Лабораторна робота # 7.

**«АЛГОРИТМИ СОРТУВАННЯ. СОРТУВАННЯ Купою»**

Виконав: Шеліхов Дмитро   
Група: ІН-13

Варіант: 18

**Завдання ЛАБОРОТОРНОЇ роботи**

1. Реалізувати алгоритм сортування злиттям для вхідних даних різної кількості та степені упорядкованості елементів в масиві. Написати програму (функцію main), що містить опис даних та алгоритм сортування.

Виконати алгоритм для числа вхідних даних – n = 10, 100, 1000, 5000, 10000 елементів, згенерувавши три види різних послідовностей – випадкову, зростаючу, спадаючу.

2. Для кожного виду послідовності та кількості елементів визначити час сортування та кількість операцій присвоювань та порівнянь.

3. Вивести в консоль результати досліджень.

4. Звести результати до таблиці та зробити висновки.

**КОД**

#include <iostream>

#include <Windows.h>

#include <random>

#include <ctime>

using namespace std;

int N = 100000, maxN = 100000;

long long int checks{}, appropriations{};

int\* mas = new int[N];

clock\_t start\_time{}, end\_time{}, search\_time{};

void cinMas(int r = 0) {

srand(time(0));

if (r != 1 && r != 2) { cout << "Введіть кількість елементів масиву: "; cin >> N; }

else if (r != 2) { N = 10; }

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

mas[i] = rand() % N;

}

checks = 0; appropriations = 0;

}

void coutMas() {

cout << "Початок виведення масиву" << endl;

int r = 0;

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

cout << mas[i] << "\t";

r++;

if (r == 10 && i != N - 1) {

cout << endl;

r = 0;

}

}

cout << endl << "Кінець масиву" << endl << endl;

}

void coutRezult() {

cout << "Час виконання - " << search\_time << "мс;\t\tПерестановки - " << checks << "; \tПрисвоєння - " << appropriations << ";" << endl;

}

void heapify(int arr[], int n, int i, char side)

{

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (side == '>') {

if (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

}

else if (side == '<') {

if (l < n && arr[l] < arr[largest])

largest = l;

if (r < n && arr[r] < arr[largest])

largest = r;

}

checks += 2;

if (largest != i)

{

swap(arr[i], arr[largest]);

appropriations++;

heapify(arr, n, largest, side);

}

}

void heapSort(int arr[], int n, char side)

{

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i, side);

for (int i = n - 1; i >= 0; i--)

{

swap(arr[0], arr[i]);

appropriations++;

heapify(arr, i, 0, side);

}

}

void sort(char side) {

checks = 0; appropriations = 0;

start\_time = clock();

if (side == '+') {

heapSort(mas, N, '>');

}

else if (side == '-') {

heapSort(mas, N, '<');

}

else {

int buff\_N = N - 1;

while (buff\_N > 0)

{

int r = rand() % buff\_N;

buff\_N -= 1;

swap(mas[r], mas[buff\_N]);

checks++;

appropriations++;

}

}

end\_time = clock();

search\_time = (end\_time - start\_time);

}

void makeDefaulTest() {

for (int i = 1; i <= 4; i++) {

int N\_B = 10;

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

N\_B \*= 10;

}

N = N\_B;

int\* masBuff = new int[N\_B];

masBuff = mas;

cinMas(2);

sort('+');

cout << "N = " << N << ", Зростання - \t";

coutRezult();

mas = masBuff;

sort('-');

cout << "N = " << N << ", Спадання - \t";

coutRezult();

mas = masBuff;

sort('?');

cout << "N = " << N << ", Випадкове - \t";

coutRezult();

cout << endl;

}

}

void printArray(int arr[], int n)

{

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

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

cout << "\n";

}

int main() {

SetConsoleOutputCP(1251);

cinMas(1);

int make = 1;

while (make != 0) {

cout << "1. Змінити масив\n2. Вивести масив\n3. Сортувати масив\n4. Вивести результати\n5. Виконати предустановлений тест\n0. Вихід\n--> ";

cin >> make;

cout << endl;

if (make == 1) {

cinMas();

}

else if (make == 2) {

coutMas();

}

else if (make == 3) {

char side{};

cout << "Ведіть вид послідовності (+ - ?): ";

cin >> side;

sort(side);

}

else if (make == 4) {

coutRezult();

}

else if (make == 5) {

makeDefaulTest();

}

else {

if (make == 0) {

cout << "Вихід" << endl;

}

else {

cout << "Неправильний аргумент дії" << endl;

}

}

cout << endl;

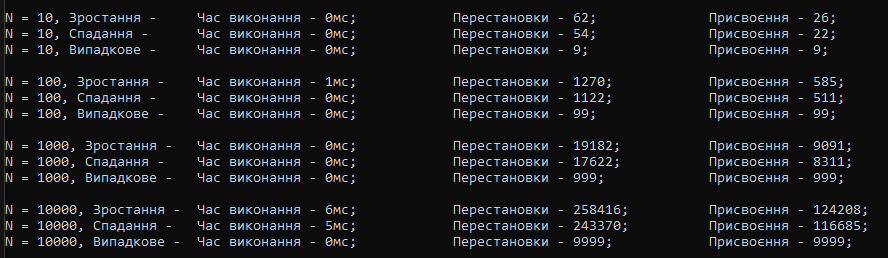
}

delete[]mas;

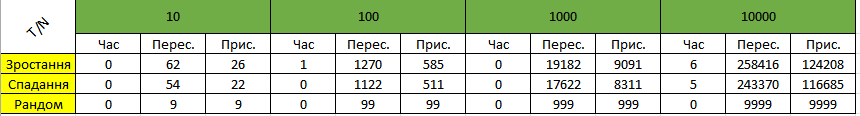
return 0;

}

**Приклад виконання**



**Таблица**

****

**Висновок:** Як ми бачимо цей алгоритм використовує набагато менше перевірок та присвоєнь, і по часу він набагато швидший чим попередні алгоритми. Працює він завжди с часом N\*logN, тому-що щоб знайти максимум у кучі нам портібно logN, а потім щоб перебрати усі елементи нам потрібно N таких операцій!!!!!