Звіт з предмету
Алгоритми та аналіз складності
Пустовіта Дмитра
КА-32

1. Умова Задачі (Варіант 16)

Відсортувати масиви з 100, 1000, 10000 елементів (випадково згенерованих) методами «Простого включення» та швидкого сортування. Проаналізувати отримані результати.

2. Математичне розв'язання задачі

Метод ***простого включення*** полягає в наступному:

Один елемент завжди можна вважати відсортованим масивом довжини 1. Розглядаючи послідовно решту елементів масиву будемо включати їх у відсортовану частину масиву, ставлячи на потрібне місце, тобто не порушуючи відсортованості цієї частини. Повторивши операцію включення n-1 раз, одержимо відсортований масив.

В середньому метод робить $C_{\text{сер}} = \frac{n(n+1)}{4} = O(n^2)$ порівнянь, хоча в найкращому та відповідно найгіршому випадку отримаємо: $C_{min} = n - 1$, $C_{max} = \frac{n(n-1)}{2}$, при цьому кількість копіювань буде такою: $M_{min} = 0$, $M_{max} = \frac{n(n-1)}{2}$, та відповідно в середньому $M_{\text{сер}} = \frac{n(n-1)}{4} = O(n^2)$.

Ідея методу ***швидкого сортування*** така:

Обирається деякий елемент, який назовемо медіаною. Всі елементи масиву на першому етапі розбирають на два підмасиви - елементів, більших від медіани та, відповідно, менших. На цьому, власне, перший етап закінчується. Далі процедура, що реалізує один етап, використовується рекурсивно, із застосуванням до обох частин масиву. Робота продовжується до тих пір, поки у кожному з підмасивчиків, що на них розбивається масив, не стане лише по одному елементу. Це означатиме, що масив відсортовано.

Оцінки кількості порівнянь та копіювань для цього методу відповідно наступні:

$C_{\text{сер}} \sim n log_2 n$, причому $C_{min} = \sum_{i=0}^{[log_2 n]+1}(n-i)$, а $C_{max} = \frac{n(n-1)}{2}$.

$M_{\text{сер}} \sim n log_2 n$, але як і в попередньому методі $M_{min} = 0$

Результати роботи

```
Непосортований масив
0.41; 4.67; 3.34; 5; 1.69; 7.24; 4.78; 3.58; 9.62; 4.64; 7.05; 1.45; 2.81; 8.27; 9.61;
4.91; 9.95; 9.42; 8.27; 4.36; 3.91; 6.04; 9.02; 1.53; 2.92; 3.82; 4.21; 7.16; 7.18; 8.95;
```

4.47; 7.26; 7.71; 5.38; 8.69; 9.12; 6.67; 2.99; 0.35; 8.94; 7.03; 8.11; 3.22; 3.33; 6.73; 6.64; 1.41; 7.11; 2.53; 8.68; 5.47; 6.44; 6.62; 7.57; 0.37; 8.59; 7.23; 7.41; 5.29; 7.78; 3.16; 0.35; 1.9; 8.42; 2.88; 1.06; 0.4; 9.42; 2.64; 6.48; 4.46; 8.05; 8.9; 7.29; 3.7; 3.5; 0.06; 1.01; 3.93; 5.48; 6.29; 6.23; 0.84; 9.54; 7.56; 8.4; 9.66; 3.76; 9.31; 3.08; 9.44; 4.39; 6.26; 3.23; 5.37; 5.38; 1.18; 0.82; 9.29; 5.41;

Посортований масив

0.06; 0.35; 0.35; 0.37; 0.4; 0.41; 0.82; 0.84; 1.01; 1.06; 1.18; 1.41; 1.45; 1.53; 1.69; 1.9; 2.53; 2.64; 2.81; 2.88; 2.92; 2.99; 3.08; 3.16; 3.22; 3.23; 3.33; 3.34; 3.5; 3.58; 3.7; 3.76; 3.82; 3.91; 3.93; 4.21; 4.36; 4.39; 4.46; 4.47; 4.64; 4.67; 4.78; 4.91; 5; 5.29; 5.37; 5.38; 5.38; 5.41; 5.47; 5.48; 6.04; 6.23; 6.26; 6.29; 6.44; 6.48; 6.62; 6.64; 6.67; 6.73; 7.03; 7.05; 7.11; 7.16; 7.18; 7.23; 7.24; 7.26; 7.29; 7.41; 7.56; 7.57; 7.71; 7.78; 8.05; 8.11; 8.27; 8.27; 8.4; 8.42; 8.59; 8.68; 8.69; 8.9; 8.94; 8.95; 9.02; 9.12; 9.29; 9.31; 9.42; 9.42; 9.44; 9.54; 9.61; 9.62; 9.66; 9.95;

N	Метод простого включення				Метод швидкого сортування			
	К-ть копіювань (М)		К-ть порівнянь (С)		К-ть копіювань (М)		К-ть порівнянь (С)	
	Теорет.	Експер.	Теорет.	Експер.	Теор.	Експер.	Теор.	Експер.
100	2475	2552	2525	2590	664	438	664	441
1000	249750	250150	250250	251342	9965	6963	9965	8865
10000	24997500	24974401	25002500	25040590	132877	94053	132877	101087

Кількість елементі в масиву	Метод простого включення		Метод швидкого сортування	
100	$\frac{M_{\text{експ}}}{\text{M}_{\text{теор}}} = \frac{2552}{2475} \approx 1.03$	$\frac{C_{\text{експ}}}{\text{C}_{\text{теор}}} = \frac{2590}{2525} \approx 1.02$	$\frac{\text{C}_{\text{експ}}}{\text{C}_{\text{теор}}} = \frac{438}{664} \approx 0.659$	$\frac{M_{\text{експ}}}{\text{M}_{\text{теор}}} = \frac{441}{664} \approx 0{,}664$
1000	$\frac{M_{\text{експ}}}{\text{M}_{\text{теор}}} = \frac{250150}{249750} \approx 1.001$	$\frac{C_{\text{експ}}}{\text{C}_{\text{теор}}} = \frac{251342}{250250} \approx 1.004$	$\frac{\text{C}_{\text{експ}}}{\text{C}_{\text{теор}}} = \frac{6963}{9966} \approx 0{,}698$	$\frac{M_{\text{експ}}}{\text{M}_{\text{теор}}} = \frac{8865}{9966} \approx 0{,}889$
10000	$\frac{M_{\text{експ}}}{\text{M}_{\text{теор}}} = \frac{24974401}{24997500} \approx 9.99$	$\frac{C_{\text{експ}}}{\text{C}_{\text{теор}}} = \frac{25040590}{25002500} \approx 1.001$	$\frac{\text{C}_{\text{експ}}}{\text{C}_{\text{теор}}} = \frac{94053}{132877} \approx 0.707$	$\frac{M_{\text{експ}}}{\text{M}_{\text{теор}}} = \frac{101087}{132878} \approx 0{,}76$

4.Код програми

```
// Sorts.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"
#include "iostream"
#include "fstream"

using namespace std;
```

```
void randomfill(double *a, int n){
	for (int i = 0; i < n; i++)
		*(a + i) = (rand() % 100000) / 100.0;
}

void print(int c, int m){
	ofstream fout("out.txt", ios_base::app);
```

```
        fout << "кількість порівняннь = " << c <<
endl;
        fout << "кількість копіюваннь = " << m <<
endl;
        fout.close();
}

void outvec(ofstream &fout, double* a, int n){
        //fout << "The vector is:\n";
        for (int i = 0; i < n; i++)
                //fout << "a[" << i+1 << "]= " << *(a
+ i) << "\n" << endl;
                fout << *(a + i)<< endl; //<< "; ";
}

void inpvec(ifstream &fin,  double* a, int n){
        for (int i = 0; i < n; i++)
                fin >> *(a + i);
}

void easyChoiseSort(double *a, int n){
        int i, j, k, c=0,m=0;
        double hold;
        for (i = 1; i < n; i++){
                hold = *(a + i);
                m++;
                j = 0;
                while (hold > *(a + j)){
                j++; c++;
                }
                if (j < i) {
                        for (k = i; k > j; k--){
                                m++;
                                *(a + k) = *(a + k -
1);
                        }
                        *(a + j) = hold; m++;
                }
        }
        print(c, m);
}

void quickSort(double *a, int n, int*c, int*m){
        int L = 0, R = n - 1, i = L, j = R;
        double tmp, med = a[(L + R) / 2];
        while (i <= j) {
                while (*(a + i) < med)  {
                        i++; *(c)+=1;
                }
                while (med < *(a + j))  {
                        j--; *(c)+=1;
                }
                if (i <= j) {
                        if (i<j) {
                                *(m) += 3;
                                tmp = *(a + i);
                                *(a + i) = *(a + j);
                                *(a + j) = tmp;
                        }
                        i++; j--;
                }
        }
        if (L<j)  quickSort(a+L,j-L+1,c,m);
        if (i<R)  quickSort(a + i, R - i + 1, c, m);
}

int _tmain(int argc, _TCHAR* argv[])
{
        ofstream fout1("початковий.txt"),
fout2("швидке.txt"), fout3("простого виключення.txt");

        int n, c = 0 , m = 0;
        cout << "Input number of elements:\n";
        cin >> n;
        double *a = new double[n];

        ofstream fout("out.txt");
        fout << "n= " << n <<endl;
        fout.close();
        randomfill(a,n);
        outvec(fout1, a, n);
        fout1.close();

        ifstream fin("початковий.txt");
        quickSort(a,n,&c,&m);
        print(c,m);
        outvec(fout2, a, n);
        inpvec(fin, a, n);
        easyChoiseSort(a,n);
        outvec(fout3, a, n);

        delete[] a;
        fin.close();
        fout2.close();  fout3.close();
        system("Pause");
        return 0;
}
```

3. **Висновок**

Метод швидкого сортування значно краще працює на великих масивах, хоча було б доречно модифікувати його, аби на підмасивах порівняно малої довжини сортування відбувалося одним з простіших випадків, в такому разі середню кількість порівнянь та копіювань можна буде дещо зменшити (не на порядок).