**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ**

**Самостійна робота №**2**​**

з дисципліни «Теорія алгоритмів»

Тема роботи: «Пошукові алгоритми» **Варіант №14**

Виконав студент

групи КН-22

Пашковський П.В.

Викладач:

Доманецька І. М.

**Київ – 2019**

**Завдання №1**

**Реалізувати пошук у впорядкованому масиві.**

**Варіант 1. Бінарний пошук елемента масиву з використанням рекурсії.**

Реалізація алгоритмів до завдання 1 див. ДОДАТОК 1.

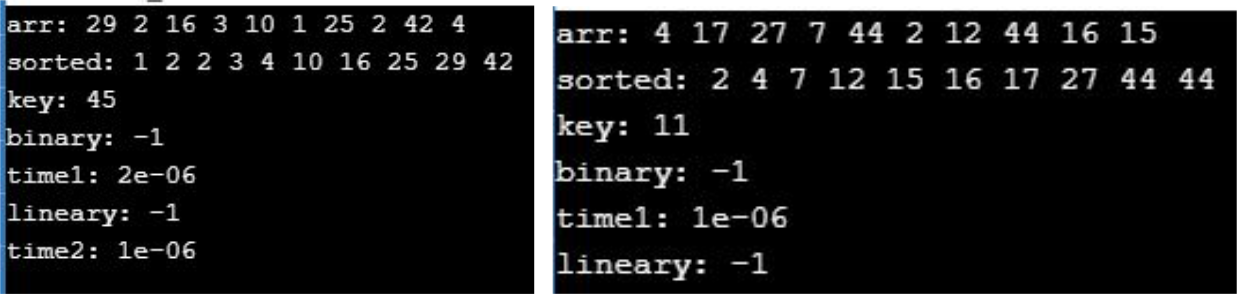
1.1. Складність алгоритму бінарного пошуку O(log n).

1.2. Порівняльна таблиця підрахунку операцій порівняння для алгоритмів бінарного та послідовного пошуку

Таблиця 1. Порівняння роботи алгоритмів пошуку елементу в масиві

|  |  |  |  |
| --- | --- | --- | --- |
| N | 10 000 | 100 000 | 1 000 000 |
|  |  |  |  |
| t бінарний | 2e-06 | 6.6e-9 | 1.3e-13 |
|  |  |  |  |
| t послідовний | 2.9e-05 | 9e-9 | 3.4e-15 |
|  |  |  |  |

1.3.



a) b)

Рис.1 Робота алгоритму за умови відсутності шуканого елементу

1. Шуканий елемент більше останнього;
2. Відсутній шуканий елемент із середини масиву;

Висновок: метод бінарного пошуку більш ефективний у великих масивах.

**Завдання №2**

**Реалізувати пошук у невпорядкованому масиві**

1.1. Оцінити складність алгоритму (аналітичний підрахунок);

1.2. Оцінити час роботи алгоритму на масивах розмірності 10 000, 100 000,

1 000 000 елементів;

1.3. Виконати порівняльний аналіз роботи алгоритму, визначеного вашим варіантом, і алгоритму, який передбачає попереднє сортування масиву методом «бульбашки» і вибору відповідного серединного елементу.

**Варіант 2. Пошук медіани масиву з використанням опорного елементу**.**​**

1.1. O(n^2) - вкладений цикл

1.2. Результати роботи. Програмний код методів див. ДОДАТОК 3

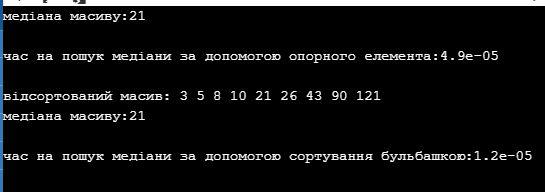


Рис.2 Робота алгоритмів пошуку медіани

**Завдання №3**

**Пошук рішення рівняння**

**Варіант 3. Метод дихотомії.**

**Варіант 7. f(x) = xlog10(x) [0.1; 10].**

Задана монотонно зростаюча (спадна) функція f(x) на проміжку [a; b].

Рівняння f(x) = 0 має єдиний розв’язок.

1.1. Пошук рішення рівняння f(x) = 0 заданої точності;

1.2. Оцінити час роботи алгоритму для різних значень точності шуканого рішення (ε = 1E-4, ε = 1E-8, ε = 1E-16).

1.3. Програмний код методу хорд див. ДОДАТОК 3

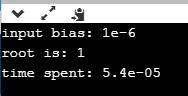


Рис.3 Робота алгоритму пошуку кореня рівняння методом дихотомії Таблиця 3. Часові характеристики алгоритму для різної точності рішення рівняння

|  |  |  |  |
| --- | --- | --- | --- |
| N | 1E-4 | 1E-8 | 1E-16 |
|  |  |  |  |
| t | 3.8e-5 sec | 5.3e-5 sec | 8.9e-5 |
|  |  |  |  |

Висновок: збільшення точності роботи алгоритму призводить до збільшення часу виконання коду.

Завдання 1. Реалізація алгоритмів Лінійного та Бінарного пошуку

ДОДАТОК 1.

#include <iostream>

#include <cmath>

#include <iomanip>

#include <random>

#include <ctime>

using namespace std;

const int N=1000;

int arr[N];

int bubble(int N){

int tmp;

int i,j;

for(j = 0; j < N - 1; j++){

for(i = 0; i < N - 1; i++){

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

tmp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = tmp;

}

}

}

cout << "sorted: ";

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

{

cout << arr[i] << " ";

}

cout << endl;

}

int binarysearch(int arr[], int left, int right, int key)

{

int m = (left + right) / 2;

if (m==left || m==right) return -1;

else if (arr[m] == key) return m;

else if (arr[m] > key) { right = m; binarysearch(arr, left, right, key); } else if (arr[m] < key) { left = m; binarysearch(arr, left, right, key); }

}

int linearysearch(int arr[], int key, int N){

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

if (arr[i] == key)

return i;

}

return -1;

}

int main(){

int find1;

int find2;

srand(time(0));

cout <<"arr: ";

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

arr[i] = rand()%50;

cout << arr[i]<<" ";

}

cout<<endl;

bubble(N);

int key;

cout <<"key: ";

cin >> key;

float time1 = clock();

find1 = binarysearch(arr,0,N,key);

float time2 = clock();

float time3 = clock();

find2 = linearysearch(arr,key,N);

float time4 = clock();

cout<<"binary: " << find1<<endl;

cout<<"time1: "<<float(time2-time1)/CLOCKS\_PER\_SEC<<endl;

cout<<"lineary: "<<find2<<endl;

cout<<"time2: "<<float(time4-time3)/CLOCKS\_PER\_SEC<<endl; return 0;

}

Завдання 2. Реалізація алгоритмів пошуку медіани

ДОДАТОК 2.

#include <iostream>

#include <climits>

#include <vector>

#include <ctime>

using namespace std;

const int n=9;

int bubble(vector<int> &mas){

int tmp;

int i,j;

for(j = 0; j < n - 1; j++){

for(i = 0; i < n - 1; i++){

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

tmp = mas[i];

mas[i] = mas[i + 1];

mas[i + 1] = tmp;

}

}

}

cout << "відсортований масив: ";

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

{

cout << mas[i] << " ";

}

cout<<endl;

int med=mas[n/2];

cout<<"медіана масиву:"<<med<<endl;

cout << endl;

}

int separation(vector<int> &mas, int l, int r) {

if (l!=r)

swap(mas[l + rand() % (r - l)], mas[r]);

int x = mas[r];

int i = l-1;

for (int j = l; j <= r; j++) {

if (mas[j] <= x)

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

}

return i;

}

int medianasearch(vector<int> mas, int n) {

int l = 0, r = mas.size() - 1;

for(;;) {

int pos = separation(mas,l,r);

if (pos < n)

l = pos + 1;

else if (pos > n)

r = pos - 1;

else return mas[n];

}

}

int main()

{

vector<int> arr = { 3, 5, 8, 90, 10, 26, 121, 21, 43};

int k = n/2;

float t1=clock();

cout << "медіана масиву:" << medianasearch(arr, k)<<endl<<endl;

float t2=clock();

cout<<"час на пошук медіани за допомогою опорного елемента:"<<float(t2-t1)/CLOCKS\_PER\_SEC<<endl;

cout <<endl;

float t3=clock();

bubble(arr);

float t4=clock();

cout<<"час на пошук медіани за допомогою сортування бульбашкою:"<<float(t4-t3)/CLOCKS\_PER\_SEC<<endl;

return 0;

}

ДОДАТОК 3.

Завдання 3. Реалізація алгоритмів рішення рівняння методом дихотомії.

#include<cmath>

#include<iostream>

#include<iomanip>

#include<ctime>

using namespace std;

double func(double x){

return x\*log10(x);

}

int main(){

double a, b, c, eps;

a = 0.1;

b = 10;

cout << "input bias: ";

cin >> eps ;

float time1 = clock();

while (b - a > eps){

c = (a + b) / 2;

if(func(b) \* func(c) < 0)

a = c;

else

b = c;

}

float time2 = clock();

cout <<"root is: " << (a + b) / 2 << endl << "time spent: " << setprecision(3) << float(time2-time1)/CLOCKS\_PER\_SEC;

return 0;

}