ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І.І.МЕЧНИКОВА Факультет математики, фізики та інформаційних технологій

**Звіт з лабораторної роботи № 1**

з дисципліни: “Алгоритми та структури даних” на тему:

“Аналіз алгоритмів”

Виконав: студент 2 курсу

денної форми навчання

спеціальності 123 «Комп'ютерна інженерія»

Мовчан Ілля Олександрович

**Лабораторна робота № 1**

**Тема:** Діагностика роботи алгоритмів обчислення різних програм.

**Мета**: зробити часові заміри оптимізованих та не оптимізованих алгоритмів на мові програмування C#, проаналізувати отримані дані

**Програмне забезпечення:** Windows 10, Visual Studio 2022

**Постановка задачі**

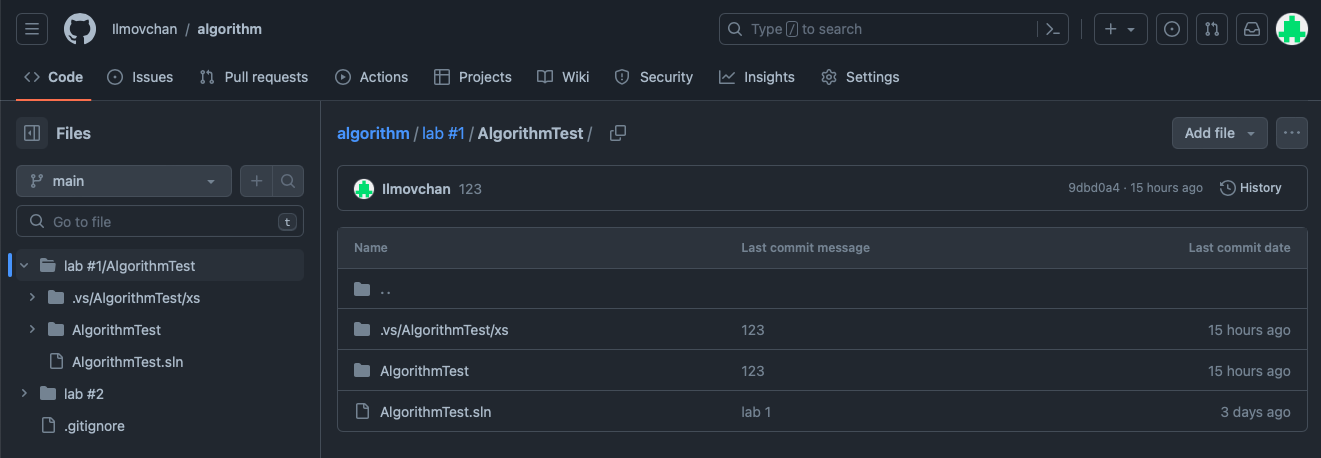
1. Реалізувати за допомогою функцій (або статичних методів мови C#) декілька варіантів вирішення завдань.
2. Порівняти їх швидкість та побудувати таблицю залежності часу виконання від n для декількох реалізацій (опціонально побудувати відповідні графіки).
3. Скласти звіт, куди включити ці дані.

**Завдання**

1. [Шифр Цезаря](#tstuxg8alzub)
2. [Пошук елемента у масиві](#q5mxbcqd3n9l)
3. [Сортування масиву](#jbvof6kx187)

**Посилання на код**

<https://github.com/Ilmovchan/algorithm/tree/9dbd0a49094b16438478d49d4cda6845c653b5a8/lab%20%231/AlgorithmTest>



**Шифр Цезаря**

Шифр Цезаря - це один із найпростіших методів сортування, який зсуває кожну букву на кілька символів, таким чином шифруючи текст.

Розглянемо два алгоритми кодування:

1. Перший спосіб -

public static string FirstEncryptionMethod(string message, int shift)

{

string result = "";

foreach (char sym in message)

{

if (char.IsLetter(sym))

{

char shiftedSym = (char)(sym + shift);

if ((char.IsLower(sym) && shiftedSym > 'z') || (char.IsUpper(sym) && shiftedSym > 'Z'))

{

shiftedSym = (char)(sym - (26 - shift));

}

result += shiftedSym;

}

else

{

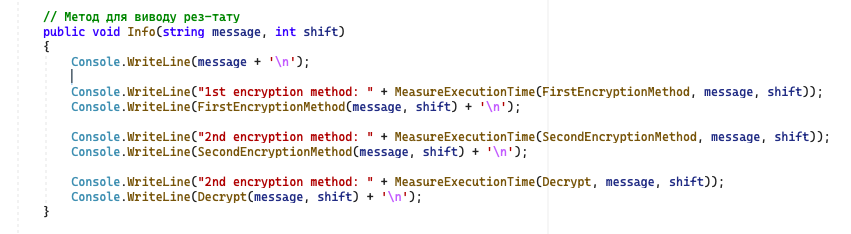
result += sym;

}

}

return result;

}



1. Другий спосіб -

public static string FirstEncryptionMethod(string message, int shift)

{

string result = "";

foreach (char sym in message)

{

if (char.IsLetter(sym))

{

char shiftedSym = (char)(sym + shift);

if ((char.IsLower(sym) && shiftedSym > 'z') || (char.IsUpper(sym) && shiftedSym > 'Z'))

{

shiftedSym = (char)(sym - (26 - shift));

}

result += shiftedSym;

}

else

{

result += sym;

}

}

return result;

}

Та метод декодування:

public string Decrypt(string message, int shift)

{

return SecondEncryptionMethod(message, -shift);

}

| Повідомлення | | | Час | | |
| --- | --- | --- | --- | --- | --- |
| Повідомлення | Шифр | | 1 метод кодування | 2 метод кодування | Декодування |
| *hello* | *mjqqt* | | 416322 | 547724 | 553476 |
| *please read my message and show me encrypted text* | *uqjfxj wjfi rd rjxxflj fsi xmtb rj jshwduyji yjcy* | | 513524 | 600243 | 602478 |

Отже можемо зробити висновок що 1 метод кодування повідомлення трохи ефективніший.

**Пошук елемента у масиві**

Я реалізував 2 способи пошуку - це пошук перебром та бінарний пошук. Кожен із них має свої переваги та недоліки, наприклад пошук перебором буде доволі повільним при пошуку числа у великому масиві, а бінарний пошук буде коректно працювати тільки з відсортованим масивом.

Отже розглянемо два алгоритми пошуку:

1. Стандартний пошук перебором :

public static bool SimpleSearch

(int[] stack, int needle, int start = 0, int end = -1)

{

if (end == -1) end = stack.Length - 1;

for (int i = start; i <= end; i++)

{

if (stack[i] == needle) return true;

}

return false;

}

1. Бінарний пошук :

public static bool BinarySearch

(int[] stack, int needle, int start = 0, int end = -1)

{

if (end == -1) end = stack.Length - 1;

while (start <= end)

{

int mid = (start + end) / 2;

if (stack[mid] == needle) return true;

else if (stack[mid] < needle) start = mid + 1;

else if (stack[mid] > needle) end = mid - 1;

}

return false;

}

Я створив випадковий масив з елементів від -50 до 50 і почав пошук числа 49. Ось які результати я отримав:

* **Не відсортований масив:**

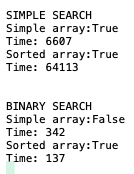
| Розмір масиву | Час | |
| --- | --- | --- |
| Пошук перебором | Бінарний пошук |
| 5000 | 3117 | 256 |
| 10000 | 3486 | 422 |
| 20000 | 3997 | 536 |
| 30000 | 5209 | 730 |

* **Відсортований масив:**

| Розмір масиву | Час | |
| --- | --- | --- |
| Пошук перебором | Бінарний пошук |
| 5000 | 15825 | 166 |
| 10000 | 32029 | 180 |
| 20000 | 52113 | 202 |
| 30000 | 65121 | 308 |

Але потрібно зазначити, що бінарний пошук у не відсортованому масиві буде працювати некоректно та видавати неправильні результати. Отже можемо зробити висновок, що:

Бінарний пошук у відсортованому масиві буде працювати набагато ефективніше, але буде працювати неправильну у не відсортованому. Натомість стандартний пошук перебором у масиві, буде працювати менш ефективно у великому масиві, але завжди буде видавати вірний результат.



**Сортування масиву**

Сортування масиву - одна із базових операцій взаємодії із масивами. Я зробив 2 методи сортування масиву - **сортування вибором** та **бульбашкове сортування**

Розглянемо два алгоритми сортування:

1. Сортування вибором:

public static int[] InsertSort(int[] originalArr)

{

int[] arr = originalArr.ToArray();

for (int j = 0; j < arr.Length; j++)

{

int minIndex = j;

for (int i = j; i < arr.Length; i++)

{

if (arr[minIndex] > arr[i])

{

minIndex = i;

}

}

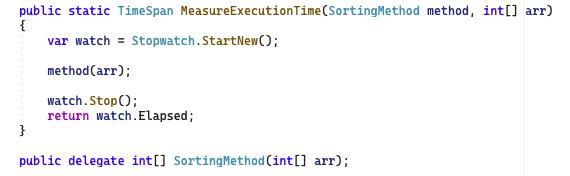
int tmp = arr[j];

arr[j] = arr[minIndex];

arr[minIndex] = tmp;

}

return arr;

}

Метод вимірювання часу виконання методу сортування:

1. Бульбашкове сортування:

public static int[] BubbleSort(int[] originalArr)

{

int[] arr = originalArr.ToArray();

for (int j = 0; j < arr.Length - 1; j++)

{

bool swapped = false;

for (int i = 0; i < arr.Length - j - 1; i++)

{

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

{

int tmp = arr[i];

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

arr[i + 1] = tmp;

swapped = true;

}

}

if (!swapped) return arr;

}

return arr;

}

| Розмір масиву | Час | |
| --- | --- | --- |
| Сортування з вибором | Бульбашкове сортування |
| 50 | 23615247 | 652364 |
| 500 | 26640491 | 2044528 |
| 5000 | 70306117 | 115602305 |
| 10000 | 250000361 | 504027084 |

Отже можна зробити висновок, що **бульбашкове сортування** працює більш ефективно у невеликих масивах, але чим більше масив, тим більш ефективно буде застосування алгоритму **сортування із вибором**

**Висновок**

* цій лабораторній роботі я реалізував по два алгоритми для наступних завдань: Шифр Цезаря, пошук елементу в масиві та сортування масиву. Було виміряно час виконання цих алгоритмів, проаналізовано їх складність, а також складено порівняльні таблиці і графіки, з яких видно, який з алгоритмів є найшвидшим. Дякую за увагу!