Выполнила: студентка группы №609-11,

Пашина Илона Руслановна

Принял: доцент кафедры АиКС

Даниленко И.Н.

Сургут

2023г.

**Цель работы:**

Изучить базовые алгоритмы поиска, исследовать их свойства, закрепить навыки структурного программирования.

**Задание:**

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

**Ход работы**

|  |  |  |
| --- | --- | --- |
|  | успешный | неуспешный |
| последовательный | сравнения | сравнения |
| 1 000 000 | 645243 | 1 000 000 |
| 2 000 000 | 1361037 | 2 000 000 |
| 3 000 000 | 1392627 | 3 000 000 |
| 4 000 000 | 2222148 | 4 000 000 |
| 5 000 000 | 2080509 | 5 000 000 |
| 6 000 000 | 3437636 | 6 000 000 |
| 7 000 000 | 2857765 | 7 000 000 |
| 8 000 000 | 4126004 | 8 000 000 |
| 9 000 000 | 4260702 | 9 000 000 |
| 10 000 000 | 5380592 | 10 000 000 |

|  |  |  |
| --- | --- | --- |
|  | успешный | неуспешный |
| Прыжками | сравнения | сравнения |
| 1 000 000 | 550 | 1026 |
| 2 000 000 | 707 | 1439 |
| 3 000 000 | 882 | 1749 |
| 4 000 000 | 847 | 2013 |
| 5 000 000 | 1012 | 2253 |
| 6 000 000 | 1371 | 2462 |
| 7 000 000 | 1506 | 2658 |
| 8 000 000 | 1422 | 2838 |
| 9 000 000 | 1252 | 3013 |
| 10 000 000 | 1461 | 3174 |

|  |  |  |
| --- | --- | --- |
|  | успешный | неуспешный |
| бинарный | сравнения | сравнения |
| 1 000 000 | 19 | 20 |
| 2 000 000 | 20 | 21 |
| 3 000 000 | 20 | 21 |
| 4 000 000 | 20 | 21 |
| 5 000 000 | 21 | 22 |
| 6 000 000 | 21 | 22 |
| 7 000 000 | 21 | 22 |
| 8 000 000 | 21 | 22 |
| 9 000 000 | 22 | 23 |
| 10 000 000 | 22 | 23 |

**Оптимальная величина «прыжка» от размера последовательности.**

Так как нам нужно найти оптимальную величину «прыжка» от размера массива, то возьмём за основу последовательный поиск.

В среднем случае успешный последовательный поиск имеет O(N/2) сравнений.

Рассмотрим 2 случая:

1. Шаг равен 1
2. Шаг равен размеру массива

Тогда в 1 случае получим: C(N, 1) = O(N/2)

А во 2 случае C(N, N) = O(N/2).

Интуитивно понятно, что между 1 и N есть такой шаг, при котором поиск будет осуществляться быстрее.

Найдём это значение:

Возьмём за размер прыжка h, тогда

Где – количество сравнений при успешном поиске в зависимости от прыжка, а

Найдём экстремумы данной функции:

Тогда получим, что самым оптимальным шагом будет

**При успешном**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Худший случай** | **Средний случай** | **Лучший случай** |
| Последовательный | O(N) | O(N/2) | O(1) |
| «Прыжками» | O( | O( | O( |
| Бинарный |  |  |  |

**При не успешном**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Худший случай** | **Средний случай** | **Лучший случай** |
| Последовательный | O(N) | O(N) | O(N) |
| «Прыжками» | O( | O( | O( |
| Бинарный |  |  |  |

**Вывод:**

В ходе лабораторной работы были изучены базовые алгоритмы поиска, исследованы их свойства, закреплены навыки структурного программирования. Самым медленным оказался последовательный поиск, а самым быстрым бинарный. Поиск «прыжками» оказался быстрее последовательного, но медленнее бинарного. Однако последовательный поиск может работать как с отсортированным массивом, так и с не отсортированным, в отличии от двух других («прыжками» и бинарный).

**Листинг**

#include <iostream>

#include <cstdlib>

#include <cmath>

#include <windows.h>

#include <ctime>

using namespace std;

void FillUp(int \*A, int N, int min, int max);

void print\_array(int \*A, int N);

int SequentialSearch(int \*A, int N, int key);

int BinarySearch(int \*A, int N, int key);

int JumpSearch(int \*A, int N, int key);

int SequentialSearch(int \*A, int begin, int end, int key);

int Cmp = 0;

int EqualCmp(int a, int b){

Cmp++;

return (a == b);

}

int LessCmp(int a, int b){

Cmp++;

return (a < b);

}

int MoreCmp(int a, int b){

// Cmp++;

return (a > b);

}

int LessEqualCmp(int a, int b){

// Cmp++;

return (a <= b);

}

int MoreEqualCmp(int a, int b){

Cmp++;

return (a >= b);

}

int main(){

int min = 0, max = 100000\*2, N = max/2; //10 000 - 100 000

int \*A;

A = new int[N];

srand(time(NULL));

int all = 0;

int find = 0;

FillUp(A, N, min, max);

for(int i = 0; i < 10; i++){

FillUp(A, N, min, max);

find = ((rand() << 15) | rand())%Nn;

cout << SequentialSearch(A, N, A[find])<< ' ' <<endl;

all += Cmp;

}

cout << all/10 << endl;

Cmp = 0;

return 0;

}

int SequentialSearch(int \*A, int N, int key){

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

if(EqualCmp(A[i], key)) return i;

}

return -1;

}

int BinarySearch(int \*A, int N, int key){

int low = 0;

int high = N-1;

int mid;

while(LessEqualCmp(low,high)){

mid = (low+high)/2;

if(EqualCmp(A[mid], key)) return mid;

if(MoreCmp(A[mid], key)) high = mid-1;

else low = mid+1;

}

return -1;

}

int JumpSearch(int \*A, int N, int key){

int JumpStep = sqrt(N);

int PrevStep = 0;

while(LessCmp(A[min(JumpStep, N)-1], key)){

PrevStep = JumpStep;

JumpStep += sqrt(N);

if(MoreEqualCmp(PrevStep, N)) return -1;

}

SequentialSearch(A, PrevStep, (JumpStep > N)?N:JumpStep , key);

}

int SequentialSearch(int \*A, int begin, int end, int key){

for(int i = begin; i < end; i++){

if(EqualCmp(A[i], key)) return i;

}

return -1;

}

void FillUp(int \*A, int N, int min, int max){

double delta = ((double)(max - min))/N;

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

A[i] = delta\*i+min;

}

}

void print\_array(int \*A, int N){

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

cout << A[i] << ' ';

}

cout << endl;

}