МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

факультет програмної інженерії та бізнесу

кафедра інженерії програмного забезпечення

**Лабораторна робота № 6**

з дисципліни « Алгоритми і структури даних »

*назва дисципліни*

на тему: «СОРТУВАННЯ ПРОСТИХ СТРУКТУР ДАНИХ»

Виконав: студент 2 курсу групи № 622п

освітньої програми

121 інженерія програмного забезпечення

(шифр і назва ОП)

Зайченко Ярослав Ігорович

(прізвище й ініціали студента)

Прийняв: канд. техн. наук, доцент на кафедрі 603 Волобуєва Ліна Олексіївна

(посада, науковий ступінь, прізвище й ініціали)

Кількість балів:

Харків – 2023

**Мета роботи** − знайомство з існуючими алгоритмами сортування простих структур даних та виконання програмної реалізації заданого алгоритму.

Постановка задачі

1. Ознайомитися з алгоритмом сортування даних для програмної реалізації алгоритму сортування чисел за варіантом.
2. Записати алгоритм сортування масиву чисел за варіантом двома способами: вербально та у вигляді блок-схеми чи псевдокоду.
3. Задавати кількість елементів масиву чисел, що буде сортуватися, в режимі діалогу.
4. Використовувати при ініціалізації масиву даних генератор випадкових чисел.
5. Реалізувати програму алгоритму сортування масиву чисел. Представити програмну реалізацію алгоритму сортування у вигляді окремого методу (функції) з параметрами.
6. Створити презентацію алгоритму за варіантом.

Зміст звіту

Зміст звіту з лабораторної роботи №6 має включати наступні підрозділи:

1. Запис алгоритму сортування масиву чисел за варіантом двома способами: вербально та у вигляді блок-схеми чи псевдокоду.
2. Приклад, що показує сутність алгоритму сортування чисел з докладним описом на кожному кроці алгоритму.
3. Лістинг програми та екранні форми результатів роботи програми.
4. Презентацію алгоритму за варіантом.
5. Висновки

**Варіант 9** індивідуального завдання до лабораторної роботи 6:

Паралельне сортування Бетчера. Алгоритм M(обміну зі злиттям). (Batcher's Merge-Exchange Sort)

хід роботи

Опис програми:

Мова програмування: С#, операційна система Windows 11 Prо, Версія 23H2, Збірка ОС 22621.1325, процесор: Apple Silicon M1 Pro 3.20 GHz (ядер: 6), компілятор: Microsoft Visual Studio Community 2022 (64-розрядна версія ARM).

Теоретичні відомості

Паралельне сортування Бетчера. Щоб отримати алгоритм обмінного сортування, час роботи якого має порядок, менший за N^2, необхідно підібрати для порівнянь пари несусідніх ключів (Kj, Кі); інакше доведеться виконати стільки операцій обміну записів, скільки інверсій є в початковій перестановці. Середнє число інверсій дорівнює ¼(N^2-N). У 1964 році К. Е. Бетчер відкрив цікавий спосіб програмування послідовності порівнянь, призначеної для пошуку можливих обмінів. Його метод далеко не очевидний. Справді, обґрунтувати його справедливість досить складно, оскільки виконується відносно мало порівнянь.

Схема сортування Бетчера дещо нагадує сортування Шелла, але порівняння виконуються по-новому, а тому ланцюжка операцій обміну записів не виникає. Сортування Бетчера діє, як 8-, 4-, 2- і 1-сортування, але порівняння не перекриваються. Оскільки в алгоритмі, по суті, відбувається злиття пар розсортованих підпослідовностей, його можна назвати обмінним сортуванням зі злиттям.

Вербальний опис алгоритму сортування М (Обмінне сортування зі злиттям)

Записи R1, ... Rn перекомпоновуються в межах того самого простору в пам'яті. Після завершення сортування їхні ключі будуть упорядковані: К1 ≤...≤ КN. Припускається, що N ≥ 2

**M1.** [Початкове встановлення р.] Встановити р 🡨 2t-1, де *t = lgN* - найменше ціле число, таке, що 2t ≥ N. (Кроки М2-М5 виконуватимуться з *р* = 2t-1, *р* = 2t-2, ... , ... , 1.)

**М2.** [Початкове встановлення *q, r, d*.] Встановити q 🡨2t-1, r 🡨 0, d 🡨 р.

**М3.** [Цикл по *i*.] Для всіх *i*, таких, що 0 ≤ *i* ≤ *N - d* і *i* /\ *р* = *r*, виконати крок М4. Потім перейти до кроку М5. (Тут через *i* /\ *р* позначено операцію "порозрядне логічне І" над поданнями цілих чисел *i* і *p*; усі біти результату дорівнюють 0, окрім тих бітів, для яких у відповідних розрядах *i* і *p* знаходяться 1. Так, 13 & 21 = (1101)2 /\ (10101)2 = (00101)2 = (00101)2 = 5. До цього моменту *d* - непарне кратне р (тобто частка від ділення *d* на *р* непарна), а *p* - ступінь двійки, тож i /\ р нерівно *(i + d) /\ р*. Звідси випливає, що крок М4 можна виконувати за всіх потрібних значень i в будь-якому порядку або навіть одночасно).

**М4.** [Порівняння/обмін Ri+1 : Ri+d+1] Якщо Ki+1 > Ki+d+1, поміняти місцями записи Ri+1 і Ri+d+1.

**М5.** [Цикл по *q*.] Якщо *q != р*, встановити *d =q - р*, *q = q/2*, *r = р* і повернутися до кроку М3.

**М6.** [Цикл по *p*.] (До цього моменту перестановка К1, К2 ... КN буде *p*-впорядкована.) Установити *р = [p/2]*. Якщо *р* > 0, повернутися до кроку М2.

Опис алгоритму сортування М (Обмінне сортування зі злиттям) у вигляді блок схеми

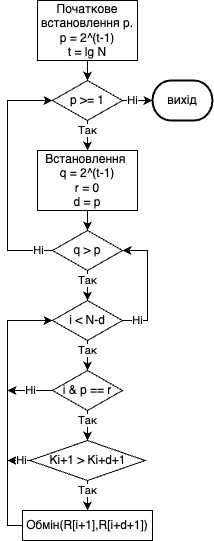


Рисунок 1 – Блок схема до алгоритму сортування

Приклад, для показу сутності алгоритму сортування чисел з докладним описом кожного кроку

1. Початкове встановлення:

N = 5, t = 3.

p = 2^(t-1) = 2^2 = 4.

1. Початкове встановлення q, r, d:

q = 4, r = 0, d = p = 4.

1. Цикл по i (M3):

* Ітерація 1 (i = 0):

Оскільки 0 /\ 4 = 0 = r, переходимо до кроку M4.

1. Порівняння/обмін (M4):

Порівнюємо K1 (5555) та K5 (1).

Оскільки K1 > K5, міняємо їх місцями, отримуючи [1, 4444, 333, 22, 5555].

1. Цикл по q (M5):

Збільшуємо r на q (r = 4), зменшуємо q удвічі (q = 2), зменшуємо d (d = 2), і повторюємо M3.

1. Цикл по i (повторення M3 та M4):

* Ітерація 2 (i = 0):

Оскільки 0 /\ 2 = 0 = r, переходимо до M4.

Порівнюємо K1 (1) та K3 (333).

Оскільки K1 < K3, залишаємо їх на місці.

* Ітерація 2 (i = 1):

Оскільки 1 /\ 2 = 0 = r, переходимо до M4.

Порівнюємо K2 (4444) та K4 (22).

Оскільки K2 > K4, міняємо їх місцями, отримуючи [1, 22, 333, 4444, 5555].

1. Цикл по q (повторення M5):

Збільшуємо r на q (r = 2), зменшуємо q удвічі (q = 1), зменшуємо d (d = 1), і повторюємо M3.

1. Цикл по i (повторення M3 та M4):

* Ітерація 3 (i = 0):

Оскільки 0 /\ 1 = 0 = r, переходимо до M4.

Порівнюємо K1 (1) та K2 (22).

Оскільки K1 < K2, залишаємо їх на місці.

* Ітерація 3 (i = 1):

Оскільки 1 /\ 1 = 1 ≠ r, не виконуємо M4.

* Ітерація 3 (i = 2):

Оскільки 2 /\ 1 = 0 = r, переходимо до M4.

Порівнюємо K3 (333) та K4 (4444).

Оскільки K3 < K4, залишаємо їх на місці.

1. Цикл по q (повторення M5):

Збільшуємо r на q (r = 1), зменшуємо q удвічі (q = 0), зменшуємо d (d = 0), і завершуємо M3.

Після виконання всіх ітерацій алгоритму M масив відсортований за зростанням [1, 22, 333, 4444, 5555].

Лістинг програми

using System.Text;

Console.OutputEncoding = Encoding.UTF8;

Console.InputEncoding = Encoding.UTF8;

Console.WriteLine("Для виходу введіть \"-0\"\n");

while (true){

try

{

int length = 0;

Console.Write("Введіть довжину масиву: ");

string input = Console.ReadLine();

if (input == "-0") Environment.Exit(0);

if (int.TryParse(input, out int temp))

if (input != null && temp >= 2)

length = temp;

else throw new Exception("Помилка введення кількості елементів масиву");

else throw new Exception("Помилка читання з рядка");

int[] array = GenerateRandomArray(length);

// Виведення початкового масиву

PrintArr("Початковий масив: ", array);

// Сортування масиву

BatcherMSort(array, length);

// Виведення відсортованого масиву

PrintArr("Відсортований масив: ", array);

}

catch (Exception ex) { Console.WriteLine(ex.Message); }

}

void PrintArr(string message, int[] array)

{

Console.WriteLine(message);

foreach (int el in array)

Console.Write($"{el} ");

Console.WriteLine();

}

static int[] GenerateRandomArray(int length)

{

Random random = new Random();

int[] array = new int[length];

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

array[i] = random.Next(1000); // Генеруємо випадкові числа в діапазоні [0, 1000)

return array;

}

void BatcherMSort(int[] array, int length)

{

int t = -1;

int tempLength = length;

while (tempLength > 0)

{

tempLength >>= 1;

++t;

}

int p0 = 1 << t;

int p = p0;

do

{

int q = p0;

int r = 0;

int d = p;

while (r == 0 || q != p)

{

if (r != 0)

{

d = q - p;

q >>= 1;

}

for (int i = 0; i < length - d; ++i)

{

if (((i & p) == r) && array[i] > array[i + d])

{

int swap = array[i];

array[i] = array[i + d];

array[i + d] = swap;

}

}

r = p;

}

p >>= 1;} while (p > 0);}

Екранні форми результатів роботи програми

Введення некоректної кількості елементів масиву. Результат подано на рисунку 2.

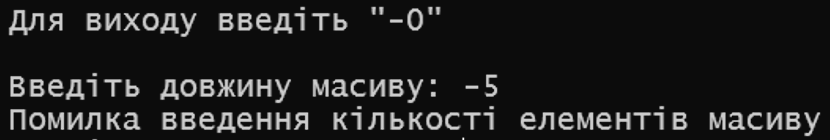


Рисунок 2 – перевірка на введення кількості елементів

Введення некоректних числових даних для кількості елементів масиву. (буквенні символи і т.п.). Результат на рисунку 3.

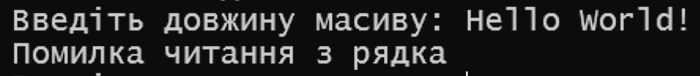


Рисунок 3 – поведінка при введенні не числових даних

Перевірка введення мінімального розміру масиву та його сортування. Результат на рисунку 4.

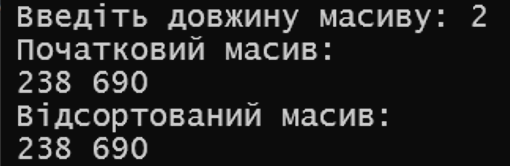


Рисунок 4 – робота з мінімальним масивом

Сортування масиву з 20 випадково згенерованими елементами. Результат на рисунку 5.

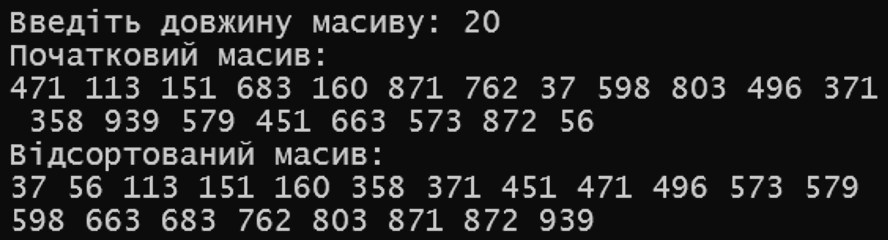


Рисунок 5 – масив з 20 елементами

ВИСНОВОК

У ході виконання лабораторної роботи №6 було вивчено алгоритми сортування масивів чисел. Алгоритм було записано двома способами: у вербальній формі та у вигляді блок-схеми.

Суть алгоритму полягає в обміні сусідніх елементів масиву за умови, що перший елемент менший за другий. Цей процес повторюється для кожної пари сусідніх елементів масиву. Після кожного проходження масиву величина останнього елемента вже впорядкована, і такий процес повторюється до тих пір, поки не буде досягнуто повного упорядкування всього масиву.

При реалізації програми було використано генератор випадкових чисел для ініціалізації масиву даних. Окремий метод з параметрами був створений для реалізації алгоритму сортування.

У ході вивчення алгоритму та його реалізації була створена презентація, яка відображає ключові етапи та особливості алгоритму сортування масиву чисел.

Завдання дозволило краще зрозуміти принципи та механізми сортування масивів чисел, а також вивчити практичні аспекти реалізації цього алгоритму в програмному коді.