МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра «Стратегічне управління»

Звіт з лабораторної роботи №5

«Алгоритми пошуку»

з дисципліни

"Алгоритми та структури даних"

Варіант № 6

Перевірив:

ст. викл. каф. стратегічного

управління Мошко Є.О.

Виконав: Колій Дмитро

ст. гр. КН-320б

Харків – 2022

Мета: ознайомитися із основними алгоритмами пошуку даних та особливостями їх програмної реалізації. Набути практичних навичок по роботі зі алгоритмами пошуку даних.

Індивідуальне завдання:

1. Визначити масив, в якому буде виконуватися пошук. Використовувати масив, створений в роботі №4.

2. Розробити функції лінійного пошуку в масиві, бінарного пошуку в масиві. Для забезпечення бінарного пошуку в масиві використовувати найбільш ефективний алгоритм сортування, визначений при виконанні лабораторної роботі №4:

− ввести інформацію в масив з файлу;

− виконати лінійний пошук в масиві;

− виконати пошук з бар'єром;

− упорядкувати елементи масиву функцією сортування і виконати бінарний пошук.

3. Розробити функції для пошуку підрядка в рядку:

− виконати прямий пошук підрядка;

− виконати алгоритм Кнута-Морріса-Пратта;

− виконати алгоритм Бойера-Мура;

− виконати алгоритм Рабина-Карпа.

4. Дослідити складність алгоритмів. Провести асимптотичний аналіз алгоритмів пошуку та зробити висновки.

− для порівняння алгоритмів пошуку виконати наступні кроки;

− створити таблицю асимптотичних оцінок трудомісткості алгоритмів в кращому, середньому, гіршому випадках;

− розставити лічильники операцій у функціях пошуку;

− провести експеримент, визначити середню кількість операцій для різних алгоритмів, побудувати графіки;

− створити таблиці і представити графіки експериментальних оцінок алгоритмів.

Лінійний пошук

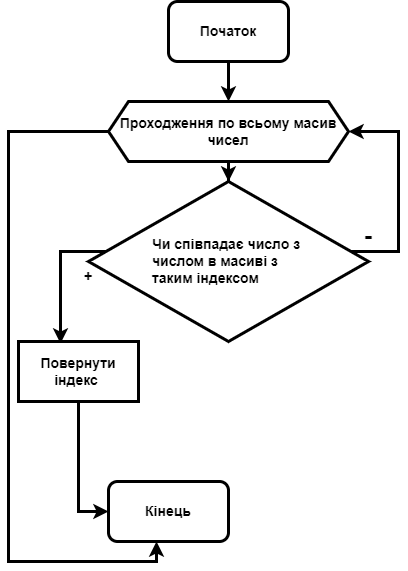


Рисунок 1 – Блок-схема лінійного пошуку

Програмна реалізація лінійного пошуку :

int lin\_search\_arr(vector<int> mass, int key, int& col\_iterachiy) {

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

col\_iterachiy++;

if (mass[i] == key) {

col\_iterachiy += 2;

return i;

}

col\_iterachiy++;

}

col\_iterachiy++;

return -1;

}

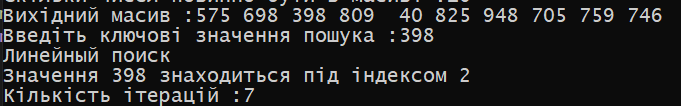


Рисунок 2 – Результат лінійного пошука

Бар’єрний пошук



Рисунок 3 – Блок-схема бар’єрного пошуку

Програмна реалізація бар’єрного пошуку:

int barrier\_search\_arr(vector<int> mass, int key, int& col\_iterachiy) {

mass.push\_back(key);

col\_iterachiy++;

int i(0);

while (mass[i] != key) {

col\_iterachiy += 2;

i++;

}

col\_iterachiy++;

return i < mass.size() - 1 ? i : -1;

}

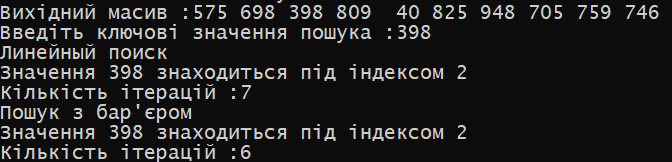


Рисунок 4 – Результат бар’єрного пошуку

Бінарний пошук

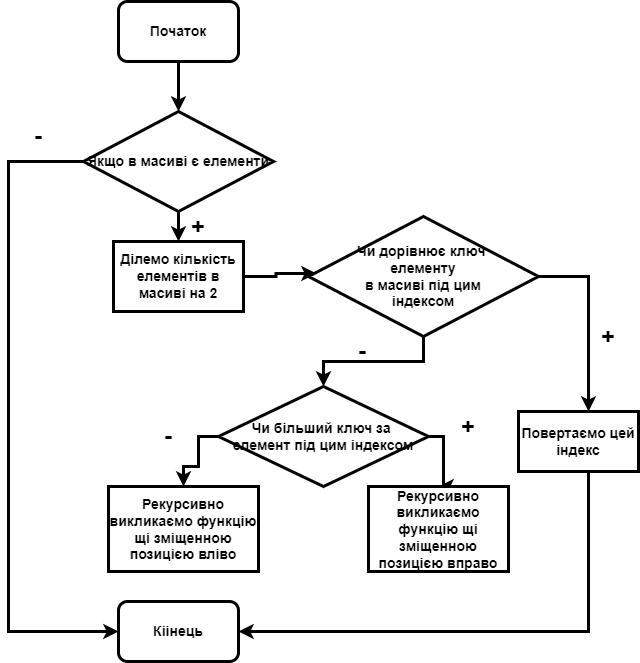


Рисунок 5 – Блок-схема бінарного пошуку

Програмна реалізація бінарного пошуку:

if (r >= l) {

int mid = l + (r - l) / 2;

col\_iterachiy += 2;

if (arr[mid] == x) {

col\_iterachiy += 2;

return mid;

}

col\_iterachiy++;

if (arr[mid] > x) {

col\_iterachiy += 2;

return binar\_search\_arr(arr, l, mid - 1, x);

}

col\_iterachiy += 2;

return binar\_search\_arr(arr, mid + 1, r, x);

}

col\_iterachiy++;

return -1;

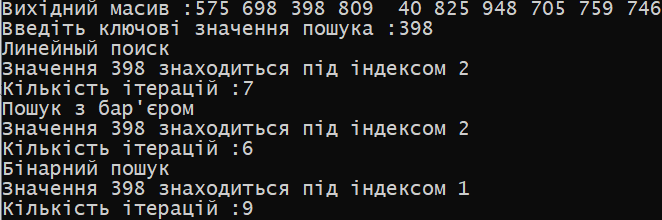


Рисунок 6 – Результат бінарного пошуку

Прямий пошук



Рисунок 7 – Блок-схема прямого пошуку підрядка

Програмна реалізація прямого пошуку підрядка:

int pramuy\_search(string osn\_str, string key\_str) {

int counetr(0);

int col\_iterachiy(0);

for (size\_t i = 0; i < osn\_str.length(); i++)

{

col\_iterachiy++;

for (size\_t j = i; j < key\_str.length() + i; j++)

{

col\_iterachiy += 2;

if (osn\_str[j] == key\_str[counetr++]) {

col\_iterachiy++;

if (counetr == key\_str.length()) {

cout << "на позиції " << j - key\_str.length() + 1 << endl;

}

}

else {

col\_iterachiy++;

counetr = 0;

break;

}

}

}

col\_iterachiy++;

return col\_iterachiy;

}

Бойер-Мур

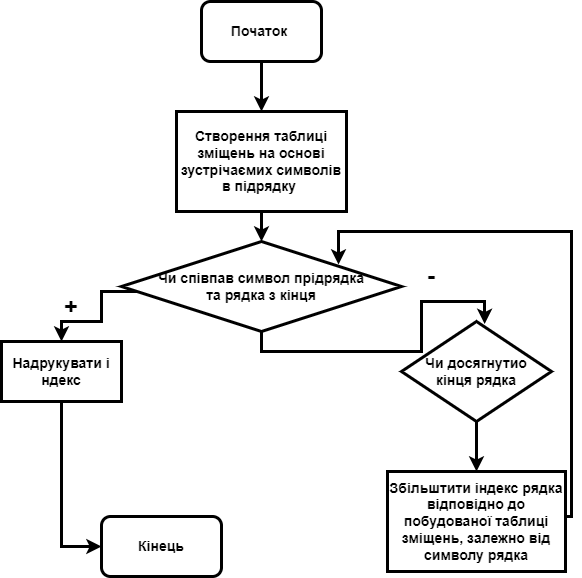


Рисунок 8 – Блок-схема алгоритму Бойера-Мура

Програмна реалізація пошуку Бойера-Мура підрядка:

int search\_Boyler(string txt, string pat)

{

int col\_iter(0);

int m = pat.size();

int n = txt.size();

col\_iter += 2;

int badchar[NO\_OF\_CHARS];

badCharHeuristic(pat, m, col\_iter, badchar);

int s = 0;

col\_iter++;

while (s <= (n - m))

{

col\_iter++;

int j = m - 1;

col\_iter++;

while (j >= 0 && pat[j] == txt[s + j]) {

col\_iter += 2;

j--;

}

if (j < 0)

{

col\_iter += 2;

cout << "на позиції " << s << endl;

s += (s + m < n) ? m - badchar[txt[s + m]] : 1;

}

else {

col\_iter += 2;

s += max(1, j - badchar[txt[s + j]]);

}

}

col\_iter++;

return col\_iter;

}

Кнут-Моріс-Пратт



Рисунок 9 – Блок-схема алгоритму Кнута-Моріса-Пратта

Програмна реалізація пошуку Кнута-Моріса-Пратта підрядка:

int KMPSearch(string pat, string txt)

{

int col\_iter(0);

int M = pat.length();

int K = txt.length();

col\_iter += 2;

int\* lps = new int[M];

col\_iter++;

col\_iter += computeLPSArray(pat, M, lps);

int i = 0;

int j = 0;

col\_iter += 2;

while (i < K) {

col\_iter++;

if (pat[j] == txt[i]) {

col\_iter += 3;

j++;

i++;

}

if (j == M) {

cout << "на позиції " << i - j << endl;

col\_iter += 2;

j = lps[j - 1];

}

else if (i < K && pat[j] != txt[i]) {

col\_iter++;

if (j != 0) {

col\_iter += 2;

j = lps[j - 1];

}

else {

i = i + 1;

col\_iter += 2;

}

}

}

col\_iter++;

return col\_iter;

}

Рабин-Карп



Рисунок 10 – Блок-схема алгоритму Рабина-Карпа

Програмна реалізація пошуку Рабина-Карпа підрядка:

int Robina\_Karpa(string pat, string txt, int q)

{

int col\_iter(0);

int M = pat.length();

int K = txt.length();

int i, j;

int p = 0;

int t = 0;

int h = 1;

col\_iter += 5;

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

h = (h \* O) % q;

col\_iter += 2;

}

for (i = 0; i < M; i++)

{

p = (O \* p + pat[i]) % q;

t = (O \* t + txt[i]) % q;

col\_iter += 3;

}

for (i = 0; i <= K - M; i++)

{

col\_iter++;

if (p == t)

{

col\_iter += 2;

bool flag = true;

for (j = 0; j < M; j++)

{

col\_iter++;

if (txt[i + j] != pat[j])

{

col\_iter += 2;

flag = false;

break;

}

}

if (j == M) {

cout << "на позиції " << i << endl;

col\_iter++;

}

}

if (i < K - M)

{

col\_iter++;

t = (O \* (t - txt[i] \* h) + txt[i + M]) % q;

col\_iter++;

if (t < 0) {

t = (t + q);

col\_iter += 2;

}

}

}

col\_iter++;

return col\_iter;

}

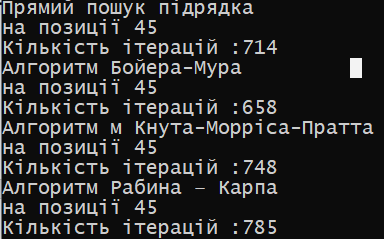


Рисунок 11 – Результати алгоритмів пошуку підрядка

Асимптотична оцінка трудомісткості алгоритмів



Кількість виконаних операцій під час пошуку:



Складність алгоритмів пошуку у рядку:



Кількість операцій під час пошуку підрядка в рядку:





Висновок

Під час виконання цієї лабораторної роботи я ознайомився із основними алгоритмами пошуку даних та особливостями їх програмної реалізації. Набув практичних навичок по роботі зі алгоритмами пошуку даних.

Прямий пошук – алгоритм пошуку підрядка в рядку, головною ідеєю якого є поелементне порівняння підрядка та основного рядка. Якщо символи не однакові, то підрядок зсувається на один символ та продовжує порівнюватися до кінця основного рядка.

Алгоритм Кнута, Морріса, Пратта – це алгоритм пошук підрядка в рядку, в якому зсув підрядка відбувається не на 1, як в лінійному, а на вирахувану кількість символів, яке зберігається в масиві.

Алгоритм Боєра-Мура – це алгоритм пошуку підрядка в рядку, при якому спочатку будується таблиця зсувів для підрядка перевірка починається з останнього символу підрядка після суміщення початку рядка і підрядка.

Алгоритм Рабина-Карпа – це алгоритм пошуку підрядка, який базується на хешуванні підрядка та порівняння його з хешом частини основного рядка, в разі збігу відбувається поелементне порівняння елементів цієї частини збігу.