Звіт

Автор: Момот Р. КІТ-119а

Дата: 04.05.2020

ЛАБОРАТОРНА РОБОТА № 11. АЛГОРИТМИ СОРТУВАННЯ ВИБОРОМ ТА ВКЛЮЧЕННЯМ

Мета: закріпити теоретичні знання та набути практичний досвід впорядкування набору статичних та динамічних структур даних.

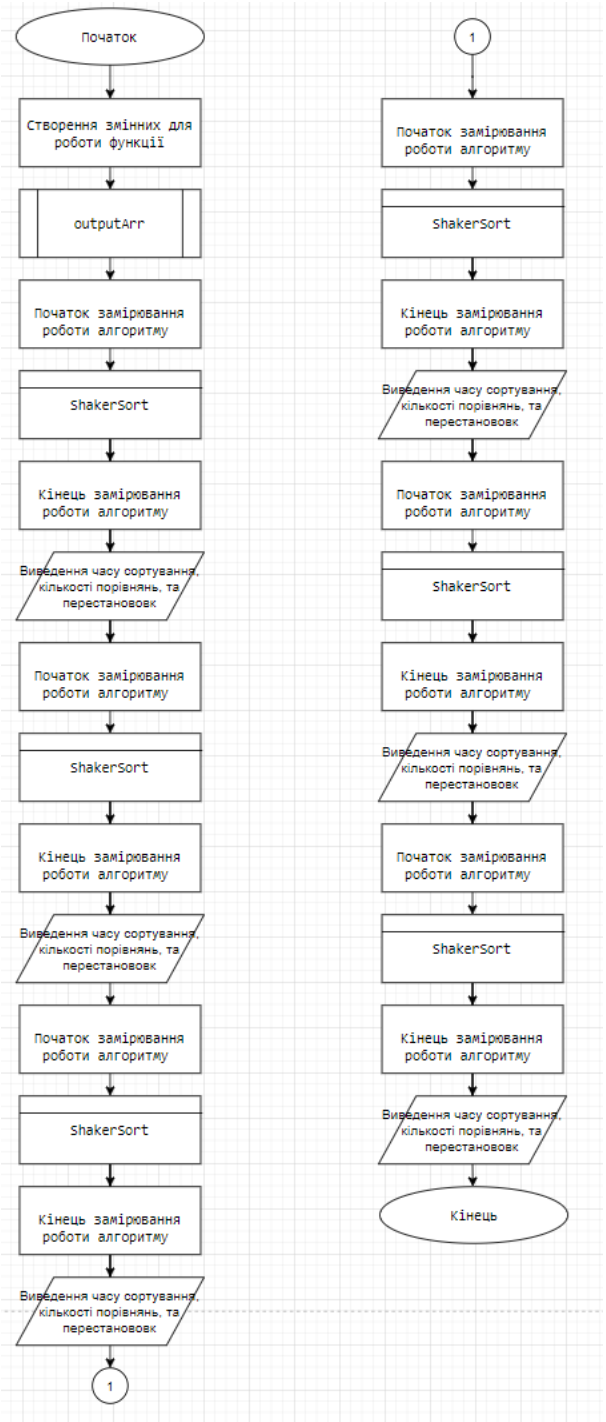
Індивідуальне завдання

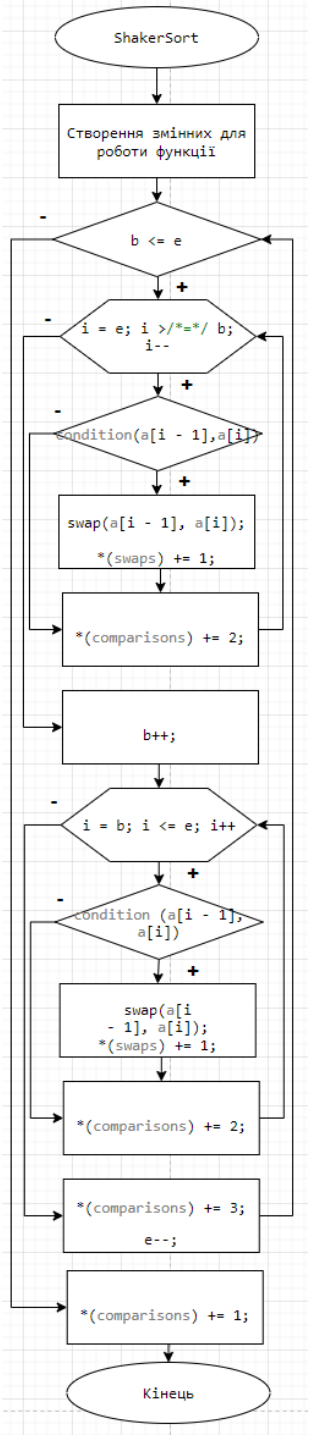
Написати програму, що реалізує сортування статичного та динамічного набору даних заданим способом. Визначити кількість порівнянь та обмінів для наборів даних, що містять різну кількість елементів (20, 1000, 5000, 10000, 50000). Оцінити час сортування. Дослідити вплив початкової впорядкованості набору даних (відсортований, відсортований у зворотньому порядку, випадковий). Всі отримані дані записати в таблицю. Зробити висновки.

Алгоритм сортування: шейкерне сортування.

Масиви даних: динамічний та статичний.

Блок-схема алгоритму програми





Текст програми

```
#define _CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, __FILE__, __LINE__)
#define new DEBUG_NEW

#include <iostream>
#include <clocale>
#include <chrono>
using namespace std;

typedef bool (comp)(const int&, const int&);
bool sortAsc(const int& a, const int& b) { return a > b; }
bool sortDesc(const int& a, const int& b) { return a < b; }

void CreateArray(int*, int*, int);
void OutputArray(int*, int);
void ShakerSort(int* , int, unsigned long long*, unsigned long long*, comp);

int main()
{
	setlocale(0, "");
	srand(time(NULL));
	const int SIZE = 50000;                          //размер массивов
	unsigned long long comparisons = 0;              //количество сравнений
	unsigned long long swaps = 0;                          //количество перестановок
	unsigned long long* pSwaps = &swaps;
	unsigned long long*pComparison = &comparisons;

	int array[SIZE];                                 //статичный массив
	int* pArray = array;
	int* dynamicArray = new int[SIZE];       //динамичный массив

	CreateArray(pArray,dynamicArray, SIZE);
	//OutputArray(pArray, SIZE);
	//OutputArray(dynamicArray, SIZE);

	cout << "Количество элементов: " << SIZE << endl;
	cout << "=========================" << endl;
	cout << "Не отсортированные данные" << endl;
	auto begin = chrono::steady_clock::now();
	ShakerSort(pArray, SIZE, pComparison, pSwaps, sortAsc);
	auto end = chrono::steady_clock::now();
	auto resultClock = chrono::duration_cast<chrono::milliseconds>(end - begin);
	cout << "\nСтатичный массив" << endl;
	cout << "Время сортировки: " << resultClock.count() << "мс" << endl;
	cout << "Количество сравнений: " << *pComparison << endl;
	cout << "Количество перестановок: " << *pSwaps << endl;

	*pComparison = 0, * pSwaps = 0;
	begin = chrono::steady_clock::now();
	ShakerSort(dynamicArray, SIZE, pComparison, pSwaps, sortAsc);
	end = chrono::steady_clock::now();
	resultClock = chrono::duration_cast<chrono::milliseconds>(end - begin);
	cout << "\nДинамический массив" << endl;
	cout << "Время сортировки: " << resultClock.count() << "мс" << endl;
	cout << "Количество сравнений: " << *pComparison << endl;
	cout << "Количество перестановок: " << *pSwaps << endl;

	cout << "=========================" << endl;

	*pComparison = 0, * pSwaps = 0;
	cout << "Отсортированные данные" << endl;
	begin = chrono::steady_clock::now();
```

```
	ShakerSort(pArray, SIZE, pComparison, pSwaps, sortAsc);
	end = chrono::steady_clock::now();
	resultClock = chrono::duration_cast<chrono::milliseconds>(end - begin);
	cout << "\nСтатичный массив" << endl;
	cout << "Время сортировки: " << resultClock.count() << "мс" << endl;
	cout << "Количество сравнений: " << *pComparison << endl;
	cout << "Количество перестановок: " << *pSwaps << endl;

	*pComparison = 0, * pSwaps = 0;
	begin = chrono::steady_clock::now();
	ShakerSort(dynamicArray, SIZE, pComparison, pSwaps, sortAsc);
	end = chrono::steady_clock::now();
	resultClock = chrono::duration_cast<chrono::milliseconds>(end - begin);
	cout << "\nДинамический массив" << endl;
	cout << "Время сортировки: " << resultClock.count() << "мс" << endl;
	cout << "Количество сравнений: " << *pComparison << endl;
	cout << "Количество перестановок: " << *pSwaps << endl;

	cout << "==========================" << endl;

	*pComparison = 0, * pSwaps = 0;
	cout << "Отсортированные данные в обратном порядке" << endl;
	begin = chrono::steady_clock::now();
	ShakerSort(pArray, SIZE, pComparison, pSwaps, sortDesc);
	end = chrono::steady_clock::now();
	resultClock = chrono::duration_cast<chrono::milliseconds>(end - begin);
	cout << "\nСтатичный массив" << endl;
	cout << "Время сортировки: " << resultClock.count() << "мс" << endl;
	cout << "Количество сравнений: " << *pComparison << endl;
	cout << "Количество перестановок: " << *pSwaps << endl;

	*pComparison = 0, * pSwaps = 0;
	begin = chrono::steady_clock::now();
	ShakerSort(dynamicArray, SIZE, pComparison, pSwaps, sortDesc);
	end = chrono::steady_clock::now();
	resultClock = chrono::duration_cast<chrono::milliseconds>(end - begin);
	cout << "\nДинамический массив" << endl;
	cout << "Время сортировки: " << resultClock.count() << "мс" << endl;
	cout << "Количество сравнений: " << *pComparison << endl;
	cout << "Количество перестановок: " << *pSwaps << endl;

	delete[] dynamicArray;
	if (_CrtDumpMemoryLeaks())
		cout << "\nУтечка памяти обнаружена." << endl;
	else
		cout << "\nУтечка памяти отсутствует." << endl;
}

void CreateArray(int* array, int* dArray, int size)
{
	int value;
	for (size_t i = 0; i < size; i++)
	{
		value = rand();
		*(array + i) = value;
		*(dArray + i) = value;
	}
}
void OutputArray(int* array, int size)
{
	cout << endl;
	for (size_t i = 0; i < size; i++)
	{
		cout << *(array + i) << "  ";
	}
	cout << endl;
```

```
}
void ShakerSort(int* a, int N, unsigned long long* comparisons, unsigned long long* swaps, comp
condition)
{
	int i, b = 0, e = N - 1;

	while (b <= e)
	{
		for (i = e; i >/*=*/ b; i--)
		{
			if (condition(a[i - 1],a[i]))
			{
				swap(a[i - 1], a[i]);
				*(swaps) += 1;
			}
			*(comparisons) += 2;

		}
		b++;

		for (i = b; i <= e; i++)
		{
			if (condition (a[i - 1], a[i]))
			{
				swap(a[i - 1], a[i]);
				*(swaps) += 1;
			}
			*(comparisons) += 2;
		}
		*(comparisons) += 3;
		e--;
	}
	*(comparisons) += 1;
}
```

Результат роботи програми

```
Количество элементов: 5000
==========================
Не отсортированные данные

Статичный массив
Время сортировки: 610мс
Количество сравнений: 25007501
Количество перестановок: 6264983

Динамический массив
Время сортировки: 624мс
Количество сравнений: 25007501
Количество перестановок: 6264983
==========================
Отсортированные данные

Статичный массив
Время сортировки: 224мс
Количество сравнений: 25007501
Количество перестановок: 0

Динамический массив
Время сортировки: 210мс
Количество сравнений: 25007501
Количество перестановок: 0
==========================
Отсортированные данные в обратном порядке

Статичный массив
Время сортировки: 910мс
Количество сравнений: 25007501
Количество перестановок: 12497128

Динамический массив
Время сортировки: 915мс
Количество сравнений: 25007501
Количество перестановок: 12497128

Утечка памяти отсутствует.
```

Результати тестування алгоритма сортування даних

Статичний масив					
Відсортований набір даних	**20**	**1000**	**5000**	**10000**	**50000**
Кількість порівнянь	431	1001501	25007501	100015001	2500075001
Кількість пересилань	0	0	0	0	0
Час сортування	0,004	8	219	812	21677
Відсортований в зворотньому порядку	**20**	**1000**	**5000**	**10000**	**50000**
Кількість порівнянь	431	1001501	25007501	100015001	2500075001
Кількість пересилань	190	499488	12497113	49993490	1249936525
Час сортування	0,0146	36	919	3701	92096
Випадковий набір даних	**20**	**1000**	**5000**	**10000**	**50000**
Кількість порівнянь	431	1001501	25007501	100015001	2500075001
Кількість пересилань	79	245349	6101309	24753085	623070648
Час сортування	0,0091	22	625	2361	58318
Динамічний масив					
Відсортований набір даних	**20**	**1000**	**5000**	**10000**	**50000**
Кількість порівнянь	431	1001501	25007501	100015001	2500075001
Кількість пересилань	0	0	0	0	0
Час сортування	0,0042	8	212	819	21504
Відсортований в зворотньому порядку	**20**	**1000**	**5000**	**10000**	**50000**
Кількість порівнянь	431	1001501	25007501	100015001	2500075001
Кількість пересилань	190	499488	12497113	49993490	1249936525
Час сортування	0,0149	37	913	3677	92096
Випадковий набір даних	**20**	**1000**	**5000**	**10000**	**50000**
Кількість порівнянь	431	1001501	25007501	100015001	2500075001
Кількість пересилань	79	245349	6101309	24753085	623070648
Час сортування	0,0091	23	690	2315	58137

*Час пошуку вказан у наносекундах.

Висновок

У результаті роботи програми було розроблено програму, яка базується на динамічному та статичному масивах та включає в себе алгоритм шейкерного сортування. Як можна бачити із таблиці вище, шейкерне сортування працює швидше на статичому масиві даних.

Відповіді на питання

1. Назвіть всі алгоритми сортування порядку $O(n^2)$, $O(n)$, $O(\log(n))$.

$O(n^2)$: Сортування вибіркою, обмінна сортування вибіркою, сортування вибіркою мах і min елеменов, бульбашкова сортування (класична), бульбашкова сортування з ознакою, бульбашкова сортування із запам'ятовуванням місця останньої перестановки, шейкер-сортування, дурна сортування, сортування простими вставками, обмінна сортування вставками, бульбашкова сортування вставками, сортування вставками з бар'єром, сортування Шелла

$O(n)$: Дурне сортування, блочне сортування, млинцеве сортування.

$O(\log(n))$: Сортування методом двійкового включення, сортування злиттям, плавне сортування, пірамідальне сортування, терпляче сортування.

2. Назвіть всі алгоритми сортувань, що відносяться до групи бульбашкових. Чим характеризуються ці алгоритми?

У бульбашковому сортуванні порівнюються та переміщуються сусідні елементи по колу, поки усі дані не будуть відсортовані.

Існують такі типи бульбашкових алгоритмів сортування: бульбашкове сортування (класична), бульбашкове сортування з ознакою, бульбашкове сортування із запам'ятовуванням місця останньої перестановки, шейкер-сортування, сортування Шелла.

3. Вкажіть всі можливі модифікації алгоритму сортування Шейкера. Напишіть одну з вказаних процедур Шейкер – сортування.

```
void shakerSort(std::vector<int>*arr) {
    int buff;
    int left = 0;
    int right = (*arr).size() - 1;
    do {
            for (int i = left; i < right; i++) {
                    if ((*arr)[i] > (*arr)[i + 1]) {
                            buff = (*arr)[i];
                            (*arr)[i] = (*arr)[i + 1];
                            (*arr)[i + 1] = buff;
                    }
            }
            right--;
            for (int i = right; i > left; i--) {
                    if ((*arr)[i] < (*arr)[i - 1]) {
                            buff = (*arr)[i];
                            (*arr)[i] = (*arr)[i - 1];
                            (*arr)[i - 1] = buff;
                    }
            }
            left++;
    } while (left < right);
 }
```

4. В алгоритмі сортування Шелла шаг *d* визначається як *N % 2*. А чи можна визначати *d* інакше, чому запропоновано саме такий алгоритм?

Помилка в запитанні, визначається як d = N /2.

- d=9*2 i -9*2 i/2 +1 якщо i парне, якщо ні - d=8*2 i -6*2 (i+2)/2 +1;
- d = {1,4,10,23,57,132,301,701,1705} емпірична послідовність Марціна Ціура;
- емпірична послідовність заснована на числах Фібоначчі.
- Послідовність Прата;
- Послідовність Седжвіка;

5. Чим пояснюється наявність великої кількості алгоритмів сортування?

Кожен алгоритм підходить для своїх типів задач. Деякі алгоритми швидко працюють на даних маленьких розмірів, якісь легші у реалізації. Також алгоритми сортування швидко розвиваються, що впливає на появу нових алгоритмів.

6. Назвіть відомі вам алгоритми сортувань, час виконання яких не залежить від ступеню впорядкованості вхідного масиву.

Сортування простою вибіркою, обмінне сортування вибіркою, сортування вибіркою мах і min елементів, бульбашкове сортування (класичне), сортування злиттям.

7. За логікою роботи на які групи поділяють всі алгоритми сортувань?

Алгоритми поділяються на 4 групи за логікою роботи:

Сортування вибіркою;

Сортування включенням;

Сортування розподіленням;

Сортування злиттям.

8. Назвіть алгоритми сортівання, які чутливі до ступеня впорядкованості вхіного набору даних.

Шейкерне сортування, сортування вставками, дурне сортування.

9. Які алгоритми соотувань вимагають дві області пам’яті: для вхідної та вихідної множини даних?

Сортування вставками, сортування підрахунком, блочне сортування, бетонне сортування, Timsort.

10.Як буде виглядати масив {2,5,4,1,3} при бульбашковому сортуванні за зростанням після першого кроку.

Після першого кроку дані не зміняться, бо на першому кроці будуть перевірятися перший та другий (2,5) елементи, які не треба переміщувати.

11.Реалізація якого алгоритма наведена в наступному фрагменті програмного кода?

```
for (i = 0; i < N; i++)
  for (k = i, j = i + 1; j < N; j++)
        if (a[j] < a[k])
        {
              k = j;
              swap(a[k], a[i]);
        }
```

Це сортування вибором.

12. За скільки кроків методом бульбашкового сортування з ознакою буде впорядковано за зростанням масив {2, 5, 6, 7, 7, 8, 9}?

За 6 кроків метод завершить свою роботу, бо елементи масива не треба переміщувати.

13. Які алгоритми сортувань називають стійкими?

Метод сортування називається стійким, якщо при його застосуванні не змінюється відносне положення записів з рівними значеннями ключа.

14. Які операції (окрім операцій порівняння) мають великий вплив на час виконання сортування?

Операція перестановки елементів має великий вплив на час виконання сортування.

15. Які фактори впливають на вибір алгоритма сортування?

Порядок алгоритма, необхідний ресурс пам'яті, початкова впорядкованість вхідної множини, часові характеристики операцій, складність алгоритма.