МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра інформаційних систем та технологій

Звіт

з комп'ютерного практикуму 8(12)

«Алгоритми сортування. Сортування даних з великими ключами»

з дисципліни

«Теорія алгоритмів»

Бригада – 10

Варіант № 1

Перевірила:
ст. вик. Солдатова М.О.

Виконали:
Бойко Катерина,
Гоголь Софія,
Павлова Софія,
Хіврич Володимир

Київ 2022

Комп’ютерний практикум 12

Тема: Алгоритми сортування. Сортування даних з великими ключами

Мета роботи: Дослідження та порівняння алгоритмів сортування.

Завдання

Постановка задачі:

Варіант 1:

Після того, як зібрали дані про скарби всіх родин гномів, король леприконів наказав скласти список сумарних скарбів родин. При збиранні таких даних з’ясувалося, що родини дуже багаті, тобто мають великі статки. Допоможіть скласти помічникам короля відповідні списки.

Мета задачі – допомогти помічникам короля скласти відсортовані списки скарбів

Використані структури даних: масиви, покажчики, цілочисельні типи, числа з плаваючою точкою, рядки.

Вибір алгоритму

Порозрядне сортування (англ. radix sort) - алгоритм сортування, який виконується за лінійний час. Існують стабільні варіанти.

Базово призначений для сортування цілих чисел, записаних цифрами. Але оскільки у пам'яті комп'ютерів будь-яка інформація записується цілими числами, алгоритм придатний для сортування будь-яких об'єктів, запис яких можна розділити на «розряди», які мають порівняні значення. Наприклад, так сортувати можна як числа, записані як набору цифр, а й рядки, є набором символів, і взагалі довільні значення пам'яті, представлені як набору байт.

Сортування LSD групуванням

Рухаємося від молодших розрядів до старших і кожної ітерації розподіляємо елементи масиву залежно від цього, яка цифра міститься у розряді.

Після чергового розподілу ми повертаємо елементи до основного масиву у порядку, у якому елементи потрапили до класів при черговому перерозподілі.

Складність алгоритму сортування – **O (n + k)**

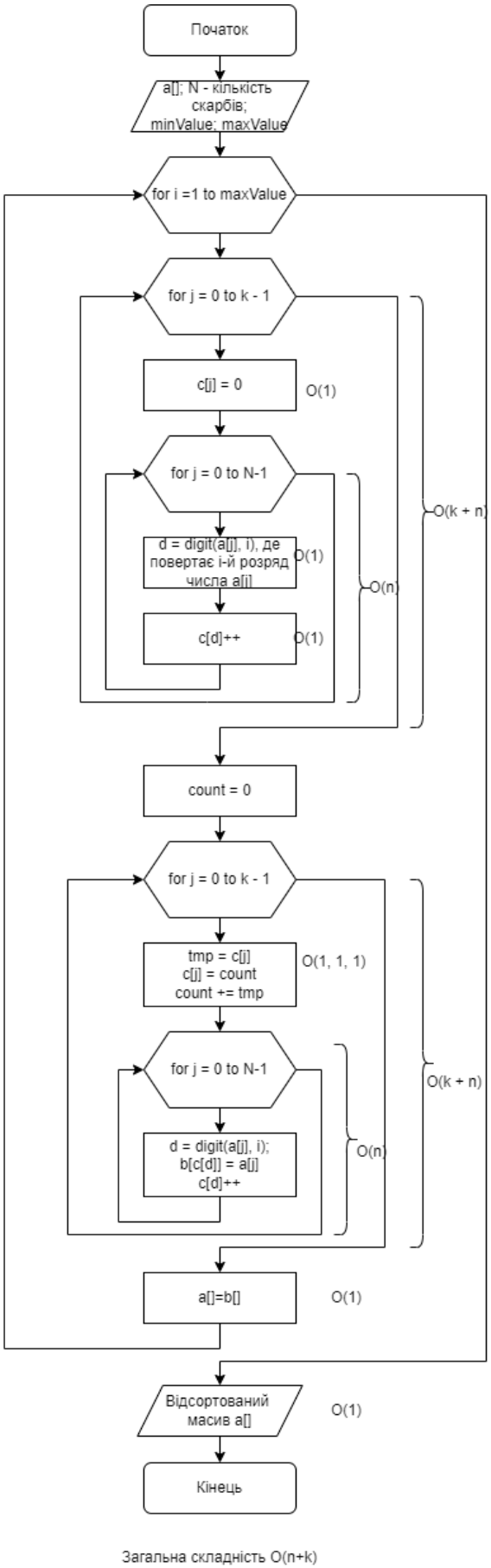


Рис. 1. Блок-схема алгоритму LSD сортуванням

Сортування MSD групуванням

Рухаємося від старших розрядів до молодших і кожної ітерації розподіляємо елементи масиву залежно від цього, яка цифра міститься у розряді.

Після чергового розподілу рекурсивно сортуються підмасиви елементів, що згруповані за кожною цифрою старшого розряду.

Складність алгоритму сортування – **O (n + k)**

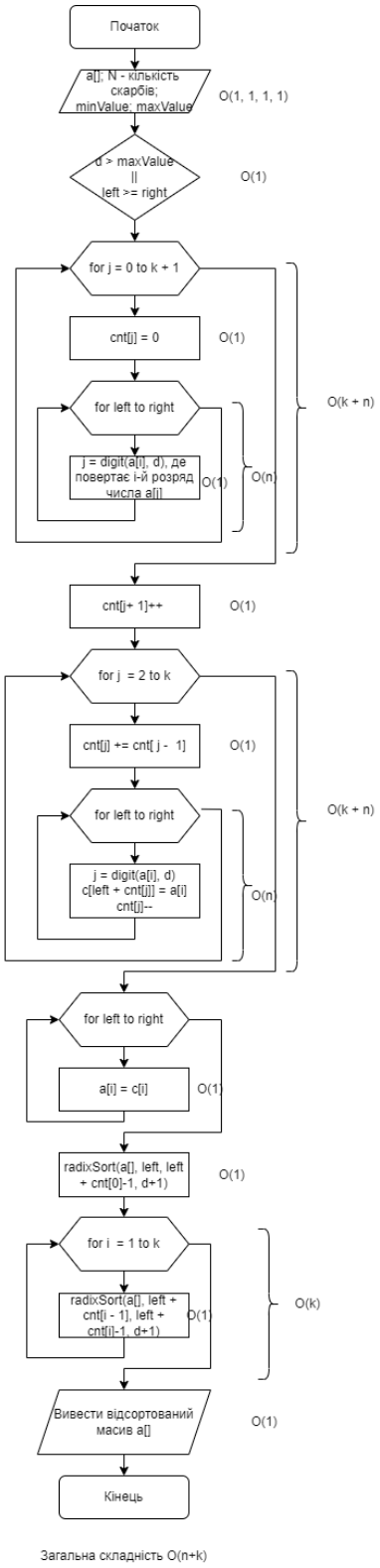


Рис. 2. Блок-схема алгоритму MSD сортуванням

Вхідні дані

Назва змінної	Тип змінної	Значення змінної
length0	float	Кількість скарбів родин гномів
end0	float	Максимальна вартість скарбу
partlySorted0	string	Чи створювати частково сортований масив
varik	int	Допомагає обрати тип сортування
addLength0	float	Додаткова кількість скарбів гномів королівства
start0	float	Мінімальна вартість скарбу

Табл. 1. Таблиця вхідних даних

Модель:

Основні величини:

Назва змінної	Тип змінної	Значення змінної
length	int	Кількість скарбів родин гномів
end	int	Максимальна кількість золота
addLength	int	Додаткова кількість скарбів гномів королівства
partlySorted	int	Чи відсортовувати скарби в державній скарбничці
start	int	Мінімальна кількість золота
varik	int	Допомагає обрати алгоритм сортування

Табл. 2. Таблиця основних величин

Допоміжні величини:

Назва змінної	Тип змінної	Значення змінної
maxLength	int	Максимальна довжина масиву
output	int*	Результуючий масив

Табл 3. Таблиця допоміжних величин

Лістинг програмної реалізації

```
#include <iostream>
#include <windows.h>
#include <ctime>
#include <unordered_map>

using namespace std;

///****************************************** LSD Radix Sort
    ******************************************///
// function that returns max value of element
int getMaxLength(int arr[])
{
   return 4;
}

// function that returns the byte i of element
short int getByte(unsigned int elem, unsigned int i)
{
   if (sizeof(elem) > i) { return (elem >> ((3 - i) * 8) & (255)); }
   else return -1;
}

// main function for LSD
void radixSort_LSD(int arr[], int n)
{
```

```
    int maxLength = getMaxLength(arr);

    // Counting sort
    for (int d = maxLength - 1; d >= 0; d--)
    {
        int* output = new int[n];
        int count[256 + 2] = { 0 };

        for (int i = 0; i < n; i++)
            count[getByte(arr[i], d) + 2]++;

        for (int r = 0; r < 256 + 1; r++)
            count[r + 1] += count[r];

        for (int i = 0; i < n; i++)
            output[count[getByte(arr[i], d) + 1]++] = arr[i];

        for (int i = 0; i < n; i++)
            arr[i] = output[i];
    }
}

///****************************************** MSD Radix Sort
    ******************************************///
// helping function for MSD
void MSD(int* arr, int* output, int low, int n, int d)
{
```

```
    int i, count[256 + 2] = { 0 };

    for (i = low; i < n; i++)
        count[getByte(arr[i], d) + 2]++;

    for (int r = 0; r < 256 + 1; r++)
        count[r + 1] += count[r];

    for (int i = low; i < n; i++)
        output[count[getByte(arr[i], d) + 1]++] = arr[i];

    for (int i = low; i < n; i++)
        arr[i] = output[i - low];

    // if there is many times the same letter we launch a sort for all of them
    for (int r = 0; r < 256 + 1; r++)
        if (count[r] < count[r + 1]) MSD(arr, output, low + count[r], low + count[r
     + 1], d + 1);
}

// main function for MSD
void radixSort_MSD(int* arr, int n)
{
    int* output = new int[n];
    MSD(arr, output, 0, n, 0);
}
```

```
///************************************** Helping functions
    ***************************************///

void createArray(int arr[], int length, int start, int end)

{

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

      arr[i] = start + rand() % (end - start);

}


void addElementsToArray(int arr[], int start, int n, int end)

{

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

      arr[i] = rand() % end;

}


void printArray(int arr[], int n)

{

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

      cout << arr[i] << " ";

}


int ErorCheck(float a)

{

   if (a <= 0 || (a - int(a) != 0))

   {

      cout << "\n * -------- * ПОМИЛКА * -------- * \n";

      cout << " Упс! Уведено не натуральне число! \n\n";

      return 0;
```

```
    }
}

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    float length0, start0, end0, addLength0;
    int i, length, start, end, varik = 0, partlySorted = 0, addLength;
    string partlySorted0;

    cout << "\t * -------- * Комп'ютерний практикум 8 * -------- *\n";
    cout << "\t\t\t Бригада 10. Варіант 1  \n";
    cout << "\t Реалізація алгоритмів сортування з великими ключами\n\n";

    cout << "\nУ родин гномів усього N скарбів\nN = ";
    cin >> length0;
    ErorCheck(length0);
    length = int(length0);
    int* arr = new int[length];

    cout << "\nСума вартості скарбів від та до: ";
    cin >> start0 >> end0;
    ErorCheck(start0);
    start = int(start0);
```

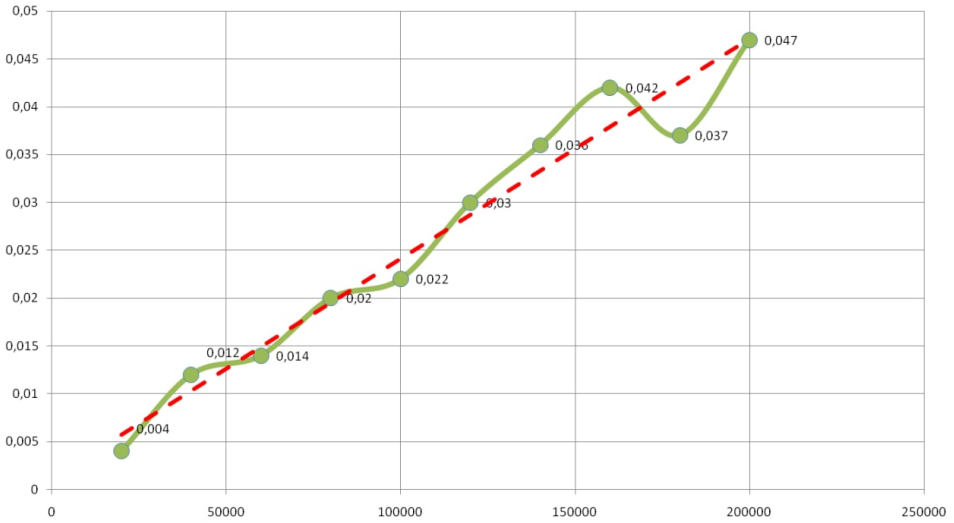
```
ErorCheck(end0);

end = int(end0);

createArray(arr, length, start, end);


cout << "\nСтворити частково сортований масив? (0/1): ";

cin >> partlySorted0;

if (partlySorted0 == "0" || partlySorted0 == "1") partlySorted =
 atoi(partlySorted0.c_str());

else

{

   cout << "\n    * -------- * ПОМИЛКА * -------- * \n";

   cout << " Неправильний варіант виконання програми!\n";

}


cout << "\nПерелік сум скарбів родин гномів у порядку їх отримання: \n";

printArray(arr, length);

cout << "\n";


cout << "\n-----------Менюшка----------";

cout << "\n0 - LSD - групування\n1 - MSD - групування";

cout << "\n---------------------------";

cout << "\nОберіть алгоритм сортування (0/1): ";

cin >> varik;


//float Start = clock();


switch (varik)
```

```
{
case 0:
    if (partlySorted == 0) cout << "\nLSD - ГРУПУВАННЯ:";
    radixSort_LSD(arr, length);
    break;
case 1:
    if (partlySorted == 0) cout << "\nMSD - ГРУПУВАННЯ:";
    radixSort_MSD(arr, length);
    break;
default:
    cout << "\n    * -------- * ПОМИЛКА * -------- * \n";
    cout << " Неправильний варіант виконання програми!\n";
    return 0;
}


if (partlySorted == 1)
{
    cout << "\nЗнайдено інформацію про суми скарбів ще N родин
 гномів\nN = ";
    cin >> addLength0;
    ErorCheck(addLength0);
    addLength = int(addLength0);
    int* array = new int[length + addLength]();
    for (int i = 0; i < length; i++)
        array[i] = arr[i];


    addElementsToArray(array, length, addLength, end);
```

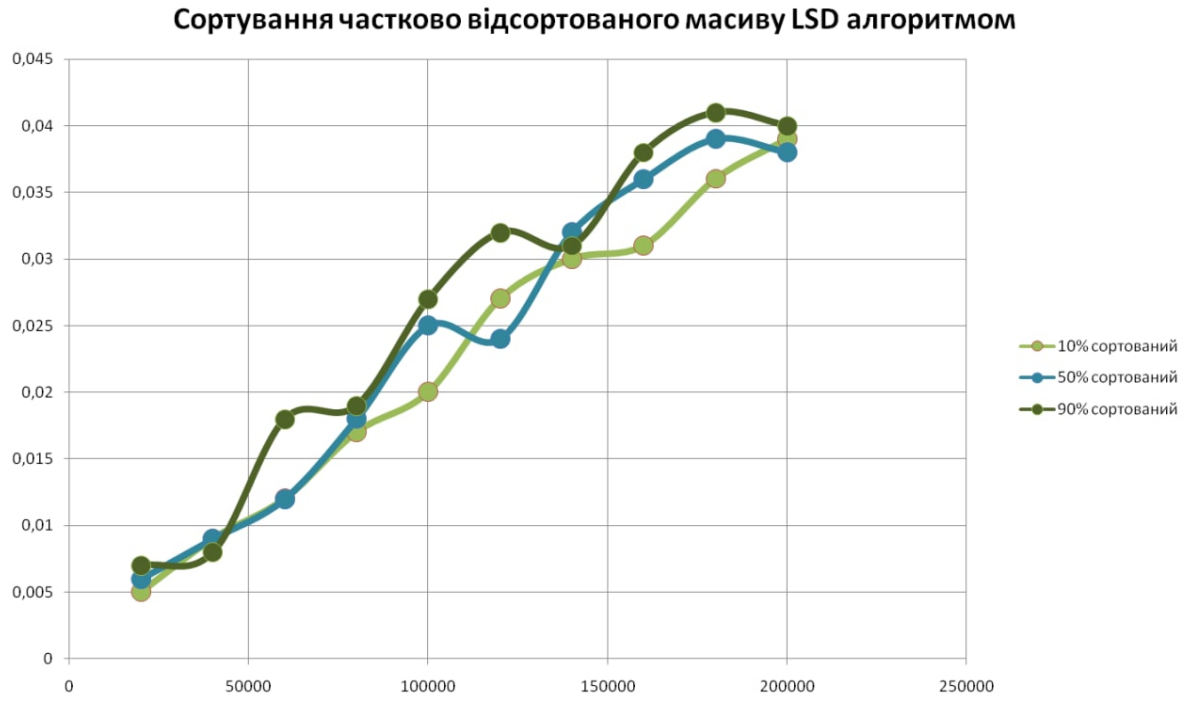
```
    cout << "\nНовий перелік скарбів леприкона: \n";
    printArray(array, length + addLength);

    //float Start = clock();
    if (varik == 0)
    {
        cout << "\n\nLSD - ГРУПУВАННЯ:";
      radixSort_LSD(array, length + addLength);
    }
    else
    {
        cout << "\n\nMSD - ГРУПУВАННЯ:";
        radixSort_MSD(array, length + addLength);
    }

    //cout << (clock() - Start) / CLOCKS_PER_SEC;
    cout << "\nВідсортований список, що подадуть королю: \n";
    printArray(array, length + addLength);
}
else
{
    //cout << (clock() - Start) / CLOCKS_PER_SEC;
    cout << "\nВідсортований список, що подадуть королю: \n";
    printArray(arr, length);}
cout << "\n\n\n";
return 0;
```

}

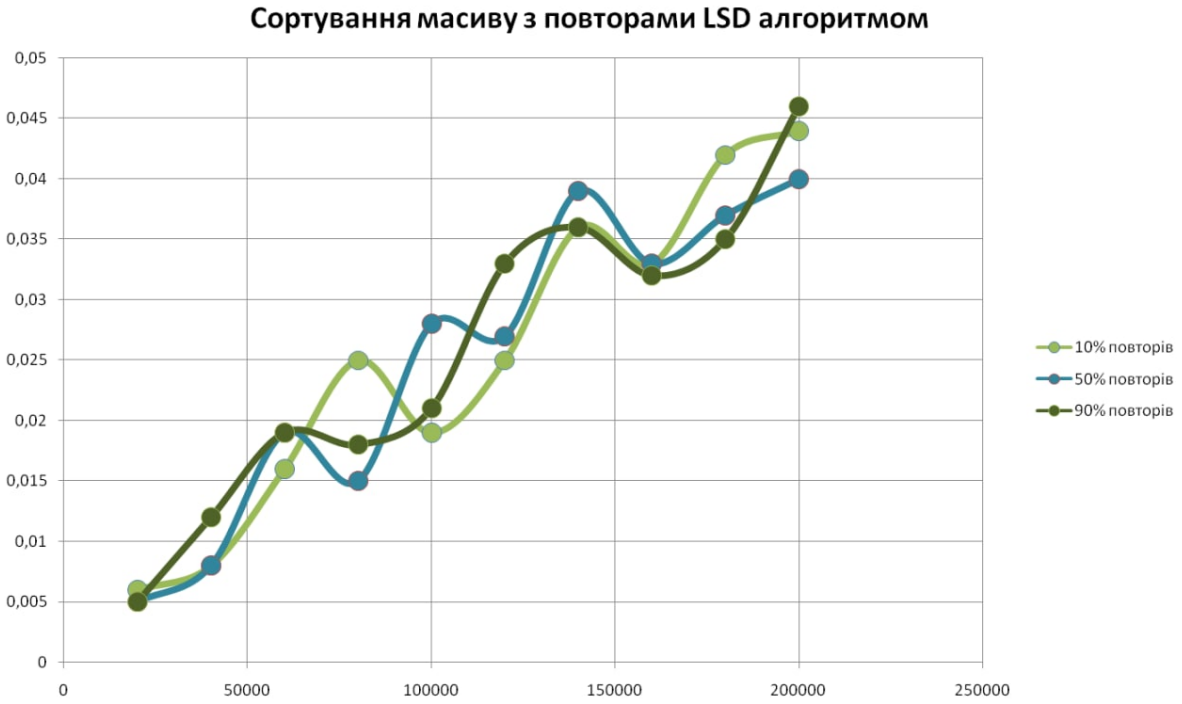
Перевірка обчислювальної складності програми:



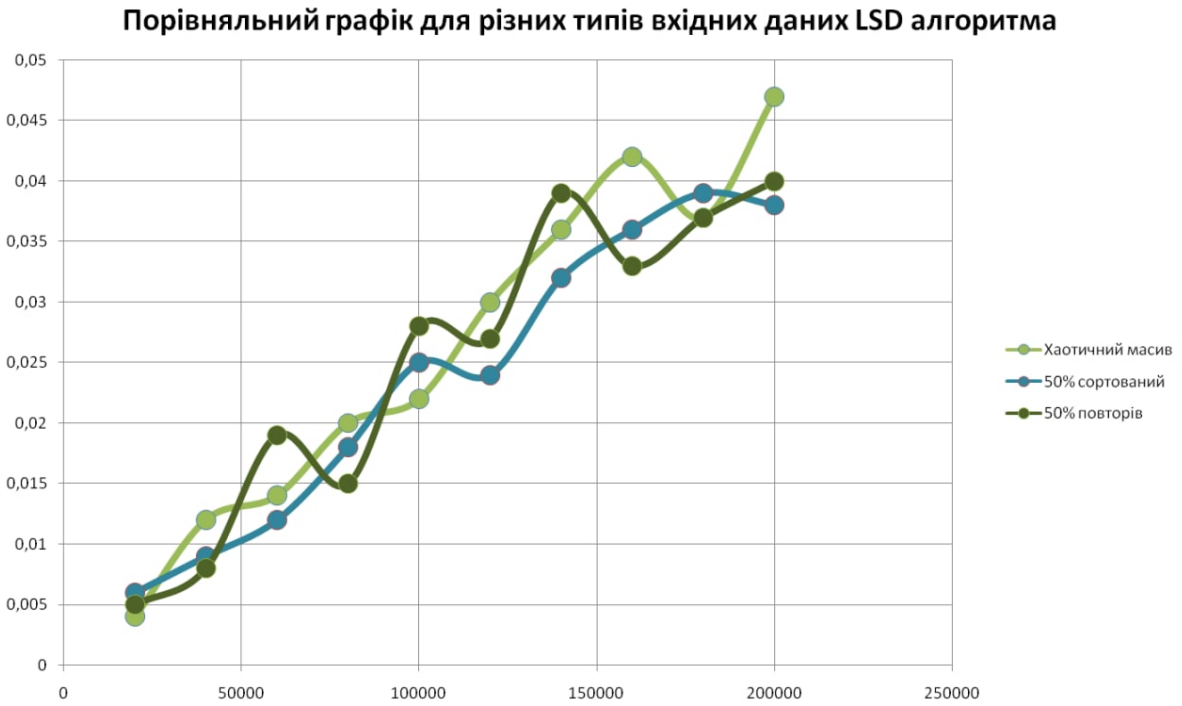
Діагр. 1. Сортування хаотичного масиву LSD - групуванням



Діагр. 2. Сортування частково відсортованого масиву LSD - групуванням

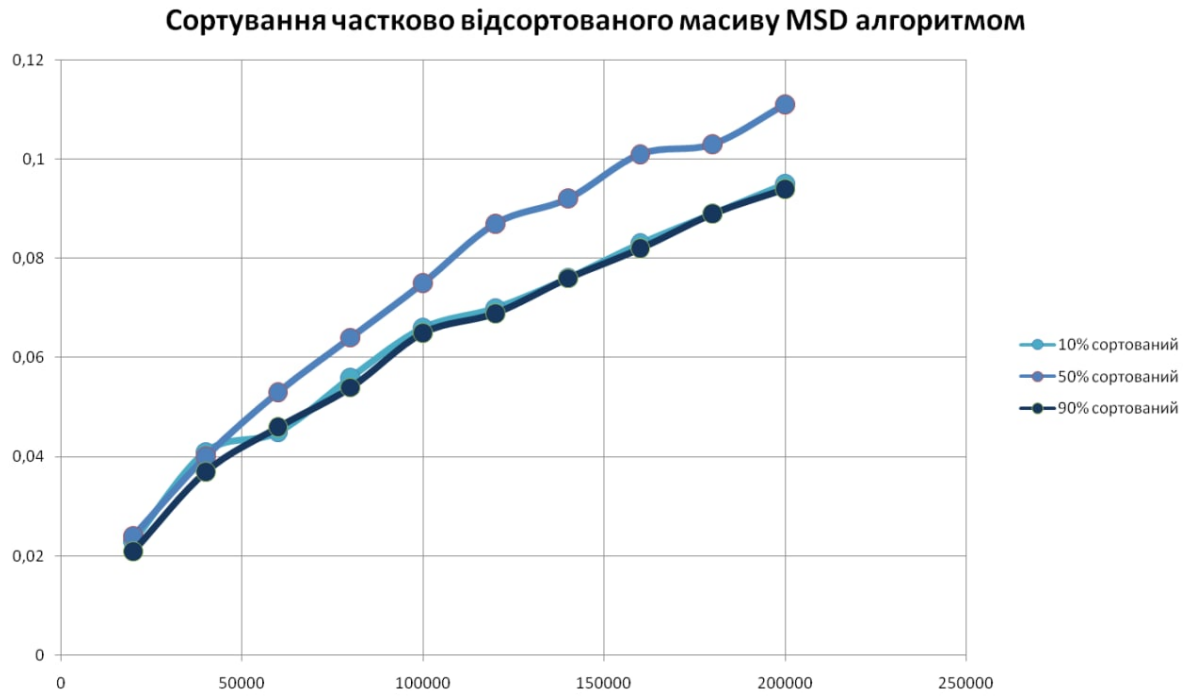


Діагр. 3. Сортування масиву з повторами LSD - групуванням

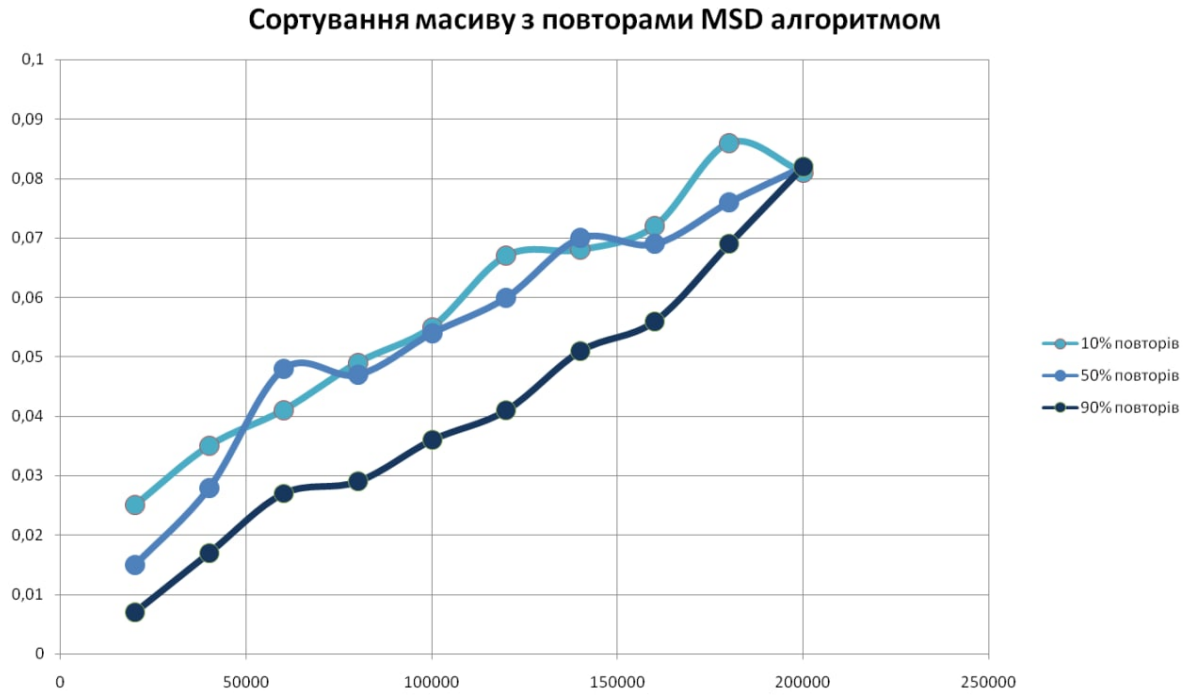


Діагр. 4. Порівняльна діаграма роботи алгоритму LSD – групування на різних типах вхідних даних

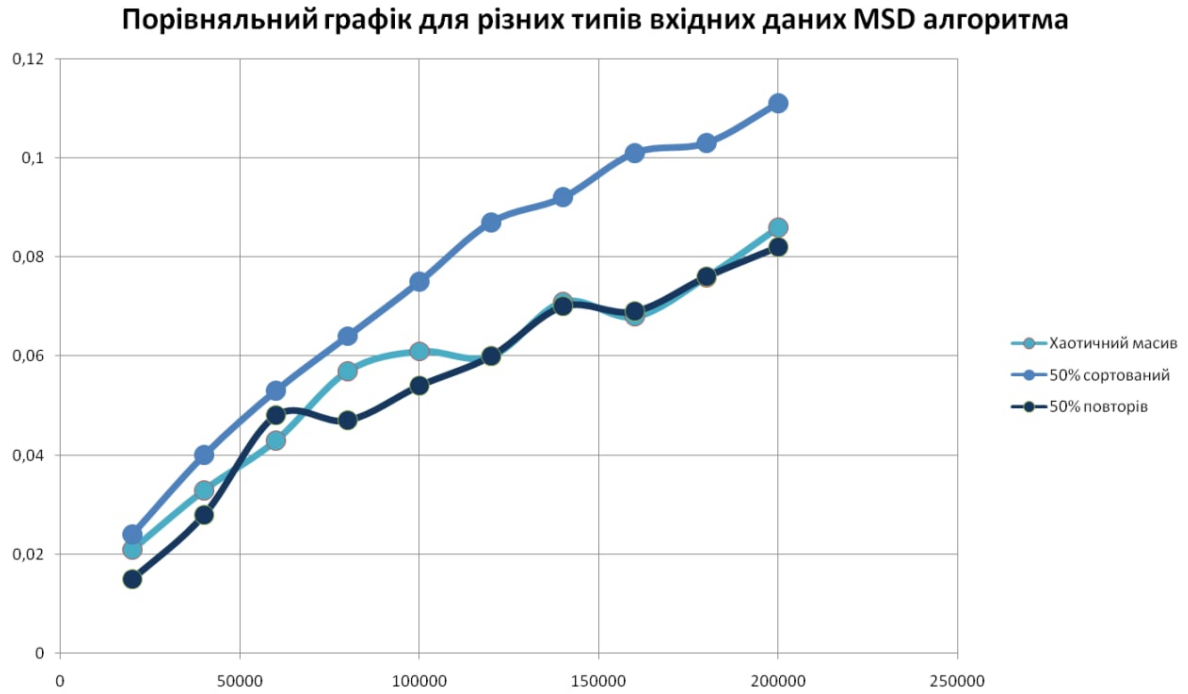
Діагр. 5. Сортування хаотичного масиву MSD - групуванням



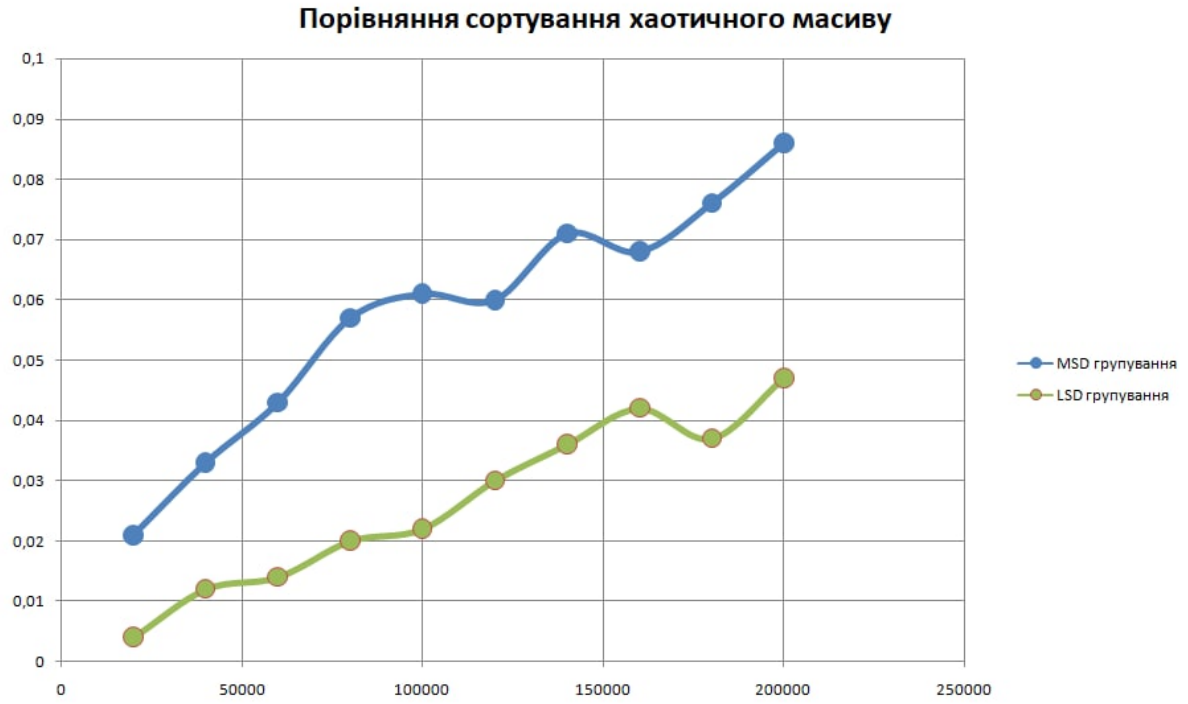
Діагр. 6. Сортування частково відсортованого масиву MSD - групуванням



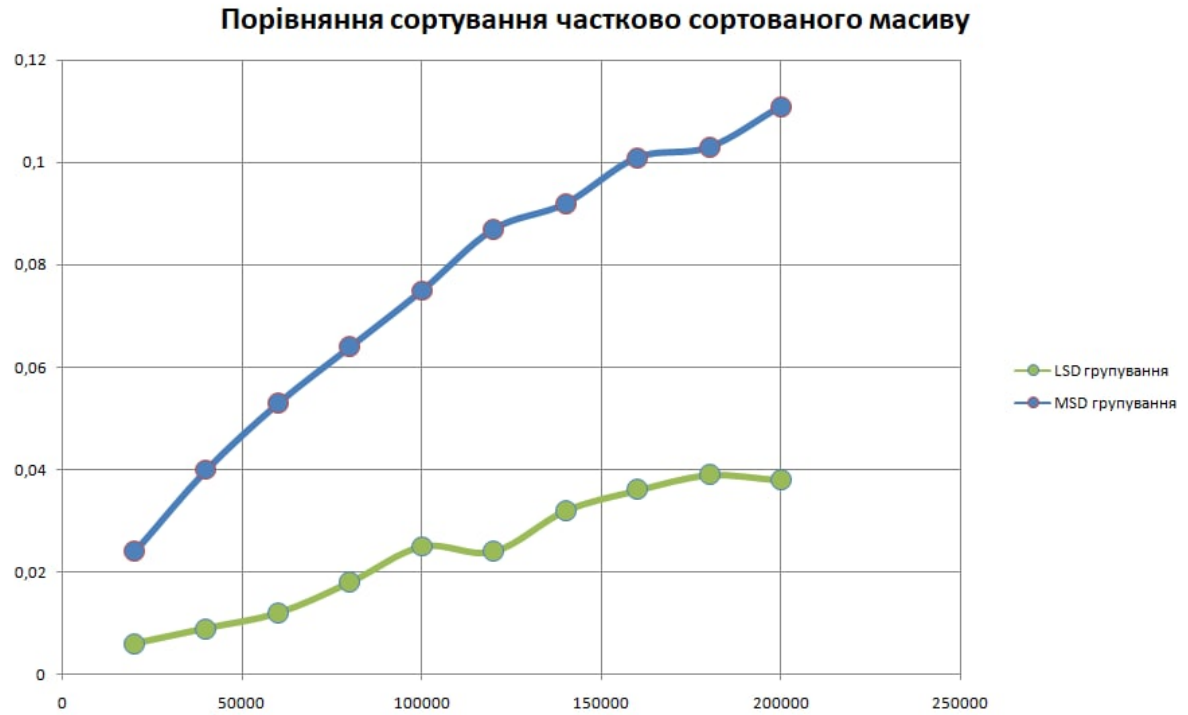
Діагр. 7. Сортування масиву з повторами MSD - групуванням



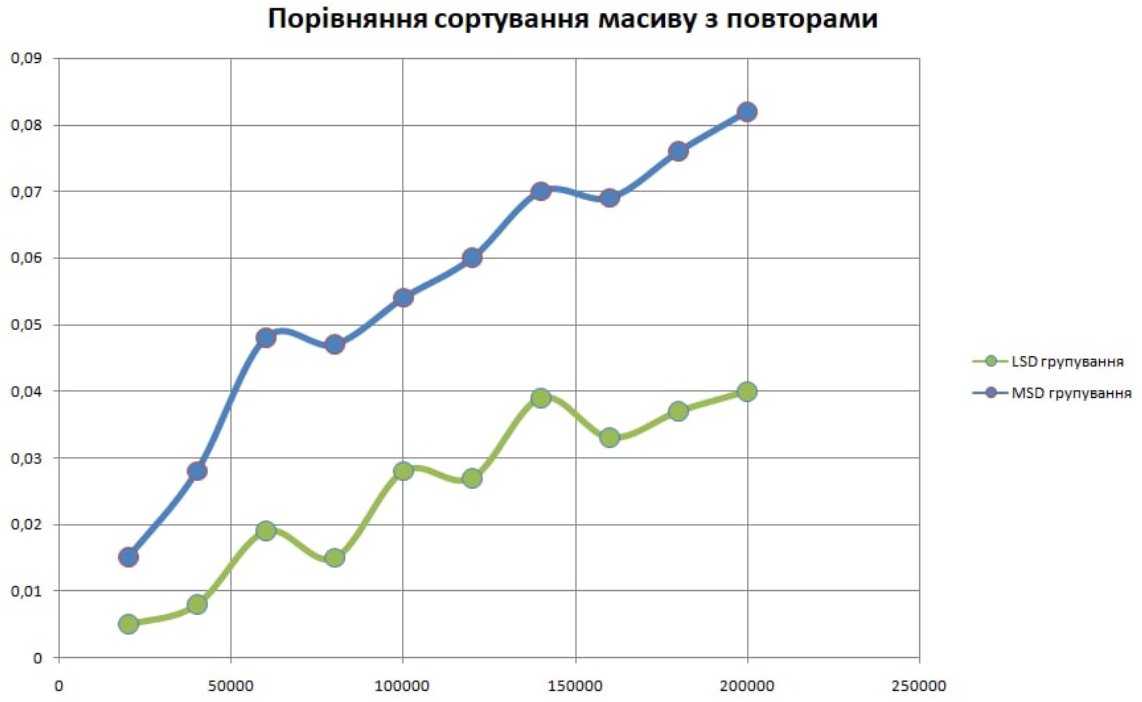
Діагр. 8. Порівняльна діаграма роботи алгоритму MSD – групування на різних типах вхідних даних



Діагр. 9. Порівняльна діаграма роботи алгоритмів на сортуванні хаотичного масиву



Діагр. 10. Порівняльна діаграма роботи алгоритмів на сортуванні частково відсортованого масиву



Діагр. 11. Порівняльна діаграма роботи алгоритмів на сортуванні масиву з повторами

Скріншоти роботи програми

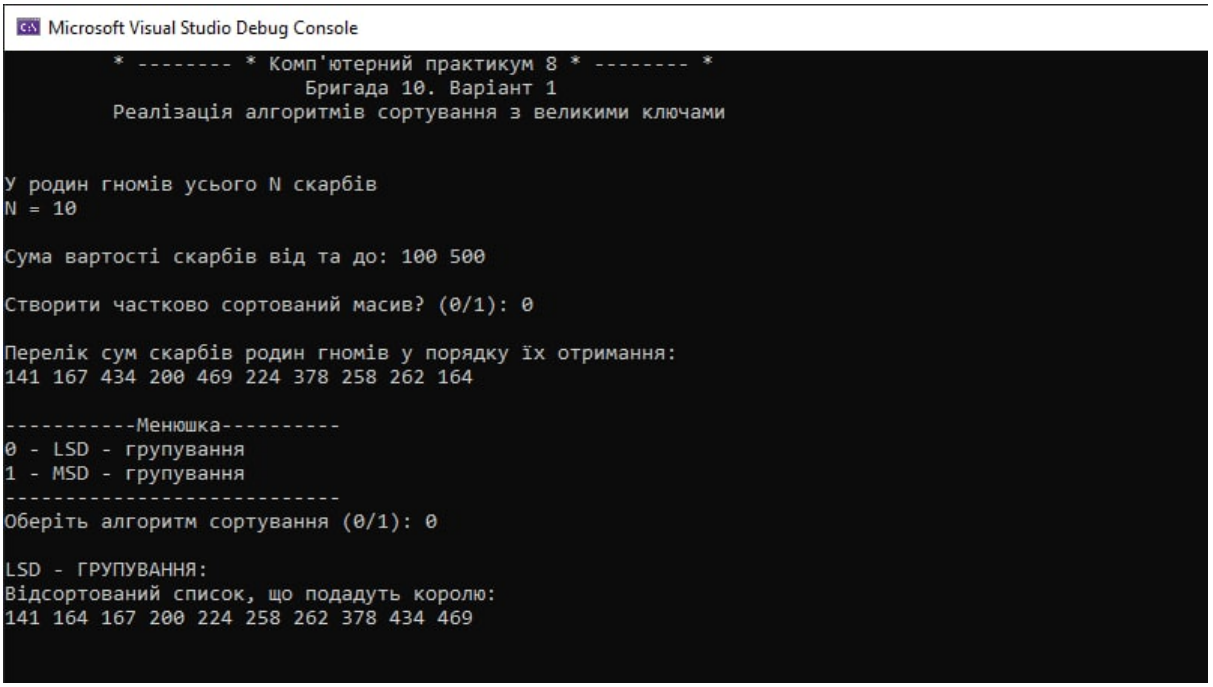


Рис. 4. LSD сортування з невідсортованим масивом

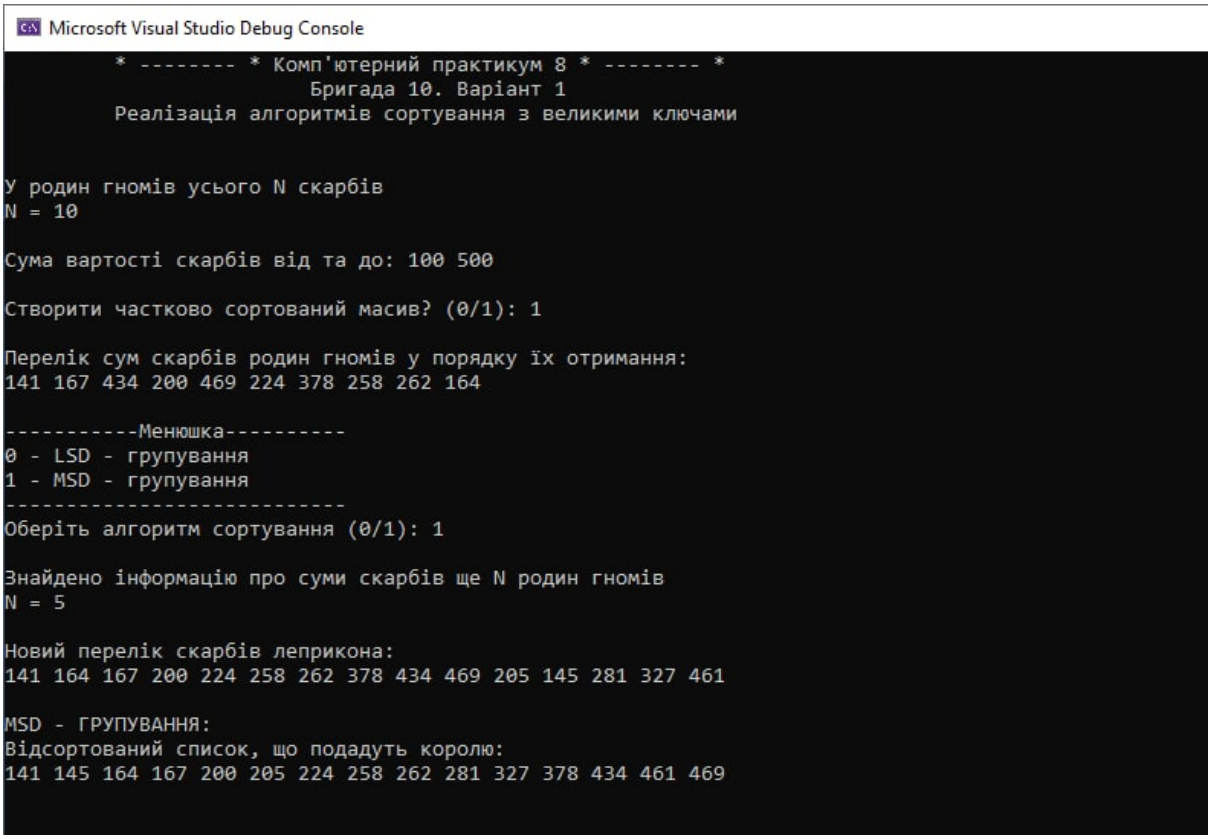


Рис. 5. MSD сортування з частково сортованим масивом

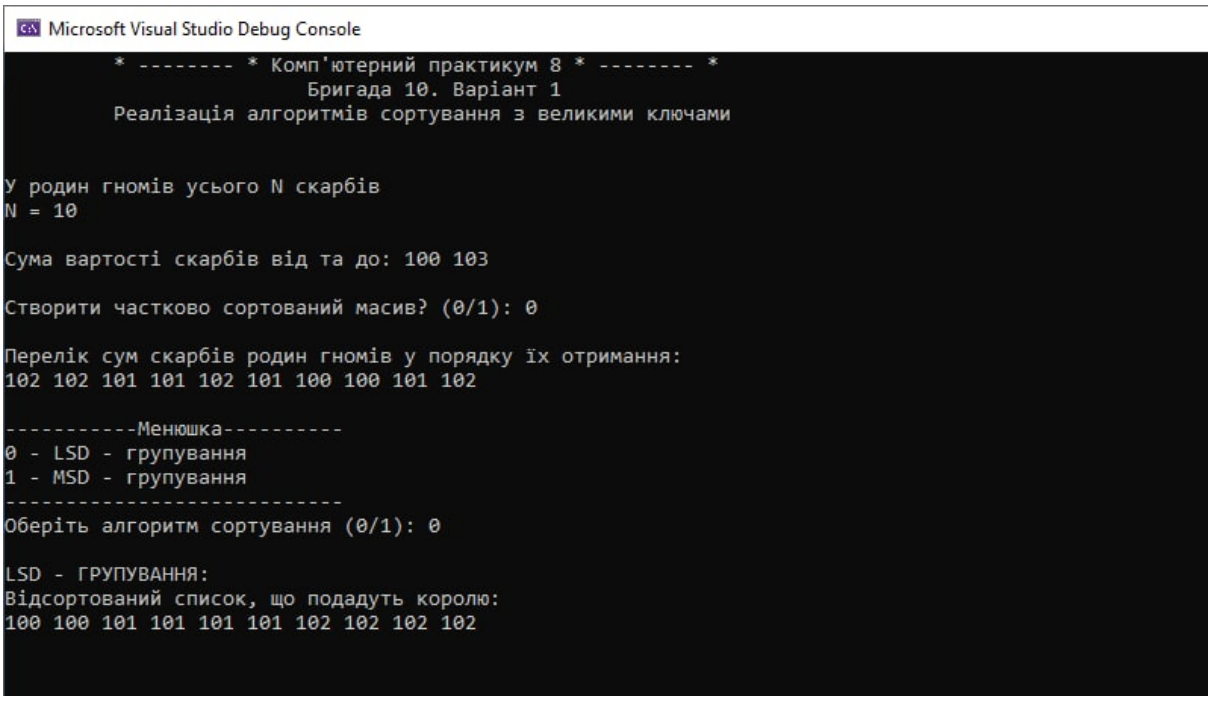


Рис. 6. LSD сортування масиву з повторами

Висновок щодо доцільності використання алгоритму

Під час виконання лабораторної роботи, нами було зроблено припущення щодо швидкодії кожного з запропонованих алгоритмів. За нашими судженнями, метод MSD сортування мав бути швидшим, за логікою реалізації, але кінцеві результати це спростували. При однаковій теоретичній складності, LSD вирізняється легшою реалізацією. Але, на відміну від MSD, алгоритм показав доволі нестабільну роботу при всіх типах вхідних даних.

Перевірка правильності програми:

Вхідні дані length0, end0, start0, addLength0	Результат	Призначення тесту
-1	Відомий	Перевірка, завершення некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
b	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
*	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
1.7	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
partlySorted0, varik (value=asdf)	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
partlySorted0, varik (value=*)	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)
partlySorted0, varik (value=3)	Відомий	Перевірка реакції на некоректні вхідні дані (додавання повідомлень про помилки, завершення роботи програми)

Табл. 4. Таблиця тестування програми

Висновок

Під час виконання лабораторної роботи нами були досліджені різні алгоритми сортування даних з великими ключами. Зроблене нами припущення не справдилось і метод LSD сортування показав найефективніші результати роботи, а також виявився найлегшим у реалізації. Теоретична складність кожного з алгоритмів була підтверджена практично. Результатами роботи є дослідження та аналіз, щодо ефективності роботи, а також побудова наочних порівняльних діаграм швидкодії роботи алгоритмів.

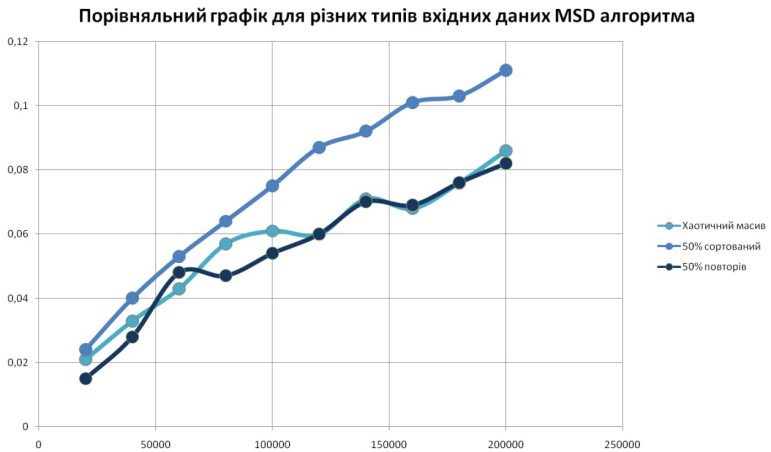
Відповіді на контрольні запитання

1. Перерахуйте та прокоментуйте переваги та недоліки порозрядного сортування.

Практика показує, що алгоритм порозрядного сортування для числових або рядкових ключів, штучно розбитих на розряди або на байти, все ж таки дещо поступається раніше описаним алгоритмам. Зате порозрядне сортування найшвидше, якщо ключ розпадається на компоненти природним чином.

2. Які найпоширеніші модифікації цього методу сортування? Вкажіть, на яких вхідних даних цей метод працює найкраще.

Найпоширенішими є реалізації LSD та MSD методів.



На діаграмах видно, що найкраще реалізації цього алгоритму показують себе на масивах з повторами.

3. Перерахуйте та прокоментуйте переваги та недоліки сортування кошиком.

Сортування комірками (англ. Bucket sort) — це стабільний алгоритм впорядкування, що доцільно використовувати, якщо вхідні дані розподілені рівномірно. В основі алгоритму лежить розподілення всіх елементів по скінченній кількості комірок. Кожна комірка впорядковується окремо іншим алгоритмом впорядкування або ж рекурсивно алгоритмом впорядкування комірками. Сортування комірками є узагальненням сортування підрахунком.

Алгоритм працює за час O(N), оскільки використовує додаткову інформацію про елементи.