Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №6

з дисципліни
«Алгоритми і структури даних»

Виконав:

Студент групи ІМ-41
Димура Ілля Олександрович
Номер у списку групи: 7

Перевірила:

Молчанова А. А.

Київ 2024

Завдання

1. Задано двовимірний масив (матрицю) цілих чисел A[m,n] або A[n,n], де m та n – натуральні числа (константи), що визначають розміри двовимірного масиву. Виконати сортування цього масиву або заданої за варіантом його частини у заданому порядку заданим алгоритмом (методом).

Сортування повинно бути виконано безпосередньо у двовимірному масиві «на тому ж місці», тобто без перезаписування масиву та/або його будь-якої частини до інших одно-або двовимірних масивів, а також без використання спискових структур даних.

2. Розміри матриці m та n взяти самостійно у межах від 7 до 10.

3. При тестуванні програми необхідно підбирати такі вхідні набори початкових значенньматриці, щоб можна було легко відстежити коректність виконання сортування і ця коректність була б протестована для всіх можливих випадків. З метою тестування дозволяється використовувати матриці меншого розміру.

При тестуванні програми необхідно перевірити коректність її роботи для трьох випадків початкового стану масиву або тієї його частини, яка сортується:

• вже відсортований початковий стан масиву;

• невідсортований початковий стан масиву (масив заповнений випадковими числами);

• обернено відсортований (до заданого за завданням) початковий стан масиву.

Варіант 7:

Задано двовимірний масив (матрицю) цілих чисел A[m,n]. Відсортувати окремо кожен рядок масиву методом швидкого сортування (методом Хоара) за не зменшенням.

Текст програми

```c
#include <stdio.h>


int partition(int *arr, const int low, const int high) {
   const int pivot = arr[(low + high) / 2];


   int i = low - 1;
   int j = high + 1;
```

```c
    while (1) {
        do { i++; } while (arr[i] < pivot);
        do { j--; } while (arr[j] > pivot);
        if (i >= j) return j;

        const int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}

void quicksort(int *arr, const int low, const int high) {
    if (low < high) {
        const int p = partition(arr, low, high);
        quicksort(arr, low, p);
        quicksort(arr, p + 1, high);
    }
}


int main(void) {
    int m, n;
    printf("Number of rows: ");
    scanf("%d", &m);
    printf("Number of columns: ");
    scanf("%d", &n);

    int matrix[m][n];
    for (int i = 0; i < m; i++) {
```

```
        printf("%d row of matrix (space separated): ", i + 1);
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }

    for (int i = 0; i < m; i++) {
        quicksort(matrix[i], 0, n - 1);
    }

    printf("\nSorted matrix:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d\t\t", matrix[i][j]);
        }
        printf("\n");
    }
}
```

Оточення:

```
→  lab06 gcc --version
Apple clang version 16.0.0 (clang-1600.0.26.4)
Target: arm64-apple-darwin24.1.0
Thread model: posix
```

```
→  lab06 gcc -o bin01 main.c
→  lab06 ls bin*
bin01
```

Тести програми:

```
Number of rows: 7
Number of columns: 7
1 row of matrix (space separated): -10 -5 0 1 3 6 8
2 row of matrix (space separated): 3 -1 7 -5 2 -6 4
3 row of matrix (space separated): 8 6 3 1 0 -5 -10
4 row of matrix (space separated): -3 1 -3 2 -3 2 -1
5 row of matrix (space separated): 6 -2 5 -3 4 -1 7
6 row of matrix (space separated): -15 20 -10 5 -3 0 12
7 row of matrix (space separated): 9 -8 2 -1 4 -7 3

Sorted matrix:
-10     -5      0       1       3       6       8
-6      -5      -1      2       3       4       7
-10     -5      0       1       3       6       8
-3      -3      -3      -1      1       2       2
-3      -2      -1      4       5       6       7
-15     -10     -3      0       5       12      20
-8      -7      -1      2       3       4       9
```

```
Number of rows: 7
Number of columns: 7
1 row of matrix (space separated): -20 -15 -10 -5 0 5 10
2 row of matrix (space separated): 12 -4 6 -8 3 -2 9
3 row of matrix (space separated): 15 10 5 0 -3 -7 -12
4 row of matrix (space separated): 25 -4 -2 -3 -9 10 12
5 row of matrix (space separated): 166 -2 5 -3 4 -1 7
6 row of matrix (space separated): 7 -4 5 -2 6 -3 8
7 row of matrix (space separated): 3 -11 5 7 -4 2 -9

Sorted matrix:
-20     -15     -10     -5      0       5       10
-8      -4      -2      3       6       9       12
-12     -7      -3      0       5       10      15
-9      -4      -3      -2      10      12      25
-3      -2      -1      4       5       7       166
-4      -3      -2      5       6       7       8
-11     -9      -4      2       3       5       7

Process finished with exit code 0
```

```
Number of rows: 7
Number of columns: 7
1 row of matrix (space separated): -1 2 -3 4 -5 6 -7
2 row of matrix (space separated): -50 10 -20 30 -10 40 -5
3 row of matrix (space separated): 1 0 1 0 1 0 1
4 row of matrix (space separated): -2 3 -5 7 -11 13 -17
5 row of matrix (space separated): 1 -1 2 -2 3 -3 4
6 row of matrix (space separated): -1000 50 -500 2000 -5 100 -10
7 row of matrix (space separated): 8 2 -3 8 -5 2 8

Sorted matrix:
-7      -5      -3      -1      2       4       6
-50     -20     -10     -5      10      30      40
0       0       0       1       1       1       1
-17     -11     -5      -2      3       7       13
-3      -2      -1      1       2       3       4
-1000       -500        -10     -5      50      100     2000
-5      -3      2       2       8       8       8

Process finished with exit code 0
```

```
Number of rows: 7
Number of columns: 7
1 row of matrix (space separated): -10 -5 0 3 6 8 12
2 row of matrix (space separated): -7 -3 0 1 4 6 10
3 row of matrix (space separated): -100 0 20 35 50 75 100
4 row of matrix (space separated): 1 2 3 4 5 6 7
5 row of matrix (space separated): 6 -5 3 0 -12 8 1
6 row of matrix (space separated): 14 -9 8 -2 11 7 -6
7 row of matrix (space separated): 3 -8 7 -15 5 -1 10

Sorted matrix:
-10     -5      0       3       6       8       12
-7      -3      0       1       4       6       10
-100        0       20      35      50      75      100
1       2       3       4       5       6       7
-12     -5      0       1       3       6       8
-9      -6      -2      7       8       11      14
-15     -8      -1      3       5       7       10

Process finished with exit code 0
```

```
Number of rows: 7
Number of columns: 7
1 row of matrix (space separated): -1 6 -9 3 5 -2 8
2 row of matrix (space separated): 12 8 6 3 0 -5 -10
3 row of matrix (space separated): 10 6 4 1 0 -3 -7
4 row of matrix (space separated): 100 75 50 35 20 0 -100
5 row of matrix (space separated): 7 6 5 4 3 2 1
6 row of matrix (space separated): 11 -22 33 -44 55 -66 77
7 row of matrix (space separated): -13 21 19 -7 -22 8 33

Sorted matrix:
-9      -2      -1      3       5       6       8
-10     -5      0       3       6       8       12
-7      -3      0       1       4       6       10
-100        0       20      35      50      75      100
1       2       3       4       5       6       7
-66     -44     -22     11      33      55      77
-22     -13     -7      8       19      21      33

Process finished with exit code 0
```

Висновок:

Для вирішення задачі був використаний алгоритм швидкого сортування (методом Хоара). За умовою сортування має бути за не зменшенням, тому змінювати стандартну імплементацію алгоритму не потрібно. Логарифмічна складність O(n log n), а також можливість змінювати напрямок сортування зробили цей алгоритм поширеним у сфері обробки даних. Обирати перший, останній чи середній елемент як опорний є не завжди самим ефективним шляхом, тому часто використовуються модифікації алгоритму з різною логікою визначення опорного елементу: наприклад, знаходять медіану. Програма була перевірена з використанням трьох перелічених в умові випадків: відсортований початковий стан масиву, масив заповнений випадковими числами і обернено відсортований масив. У випадку з обернено відсортованим масивом складність є найгіршою - $O(n^2)$