**Київський національний університет імені Тараса Шевченка**

**Факультет комп’ютерних наук та кібернетики**

**Кафедра інформаційних систем**

**Алгоритми та складність**

**Лабораторна робота № 8**

**Звіт**

**Виконав:**

студент групи К-29

Карсім Олег Сергійович

**Київ-2018**

**Зміст завдання :**

Реалізувати алгоритми пошуку зразка в текстовому рядку: наївний, Хорспула, Боєра-Мура, КПМ та Рабіна-Карпа і порівняти їх ефективність. Виконати пошук зразків різної довжини: випадкового бінарного зразка у випадковому бінарному тексті та випадкового слова у природному тексті на цій мові.

**Алгоритми та основні модулі програми :**

1. Алгоритм Кнута-Морріса-Пратта — один із алгоритмів пошуку рядка, що шукає входження слова W у рядку S, використовуючи просте спостереження, що коли відбувається невідповідність, то слово містить у собі достатньо інформації для того, щоб визначити, де наступне входження може початися, таким чином пропускаючи кількаразову перевірку попередньо порівняних символів.

Алгоритм, що винайшли Дональд Кнут та Вон Пратт, а також незалежно від них Джеймс Морріс.

Часова асимптотична складність алгоритму становить O(N+M), де N — довжина слова W, M — довжина рядка S.

* void preKMP(char\* pattern, int f[]) - створення префікс-функції
* bool KMP(char\* pattern, char\* target) — реалізація алгоритму
* void KMPdemo() - функція для демонстрації коректності результатів алгоритму в консольному вікні

1. Наївний алгоритм

Дано текст t[0 … n – 1] та зразок p[0 … m – 1] такі, що n>=m та елементи даних рядків — символи деякого скінченного алфавіту А.

Потрібно перевірити, чи входить зразок р у рядок t.

Визначення: будемо казати, що зразок р міститься у тексті t із зсувом s, якщо 0 <= s <= n – m та t[s … s + m – 1] = p. Якщо зразок p міститься у рядку t, то p є підрядком t.

У наївному алгоритмі пошук всіх допустимих зсувів реалізується за допомогою циклу, в якому перевіряється умова t[s … s + m – 1] = p для кожного з n – m + 1 можливих значень s.

Алгоритм працює за O(m(n-m)), в найгіршому випадку m = n/2 деградуючи до O(n^2). Однак якщо довжина зразка порівняно мала, то час роботи алгортму прямує до лінійного від n. Алгоритм потребує лише О(1) додаткової пам’яті.

* void naiveSearch(char \*pat, char \*txt) — реалізація наївного алгоритму
* void naiveDemo() - демонстрація коректності результатів алгоритму в консольному вікні

1. Алгоритм Рабіна-Карпа

Алгоритм пошуку рядка, запропонований Рабіном і Карпом. Ідея алгоритму полягає в заміні текстових рядків числами(використовуючи хешування), порівняння яких можна виконувати значно швидше.

* void rabinKarpSearch(char \*pat, char \*txt, int q) — реалізація алгоритму Рабіна-Карпа
* void rabinKarpDemo() - демонстрація коректності результатів алгоритму в консольному вікні

1. Алгоритм Хорспула

Алгоритм Хорспула є дещо спрощеним варіантом алгоритму Боєра-Мура, він працює краще на випадкових рядках та не використовує досить складно обчислювану суфікс-евристику.

Фаза попередньої обробки знаходиться в O (m + сигма) і складності O (сигма). Фаза пошуку має квадратичний найгірший випадок, але можна довести, що середнє число порівнянь для одного текстового символу становить від 1 / сигма до 2 / (сигма + 1).

* void shifttable(char p[]) - побудова таблиці зсувів
* int horspool(char src[], char p[]) - реалізація алгоритму Хорспула
* void horspoolDemo() - демонстрація коректності результатів алгоритму в консольному вікні

1. Алгоритм Боєра-Мура

Даний алгоритм є еталоном при практичних дослідженнях алгоритмів пошуку рядка. Перевага цього алгоритму в тому, що ціною деякої кількості попередніх обчислень над зразком або шаблоном (але не над рядком, в якому ведеться пошук) шаблон порівнюється з вихідним текстом не у всіх позиціях — частина перевірок пропускаються як такі, що не дадуть результату. Цей алгоритм швидко працює у ситуаціях коли зразок набагато коротший від тексту пошуку, або коли відбувається пошук в декількох документах. Зазвичай, чим довше зразок, тим швидше працює алгоритм.

Даний алгоритм має асимптотику О(N+M)

Асимптотика алгоритму Боєра-Мура складає в гіршому випадку

M \* N, але ймовірність найгіршого випадку доволі мала.

* int max(int a, int b) — допоміжна функція визначення максимуму з двох чисел
* void badCharHeuristic(char \*str, int size, int badchar[NO\_OF\_CHARS]) - функція визначення таблиці відносних незбігів, або “поганих символів”
* void boyerMooreSearch(char \*txt, char \*pat) — реалізація алгоритму Боєра-Мура
* void boyerMooreDemo() - демонстрація коректності результатів алгоритму в консольному вікні

Алгоритм базується на трьох ідеях.

1. Сканування зліва направо, порівняння справа наліво. Зіставляються початок рядку і зразка, перевірка починається з останнього символу зразка. Якщо символи збігаються, виконується перевірка передостаннього символу зразка з відповідним символом рядка і т. д. Якщо всі символи зразка збігаються із символами рядка, то рядок містить зразок. Інакше, якщо певний символ зразка не збігається з відповідним символом рядка, зразок зсувається на кілька символів вправо, і перевірка знову починається з останнього символу зразка. Кількість символів зсуву визначається наступними ідеями:
2. Евристика стоп-символу (англ. bad character).
3. Евристика суфікса, що збігся.

Алгоритм потребує О(1) додаткової пам’яті.

**Література :**

1. <https://uk.wikipedia.org/wiki/Алгоритм_Кнута_—_Морріса_—_Пратта> - Алгоритм Кнута - Моріса - Прата
2. [**https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%A0%D0%B0%D0%B1%D0%B8%D0%BD%D0%B0\_%E2%80%94\_%D0%9A%D0%B0%D1%80%D0%BF%D0%B0**](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A0%D0%B0%D0%B1%D0%B8%D0%BD%D0%B0_%E2%80%94_%D0%9A%D0%B0%D1%80%D0%BF%D0%B0) **– Алгоритм Рабіна - Карпа**
3. [**https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC\_%D0%91%D0%BE%D0%B9%D0%B5%D1%80%D0%B0\_%D0%9C%D1%83%D1%80%D0%B0**](https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%91%D0%BE%D0%B9%D0%B5%D1%80%D0%B0_%D0%9C%D1%83%D1%80%D0%B0) **– Алгоритм Боєра – Мура**
4. [**https://www.youtube.com/watch?v=D4nI\_oH-JjI**](https://www.youtube.com/watch?v=D4nI_oH-JjI)[**http://www.cyberforum.ru/cpp-builder/thread389497.html**](http://www.cyberforum.ru/cpp-builder/thread389497.html) **– Алгоритм Хорспула**