Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Звіт

з лабораторної роботи №8

«Алгоритми сортування»

з дисципліни «Алгоритми та структури даних»

Виконала:

ст.гр. ІП-22-4

Томин Є.В.

Перевірив:

Григорчук Л. І.

Івано-Франківськ

2023

**Тема.** Алгоритми сортування

**Мета:** познайомитися з алгоритмами сортування та бінарним пошуком.

**Теоретичні відомості:**

Сортування бульбашкою (*bubble sort*) складається з повторюваних проходів по сортованому масиву. За кожен прохід елементи послідовно порівнюються попарно, і якщо порядок у парі неправильний, виконується обмін елементів. Проходи по масиву повторюються N-1 раз або доти, поки на черговому проході не з’ясується, що обміни більше не потрібні, що означає – масив відсортований. При кожному проході алгоритму по внутрішньому циклу черговий найбільший елемент масиву ставиться на своє місце в кінці масиву поруч з попереднім найбільшим елементом, а найменший елемент переміщається на одну позицію до початку масиву («плине» до потрібної позиції, як бульбашка у воді, звідси і назва алгоритму).

Сортування включенням (*insertion sort*) зручне для сортування коротких послідовностей. Саме в такий спосіб зазвичай сортують карти: тримаючи в лівій руці вже впорядковані карти і взявши правою рукою чергову кар ту, ми вставляємо її в потрібне місце, порівнюючи з наявними і йдучи справа наліво.

*Сортування злиттям* основане на принципі «розділяй і володарюй». Спочатку ми розбиваємо масив на дві половини меншого розміру. Потім ми сортуємо кожну з половин окремо. Після цього нам залишається з’єднати два впорядкованих масиви половинного розміру в один. Рекурсивне розбиття задачі на менші відбувається доти, поки розмір масиву не дійде до одиниці (будь-який масив довжини можна вважати впорядкованим).

Нетривіальною частиною є з’єднання двох упорядкованих масивів в один. Воно виконується за допомогою допоміжної процедури *Merge(A,p,q,r)*. Параметрами цієї процедури є масив A і числа *p,q,r,* що вказують на межі об’єднуваних ділянок. Процедура передбачає, що *p<q<r* та що ділянки *A[p..q] і A[q+1,r]* вже відсортовані, і зливає (*merges*) їх до однієї ділянки *A[p..r].*

*Сортування купою (пірамідальне сортування)* базується на використанні сортувального дерева (бінарної купи). Сортувальне дерево – це та ке двійкове дерево, у якого виконані такі умови.

1. Кожен листок має глибину або d, або d-1, де d – максимальна глибина дерева.

2. Значення в будь вершини не менше (інший варіант – не більше) від значення її нащадків.

Зручна структура даних для сортувального дерева – такий масив A, що A[0]– елемент в корені, а нащадки елемента *A[i]* є *A[2i+1]* і *A[2i+2].*

Алгоритм сортування буде складатися з двох основних кроків: 1. Побудова сортувального дерева, такого, щоб кожний нащадок був меншим за предка. Цей крок вимагає *0(n)* операцій.

2. Видалення елементів з кореня по одному за раз і перебудова дерева. Тобто на першому кроці обмінюємо A[0] і A[n-1], перетворюємо *A[0], A[1],…,A[n-2]* в сортувальне дерево. Потім переставляємо *A[0]* і *A[n-2]* , перетворюємо *A[0], A[1],…,A[n-3]* в сортувальне дерево. Процес продовжується доти, поки в сортувальному дереві не залишиться один елемент. Тоді *A[0], A[1],…,A[n-1]*– упорядкована послідовність.

**Завдання до лабораторної роботи 8**

Розробити програму, яка читає з клавіатури:

1) числа *N, M (1<N, M<256*)

2) послідовність N ключів (цілих або дійсних чисел залежно від варіанту завдання);

3) послідовність M ключів.

Програма зберігає першу послідовність до масиву та виконує сортування. Потім програма виводить відсортовану послідовність на екран та виконує бінарний пошук кожного елемента другої послідовності : для кожного повідомити, чи є він у першій послідовності, а якщо є, то на якому місці.

***Варіант 1:*** сортування бульбашкою.

**Виконання**

#include <iostream>

#include <algorithm>

// Бінарний пошук у відсортованому масиві

int binarySearch(const int sortedArray[], int size, int key) {

int left = 0;

int right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (sortedArray[mid] == key)

return mid;

else if (sortedArray[mid] < key)

left = mid + 1;

else

right = mid - 1;

}

return -1; // Повертає -1, якщо ключ не знайдено

}

// Сортування "бульбашкою"

void bubbleSort(int array[], int size) {

for (int i = 0; i < size - 1; ++i) {

for (int j = 0; j < size - i - 1; ++j) {

if (array[j] > array[j + 1]) {

// Міняємо місцями, якщо потрібно

std::swap(array[j], array[j + 1]);

}

}

}

}

int main() {

int N, M;

// Введення розмірності масивів

std::cout << "Введіть розмірність першого масиву (N): ";

std::cin >> N;

std::cout << "Введіть розмірність другого масиву (M): ";

std::cin >> M;

// Створення масивів

int array1[N];

int array2[M];

// Введення елементів першого масиву

std::cout << "Введіть " << N << " елементів першого масиву:\n";

for (int i = 0; i < N; ++i) {

std::cin >> array1[i];

}

// Введення елементів другого масиву

std::cout << "Введіть " << M << " елементів другого масиву:\n";

for (int i = 0; i < M; ++i) {

std::cin >> array2[i];

}

// Сортування першого масиву "бульбашкою"

bubbleSort(array1, N);

// Виведення відсортованого першого масиву

std::cout << "Відсортований перший масив:\n";

for (int i = 0; i < N; ++i) {

std::cout << array1[i] << " ";

}

std::cout << "\n";

// Бінарний пошук та виведення результатів

std::cout << "Результати бінарного пошуку для другого масиву:\n";

for (int i = 0; i < M; ++i) {

int result = binarySearch(array1, N, array2[i]);

if (result != -1) {

std::cout << "Елемент " << array2[i] << " є на позиції " << result + 1 << "\n";

} else {

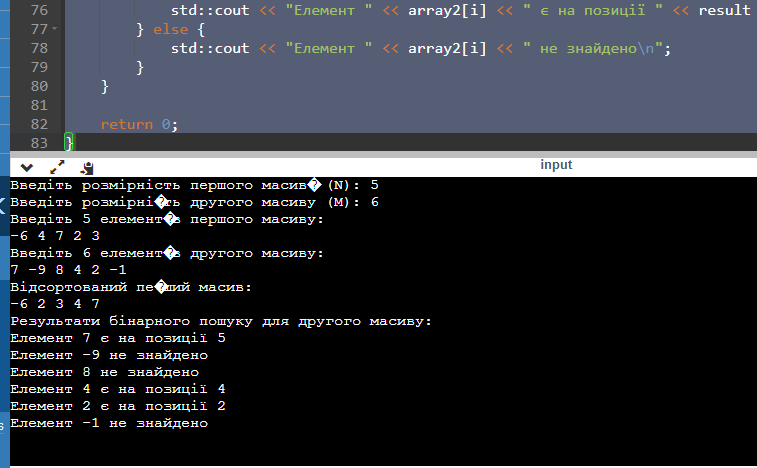
std::cout << "Елемент " << array2[i] << " не знайдено\n";

}

}

return 0;

}



**Мал. 1.** Створення масивів, сортування бальбашкою та бінарний пошук

**Висновок**

Виконуючи лабораторну роботу, я навчилася застосовувати в програмі бінарний пошук та сортування бульбашкою.