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-42
Туров Андрій Володимирович
номер у списку групи: 29

Перевірила:

Молчанова А. А.

Київ 2024

Завдання

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).

Варіант 29

Задано квадратну двовимірний масив (матрицю) цілих чисел $A[n;\ n]$. Відсортувати побічну діагональ масиву методом Шелла за незбільшенням.

Текст програми

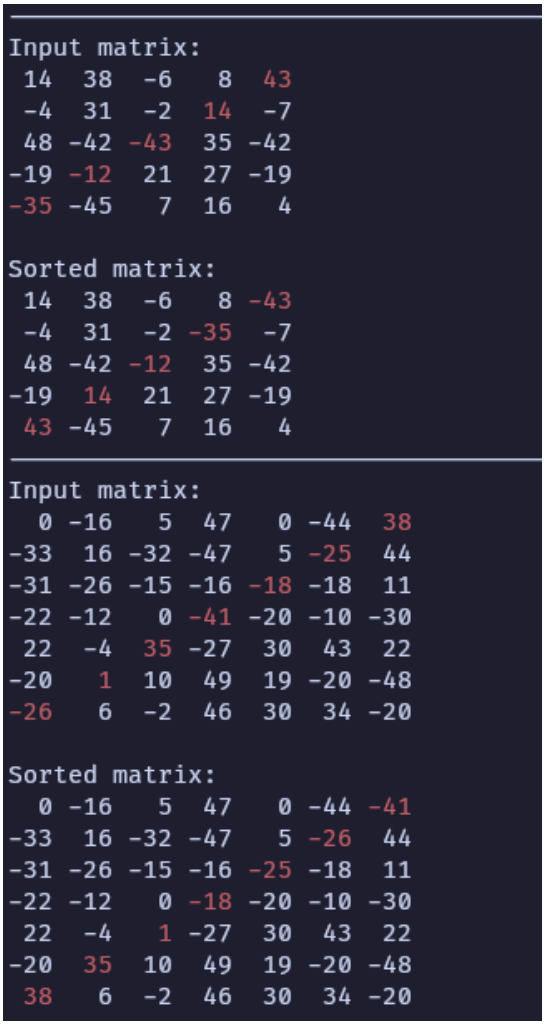
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

const int SHELL_GAPS[3] = {7, 3, 1};

void swap(int *a, int *b) {
	int temp = *a;
	*a = *b;
	*b = temp;
}

// Sorts the antidiagonal of a square matrix, ascension order (top-to-bottom)
void shell_sort(int size, int **array) {
	for (int g = 0; g < sizeof(SHELL_GAPS) / sizeof(int); g++) {
		int gap = SHELL_GAPS[g];
		for (int i = gap; i < size; i++) { // if gap is greater than size, we skip it
			for (int j = i - gap; j >= 0; j -= gap) {
				int *diagonal_j     = &array[j][size - j - 1];
				int *diagonal_j_next = &array[j + gap][size - (j + gap) - 1];

				if (*diagonal_j > *diagonal_j_next) { // Condition that defines order
					swap(diagonal_j, diagonal_j_next);
				}
			}
		}
	}
}

void print_matrix(int size, int **array) {
	for (int y = 0; y < size; y++) {
		for (int x = 0; x < size; x++) {
			if (x == size - y - 1) {
				printf("\033[31m%3d\033[0m ", array[y][x]); // print in red
			} else {
				printf("%3d ", array[y][x]);
```

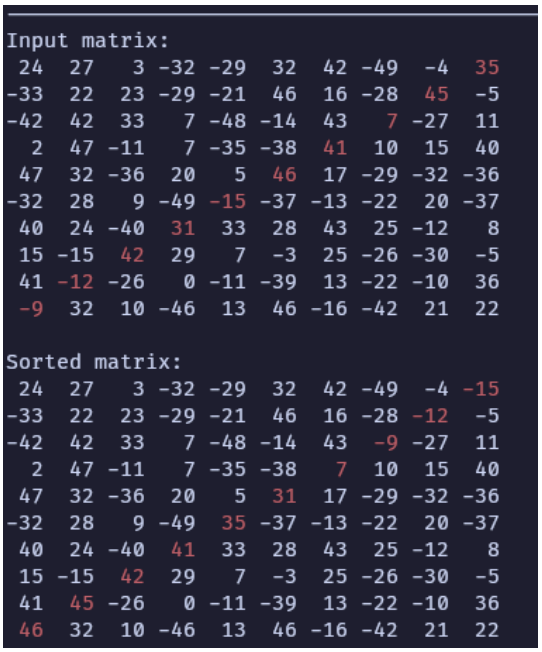
```
                }
            }
            printf("\n");
        }
}

int main(int argc, char **argv) {
        srand(clock());

        int size;
        int **matrix;

        if (argc < 3) {
            if (argc == 1) {
                printf("Enter the size of the matrix: ");
                scanf("%d", &size);
            } else {
                size = atoi(argv[1]);
            }

            matrix = malloc(sizeof(int *) * size);
            for (int y = 0; y < size; y++) {
                matrix[y] = malloc(size * sizeof(int));
                for (int x = 0; x < size; x++) {
                matrix[y][x] = rand() % 100 - 50;
                }
            }
        } else {
            size = atoi(argv[1]);
            if (argc < size * size + 2) {
                printf("Invalid argument count for matrix size %d\n", size);
                return 0;
            }

            matrix = malloc(sizeof(int *) * size);
            for (int i = 0; i < size*size; i++) {
                int y = i / size;
                int x = i % size;
                if (x == 0) {
                matrix[y] = malloc(size * sizeof(int));
                }
                matrix[y][x] = atoi(argv[i + 2]);
            }
        }

        printf("\nInput matrix:\n");
        print_matrix(size, matrix);

        shell_sort(size, matrix);

        printf("\nSorted matrix:\n");
        print_matrix(size, matrix);

       free(matrix);
        return 0;
```

```
}
```

Результати тестування програми

Рандомні значення:





Відсортовані випадки:

```
2. Sorted matrices:


Input matrix:
 -4  36  15   3 -39
 19 -34 -44 -35  19
-39 -17 -22 -19 -32
-34  -8 -38   0 -50
 34 -42  12  28   3

Sorted matrix:
 -4  36  15   3 -39
 19 -34 -44 -35  19
-39 -17 -22 -19 -32
-34  -8 -38   0 -50
 34 -42  12  28   3
```

```
Input matrix:
-42  -4  34  49 -37  12 -43
 26  15  49 -11 -20 -41  16
 -8 -48  -2 -50 -18 -32  24
 34 -38  -2 -16 -20  13 -50
-45  -7   6 -37  -9 -32  15
-44  27  23  32  47 -26  24
 29 -48 -10 -29   6  38 -29

Sorted matrix:
-42  -4  34  49 -37  12 -43
 26  15  49 -11 -20 -41  16
 -8 -48  -2 -50 -18 -32  24
 34 -38  -2 -16 -20  13 -50
-45  -7   6 -37  -9 -32  15
-44  27  23  32  47 -26  24
 29 -48 -10 -29   6  38 -29
```

```
Input matrix:
 25 -11  45  35  -2 -12  32  25 -10 -17  16 -45
-24  34  27 -28 -28  25 -31  47  30 -30 -40 -29
-45  14 -29  38 -35 -29  -9  -8  10 -36 -23 -42
-24 -40 -15 -32  -7   3 -18 -29 -32 -41  46 -39
-14  17 -42  18  37  15  39 -23 -20  13  34  -5
 34 -25  38  46 -37  17 -13  42  27  41  12  23
 46  45  -4  35   6  -8  46  43 -41   6 -37  -2
-28   5  -5   2  -6 -21  -1   4 -43  39   0  22
  7 -44 -34   6  -1  29 -39  -5 -24   8 -20  32
  2  28  14  12 -16  -9  12   6  46 -43  10  16
-13  36 -30  -6   1 -28 -34  10 -22  35  49  27
 38 -40  24  40 -30   6 -26 -25 -16   2  37  18

Sorted matrix:
 25 -11  45  35  -2 -12  32  25 -10 -17  16 -45
-24  34  27 -28 -28  25 -31  47  30 -30 -40 -29
-45  14 -29  38 -35 -29  -9  -8  10 -36 -23 -42
-24 -40 -15 -32  -7   3 -18 -29 -32 -41  46 -39
-14  17 -42  18  37  15  39 -23 -20  13  34  -5
 34 -25  38  46 -37  17 -13  42  27  41  12  23
 46  45  -4  35   6  -8  46  43 -41   6 -37  -2
-28   5  -5   2  -6 -21  -1   4 -43  39   0  22
  7 -44 -34   6  -1  29 -39  -5 -24   8 -20  32
  2  28  14  12 -16  -9  12   6  46 -43  10  16
-13  36 -30  -6   1 -28 -34  10 -22  35  49  27
 38 -40  24  40 -30   6 -26 -25 -16   2  37  18
```

Відсортовані навпаки випадки:

```
Input matrix:
 26  20 -31 -45  18 -22  38  41  23  37
 15  -2 -43  -1 -44  32  13  36  19  21
 22 -49 -11  28  19  42  35  17  -8 -26
 32  20  -3 -49  26  17  12  14   8   5
-22  25   5 -14  26  11  20  41  49 -11
 12  21  42   4   0  11  48  37 -49  -9
 13 -15  13  -3 -13 -11 -21 -31   5 -10
 24  34 -20  32  22  -6  -7  -8  37  -7
-16 -38  16 -24   6  18 -10   4 -43  -9
-48  21 -22  13  35  15 -48 -35 -14   8

Sorted matrix:
 26  20 -31 -45  18 -22  38  41  23 -48
 15  -2 -43  -1 -44  32  13  36 -38  21
 22 -49 -11  28  19  42  35 -20  -8 -26
 32  20  -3 -49  26  17  -3  14   8   5
-22  25   5 -14  26   0  20  41  49 -11
 12  21  42   4  11  11  48  37 -49  -9
 13 -15  13  12 -13 -11 -21 -31   5 -10
 24  34  17  32  22  -6  -7  -8  37  -7
-16  19  16 -24   6  18 -10   4 -43  -9
 37  21 -22  13  35  15 -48 -35 -14   8
```

```
Input matrix:
 26   3  33  34  16  40   9 -42  -2  38   9  49
-50  48 -12 -41  18   9   6  25 -38  26  30   4
-19 -29 -28 -42   1  -1 -15  29 -48  -3 -35 -29
-42  24 -21 -42 -36  38   5  16  -3  -4 -23 -44
-45  33 -17 -32 -38  16 -26  -7  37  -4 -47  40
 -3 -12 -29 -49 -41 -13 -12  19  13 -47  28 -20
 -7  35  46  31 -19 -17 -11  39   9 -25   9  21
 41 -15  17  28 -19 -28 -29  30  11  -8 -17 -28
-19 -42  41 -26 -39 -29  27 -43 -43 -25 -10  40
 -1  30 -28 -41   7  -8 -18  -2  27  49 -24 -40
-28 -28  -8 -15  -8  26   7  23 -14 -50  22  -3
-32  -1 -44  31  25  -3 -27  26  29 -43  35 -14

Sorted matrix:
 26   3  33  34  16  40   9 -42  -2  38   9 -32
-50  48 -12 -41  18   9   6  25 -38  26 -28   4
-19 -29 -28 -42   1  -1 -15  29 -48 -28 -35 -29
-42  24 -21 -42 -36  38   5  16 -26  -4 -23 -44
-45  33 -17 -32 -38  16 -26 -19  37  -4 -47  40
 -3 -12 -29 -49 -41 -13 -17  19  13 -47  28 -20
 -7  35  46  31 -19 -12 -11  39   9 -25   9  21
 41 -15  17  28  -7 -28 -29  30  11  -8 -17 -28
-19 -42  41  -3 -39 -29  27 -43 -43 -25 -10  40
 -1  30  -3 -41   7  -8 -18  -2  27  49 -24 -40
-28  30  -8 -15  -8  26   7  23 -14 -50  22  -3
 49  -1 -44  31  25  -3 -27  26  29 -43  35 -14
```

Висновки

Використав алгоритм сортування Шелла – модернізацію сортування за вставкою (insertion sort), яка ґрунтується на тому, що проміжне сортування кожного n-ного елемента масиву прискорює надалі сортування кожного першого елемента.

Написав bash-скрипт для тестування програми в трьох різних групах випадків: генерація випадкових значень, сортування уже відсортованих діагоналей, сортування відсортованих навпаки діагоналей.

Для цього модернізував код програми, дозволивши як і уведення розміру через `scanf`, а потім рандомну генерацію даних, так уведення його як аргумент в командному рядку, так і повне введення усієї матриці.

Для цього використав динамічний масив, який для якого процес резервує місце в пам’яті тільки тоді, коли відомий розмір майбутньої матриці.