**Модульна контрольна робота №2**

Маньківський Владислав

ВТ-21-1[2]

ВАРІАНТ 2

СОРТУВАННЯ ЗА РОЗРЯДАМИ (RADIX SORT)

***Мета*** : реалізація та опис алгоритма сортування за розрядами.

**2.1 Хід роботи**





2.1.1

**Завдання**:

***Результати роботи програми:***

На скріншоті можна побачити, що алгоритм правильно відсортував масив з 10 елементів, та тестування різної кількості елементів. Можна чітко побачити, що час сортування на скріншотах (Рис 2.1) відрізняється певною похибкою, чиб більше елементів, тим більше похибка.

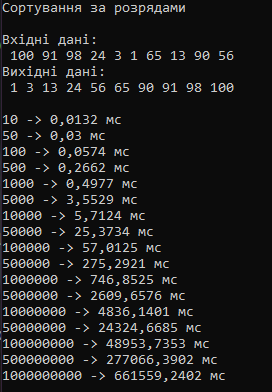
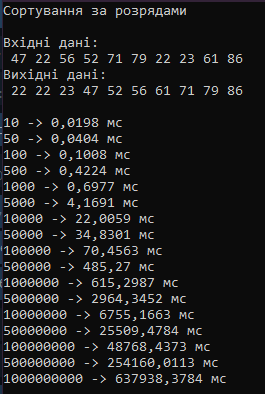
 

Рисунок 2.1 – Результати виконання завдання

***Загальний словесний опис алгоритму:***

**Сортування за розрядами (radix sort)** – швидкий стабільний алгоритм сортування даних, який використовує розряди числа. Головна ідея алгоритму, сортувати за кожним розрядом, від наймолодшого до найстаршого. Плюс алгоритма – швидкість завжди стала. Мінус в тому, що він сортує лише цілі числа. Складність алгоритму залежить лише від кількості елементів, чим їх більше, тим довше сортує.

Загальна складність роботи алгоритму з використанням сортування підрахунком є {\displaystyle O(D\cdot \;(N+K))}O(D \* (N + K)) (де N — кількість елементів в масиві; K — кількість символів у алфавіті, якщо впорядковуються десяткові числа, то K = 10 D — кількість розрядів). Якщо впорядковувати цим алгоритмом цілі числа, то складність буде {\displaystyle \;O(N\log M)}O(N log M), де M — найбільший елемент масиву.

***Короткий опис алгоритму:***

1. Пошук найбільшого елемента.
2. Розрахунок кількості ітерацій розряду в найбільшому елементі, входження до циклу починаючи з найменшого.
3. Створення масиву для вихідних даних, та додаткового масиву для оперування над елементами.
4. Заповнення додаткового масиву ‘0’.
5. Пошук кількості цифр певного розряду від 0 до 9.
6. Перерозподіл цифр в додатковому масиві під їх фактичне розташування.
7. Встановлення елементів на їх відсортоване місце одразу в масиві вихідних данних.
8. Копіювання до основго масиву.

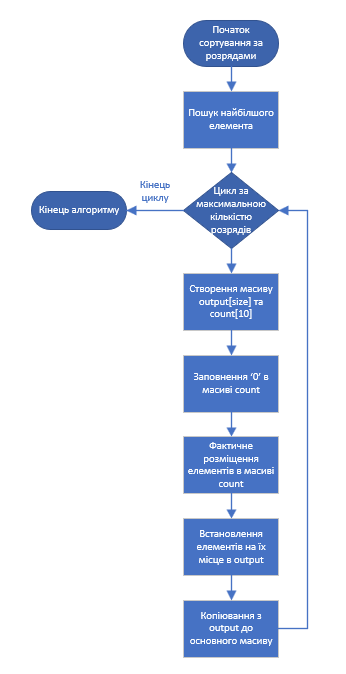
******

Рисунок 2.2 – Блок-схема до алгоритму

***Структура даних:***

Було вирішено використати структуру даних типу int, заповнену числами від 0 до 100 для демонстрації, від 0 до 2147483646 для тестування, так як алгоритм не підтримує числа з плаваючую комою. Структур кілька, так як необхідно було переглянути результати сортування за часом, а саме: arr для демонстрації сортування, всі інші, для тестування, arr\_10, arr\_50, arr\_100, arr\_500, arr\_1000, arr\_5000, arr\_10000, arr\_50000, arr\_100000, arr\_500000, arr\_1000000, arr\_5000000, 10000000, ще окремо були використані структури для дуже великої кількості чисел, arr\_50000000, arr\_100000000, arr\_500000000, arr\_1000000000.

Для самого алгоритму було використано arr – основний масив, output – для кінцевого розміщення елементів під час ітерації, count – для оперування над розмішенням чисел.

***Тестові дані для програми:***

Було перевіренно алгоритм на такому діапазоні чисел [0; 2147483646].

Було перевіренно алгоритм на такій кількості елементів: 10, 50, 100, 500, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 5000000, 10000000, 50000000, 100000000, 500000000, 1000000000.

***Графіки та таблиця:***

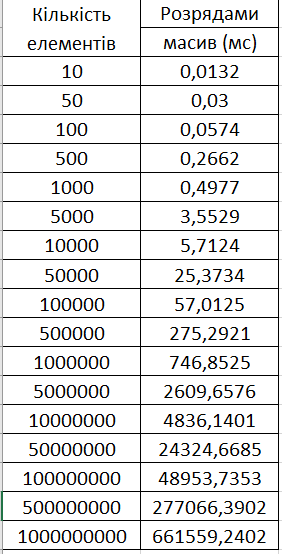


Рисунок 2.3 – Таблиця

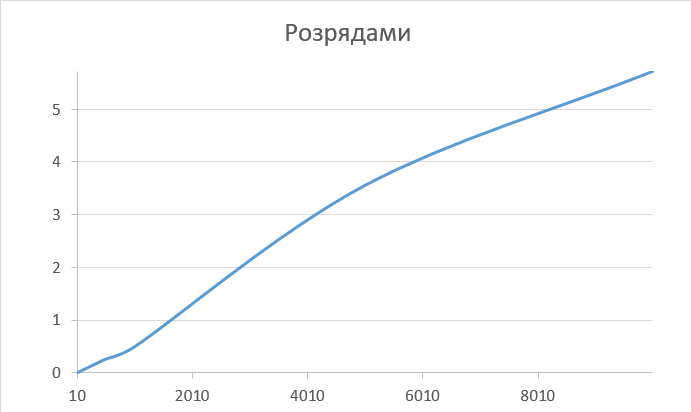


Рисунок 2.4 – Графік від 10 до 10000



Рисунок 2.5 – Графік від 10 до 1000000000

***Посилання на програму в гітлаб:***

<https://gitlab.com/RAK_MANIAK/algoritmi_modular_control_work_2>

Лістинг:

using System.Diagnostics;

namespace Radix\_Sort

{

class Program

{

public static int Max(int[] arr, int size)//знаходження максимального елемента

{

int max = arr[0];

for (int i = 1; i < size; i++)

if (arr[i] > max)

max = arr[i];

return max;

}

public static void RadixSort(int[] arr, int size)//сортування за розрядами

{

int max = Max(arr, size);//максимальний елемент

for (int exponent = 1; max / exponent > 0; exponent \*= 10)//кількість ітерацій відповідна кількості розрядів максимального елемента

CountingSort(arr, size, exponent);//сортування

}

public static void CountingSort(int[] arr, int size, int exponent)

{

int[] output = new int[size];//вихідний масив

int[] count = new int[10];//масив для оперування над елементами

for (int i = 0; i < 10; i++)//заповнення додаткового масива значенням '0'

count[i] = 0;

for (int i = 0; i < size; i++)//кількість цифр певного розряду від 0 до 9

count[(arr[i] / exponent) % 10]++;

for (int i = 1; i < 10; i++)//фактичне розміщення цифр

count[i] += count[i - 1];

for (int i = size - 1; i >= 0; i--)//встановлення елементів на їх відсортоване за розрядом місце

{

output[count[(arr[i] / exponent) % 10] - 1] = arr[i];

count[(arr[i] / exponent) % 10]--;

}

for (int i = 0; i < size; i++)//копіювання в основний масив

arr[i] = output[i];

}

public static void check(int[]arr, int k, int min, int max)

{

Stopwatch stopwatch = new Stopwatch();

Random random = new Random();

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

arr[i] = random.Next(min, max + 1);

stopwatch.Start();

RadixSort(arr, k);

stopwatch.Stop();

Console.WriteLine($"{k} -> {stopwatch.Elapsed.TotalMilliseconds} мс");

stopwatch.Reset();

}

static void Main()

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.InputEncoding = System.Text.Encoding.Unicode;

Random random = new Random();

Console.WriteLine("Сортування за розрядами");

int size = 10, min = 0, max = 100;

//k >= 1, min >= 0, max <= 2147483646;

int[] arr = new int[size];

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

arr[i] = random.Next(min, max + 1);

Console.WriteLine("\nВхідні дані: ");

foreach (var x in arr)

{

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

}

RadixSort(arr, size);

Console.WriteLine("\nВихідні дані: ");

foreach (var x in arr)

{

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

}

Console.WriteLine("\n");

////////////////////////////////////////////////////////////

min = 0;

max = 2147483646;

size = 10;

int[] arr\_10 = new int[size];

check(arr\_10, size, min, max);

size = 50;

int[] arr\_50 = new int[size];

check(arr\_50, size, min, max);

size = 100;

int[] arr\_100 = new int[size];

check(arr\_100, size, min, max);

size = 500;

int[] arr\_500 = new int[size];

check(arr\_500, size, min, max);

size = 1000;

int[] arr\_1000 = new int[size];

check(arr\_1000, size, min, max);

size = 5000;

int[] arr\_5000 = new int[size];

check(arr\_5000, size, min, max);

size = 10000;

int[] arr\_10000 = new int[size];

check(arr\_10000, size, min, max);

size = 50000;

int[] arr\_50000 = new int[size];

check(arr\_50000, size, min, max);

size = 100000;

int[] arr\_100000 = new int[size];

check(arr\_100000, size, min, max);

size = 500000;

int[] arr\_500000 = new int[size];

check(arr\_500000, size, min, max);

size = 1000000;

int[] arr\_1000000 = new int[size];

check(arr\_1000000, size, min, max);

size = 5000000;

int[] arr\_5000000 = new int[size];

check(arr\_5000000, size, min, max);

size = 10000000;

int[] arr\_10000000 = new int[size];

check(arr\_10000000, size, min, max);

////

// size = 50000000;

// int[] arr\_50000000 = new int[size];

// check(arr\_50000000, size, min, max);

// size = 100000000;

// int[] arr\_100000000 = new int[size];

// check(arr\_100000000, size, min, max);

// size = 500000000;

// int[] arr\_500000000 = new int[size];

// check(arr\_500000000, size, min, max);

// size = 1000000000;

// int[] arr\_1000000000 = new int[size];

// check(arr\_1000000000, size, min, max);

}

}

}

***Висновки:*** під час цієї модульної контрольної роботи я вивчив новий метод сортування, створив його алгоритм та дослідив. Під час виконання описав алгоритм та результати програми, словами та блок-схемою, зобразив результати програми.