Лабораторна робота # 8.

**«АЛГОРИТМИ СОРТУВАННЯ. ШВИДКЕ СОРТУВАННЯ»**

Виконав: Шеліхов Дмитро   
Група: ІН-13

Варіант: 18

**Завдання ЛАБОРОТОРНОЇ роботи**

1. Реалізувати алгоритм швидкого сортування 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.

***Аналіз алгоритму***. Асимптотичні оцінки – n2;(худший), nlog2n(среднее). Но из-за того что он выполняет очень легкие операции он с такой ассимптотической оценкой выполняет сортировку быстрее. Алгоритм додаткової пам’яті не потребує, перегрупування чисел відбувається в межах заданого масиву, то ж об’єм оперативної пам’яті не перевищує деяку постійну величину. Коли ми беремо опорним элементом найбільший або найменший елемент при кожному розбиті.

**КОД**

#include <iostream>

#include <algorithm>

#include <ctime>

using namespace std;

int K = 0,K2 = 0;

int partition(int a[], int start, int end)

{

// Беремо крайній правий элемент як опорний

int pivot = a[end];

int pIndex = start;

for (int i = start; i < end; i++)

{

K++;

if (a[i] <= pivot)

{

swap(a[i], a[pIndex]);

K2+=3;

pIndex++;

}

}

swap (a[pIndex], a[end]);

K2++;

return pIndex;

}

void quicksort(int a[], int start, int end)

{

if (start >= end) {

return;

}

int pivot = partition(a, start, end);

quicksort(a, start, pivot - 1);

quicksort(a, pivot + 1, end);

}

void random(int a,int \*a2,int i2 = 0){

if(i2==0){//генерування рандомних єлементів у будь якому порядку

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

a2[i]=rand() % 100000;

}}

else if(i2==1){//генерування рандомних єлементів у зростаючому порядку

int k = 1;

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

a2[i] = i;

}}

else{//генерування рандомних єлементів у вибиваючуму порядку

int k = a;

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

a2[i] = k;

k=k-1;

}}

}

int main()

{

srand(time(NULL));

int e[]={10,100,1000,5000,10000};

for(int i2=0;i2<5;i2++){

cout<<"================="<<e[i2]<<" size "<<"=================";

for(int i=0;i<=2;i++){

if(i==0){

cout<<"\nrandom"<<endl;

int \*a = new int[e[i2]];

random(e[i2],a,i);

quicksort(a, 0, e[i2] - 1);

printf("\nProgram's time = %.3f\n",clock()/100000.0);

cout<<"Count of permutations = "<<K<<endl;

cout<<"Count of assignment = "<<K2;

K = 0;K2=0;

}

else if(i==1){

cout<<"\nhigh"<<endl;

int \*a = new int[e[i2]];

random(e[i2],a,i);

quicksort(a, 0, e[i2] - 1);

printf("\nProgram's time = %.3f\n",clock()/100000.0);

cout<<"Count of permutations = "<<K<<endl;

cout<<"Count of assignment = "<<K2;

K = 0;K2=0;

}else{

cout<<"\nlow"<<endl;

int \*a = new int[e[i2]];

random(e[i2],a,i);

quicksort(a, 0, e[i2] - 1);

printf("\nProgram's time = %.3f\n",clock()/100000.0);

cout<<"Count of permutations = "<<K<<endl;

cout<<"Count of assignment = "<<K2<<endl;

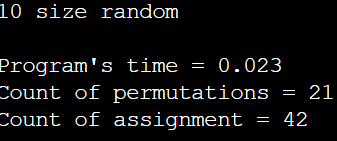
K = 0;K2=0;

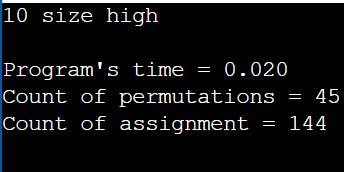
}}}

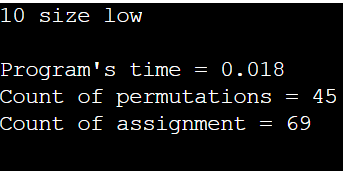
return 0;

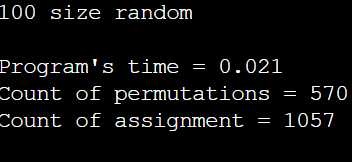
}

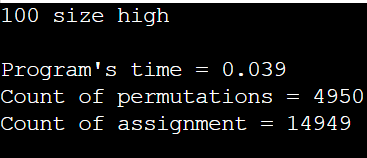
**Приклад виконання**

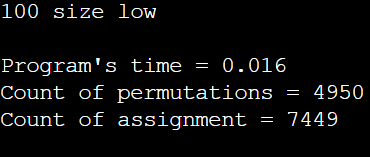


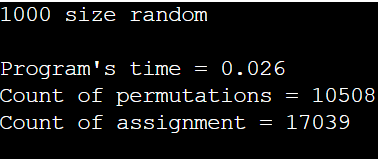


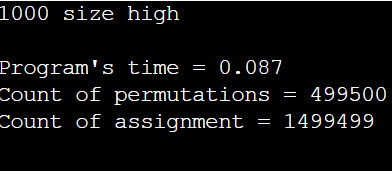


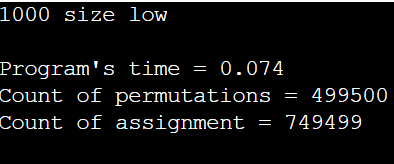


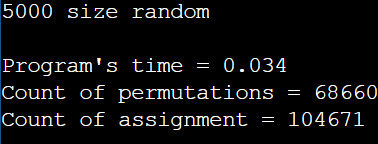


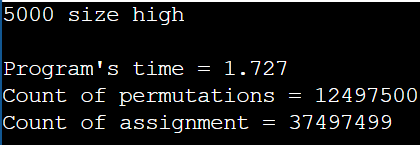


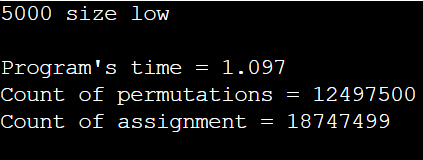


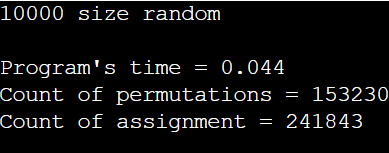


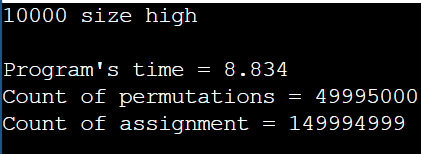


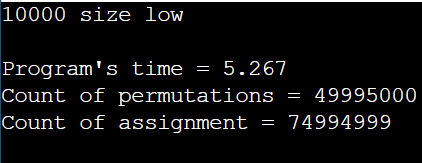












**Таблица**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Порядок та числа | 10 | 100 | 1000 | 5000 | 10000 |
| Випадковий | 0.023 мсек.  21 перевірок.  42 присваю. | 0.022 мсек.  570 перевірок.  1057 присваю. | 0.026 мсек.  10508 перевірок.  17039 присваю. | 0.034 мсек.  68660 перевірок.  104671 присваю. | 0.044 мсек.  153230 перевірок.  241843 присваю. |
| Зростаючий | 0.02 мсек.  45 перевірок.  144 присваю. | 0.039 мсек.  4950 перевірок.  14949 присваю. | 0.087 мсек.  499500 перевірок.  1499499 присваю. | 0.17 мсек.  12497500 перевірок.  37497499 присваю. | 0.8 мсек.  499950000 перевірок.  149994999  присваю. |
| Спадаючий | 0.018 мсек.  45 перевірок.  69 присваю. | 0.016 мсек.  4950 перевірок.  7449 присваю. | 0.074 мсек.  499500 перевірок.  749499 присваю. | 0.1 мсек.  12497500 перевірок.  18747499 присваю. | 0.52 мсек.  49995000 перевірок.  74994999 присваю. |

**Висновок:** Я можу сказати що це самий швидкий алгоритм з усіх алгоритмів у світі які базуються на зрівнянні чисел, у самому поганому випадку він працює за часом як усі інші алгоритми. Він працює за принципом “розділяй та влавствуй”. Це дуже швидкий аглоритм але доцільно його використовувати для массивів великої розмірності.