**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ**

**Самостійна робота №**3**​**

З дисципліни «Теорія алгоритмів»

Тема роботи: «Аналіз складності алгоритмів сортування» **Варіант №14**

Виконав студент

групи КН-22

Пашковський П.В.

Викладач:

Доманецька І. М.

**Київ – 2019**

**Методи для дослідження:**

1. Сортування включенням.
2. Сортування злиттям.

**Теоретична частина:**

**Сортування включенням:**

На кожному кроці алгоритму ми вибираємо один з елементів вхідних даних і вставляємо його на потрібну позицію у вже відсортованому списку доти, доки набір вхідних даних не буде вичерпано. Метод вибору чергового елементу з початкового масиву довільний; може використовуватися практично будь-який алгоритм вибору. Зазвичай (і з метою отримання стійкого алгоритму сортування), елементи вставляються за порядком їх появи у вхідному масиві.

Оцінка складності алгоритму:

Стійке сортування.

* Найкращий випадок: Оmega(n)
* Середній випадок: Teta(n^2)
* Найгірший випадок: O(n^2)

Переваги та недоліки:

Переваги: простота у реалізації; ефективний (зазвичай) на маленьких масивах; ефективний при сортуванні масивів, дані в яких вже непогано відсортовані; на практиці ефективніший за більшість інших квадратичних алгоритмів; є стабільним алгоритмом.

Недоліки: є неефективним на масивах великого розміру.

**Сортування злиттям:**

* основі цього способу сортування лежить злиття двох упорядкованих ділянок масиву в одну впорядковану ділянку іншого масиву. Злиття двох упорядкованих послідовностей можна порівняти з перебудовою двох колон солдатів, вишикуваних за зростом, в одну, де вони також розташовуються за зростом. Якщо цим процесом керує офіцер, то він порівнює зріст солдатів, перших у своїх колонах і вказує, якому з них треба ставати останнім у нову колону, а кому залишатися першим у своїй.

Так він вчиняє, поки одна з колон не вичерпається — тоді решта іншої колони додається до нової.

Під час сортування в дві допоміжні черги з основної поміщаються перші дві відсортовані підпослідовності, які потім зливаються в одну і результат записується в тимчасову чергу. Потім з основної черги беруться наступні дві відсортовані підпослідовності і так доти, доки основна черга не стане порожньою. Після цього послідовність з тимчасової черги переміщається в основну чергу. І знову продовжується сортування злиттям двох відсортованих підпослідовностей. Сортування триватиме доти, доки довжина відсортованої підпослідовності не стане рівною довжині самої послідовності.

Оцінка складності алгоритму:

Стійке сортування.

Потреба у додатковій пам’яті - Так.

* Найкращий випадок: O(nlogn)**​**
* Найгірший випадок: Ω(nlogn)**​**

**Практична частина:**

Таблиця 1, порівняння роботи двох алгоритмів сортування.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Сортування включенням** | | | **Сортування злиттям** | |  |
|  |  |  |  |  |  |  |
| **К-сть** | Найкр. | Середній | Найгірший | Найкр. | Середній | Найгірший(у |
| **елем.** | спосіб(у | спосіб(вип | (у | спосіб(у | спосіб(випа | зворотньому |
|  | потр. пор.) | адково) | зворотньо | потр. пор.) | дково) | пор.) |
|  |  |  | му пор.) |  |  |  |
|  |  |  |  |  |  |  |
| **10** | 1e-06 | 1e-06 | 2e-06 | 1e-06 | 1e-06 | 2e-06 |
|  |  |  |  |  |  |  |
| **50** | 2e-06 | 9e-06 | 3e-05 | 1e-06 | 2e-06 | 7e-06 |
|  |  |  |  |  |  |  |
| **100** | 3e-06 | 9e-06 | 7e-05 | 2e-06 | 7e-06 | 8e-05 |
|  |  |  |  |  |  |  |

**ДОДАТОК 1(сортування включенням):**

#include <iostream>

#include <algorithm>

#include <vector>

#include <random>

#include <ctime>

using namespace std;

int bubble(int\* arr, int n){

int tmp;

int i,j;

for(j = 0; j < n - 1; j++){

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

if (arr[i] > arr[i + 1]) {

tmp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = tmp;

}

}

}

}

void sorted(int\* arr,int n){

int counter=0;

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

for(int j=i; j>0 && arr[j-1]>arr[j];j--){

counter++;

int tmp=arr[j-1];

arr[j-1]=arr[j];

arr[j]=tmp;

}

}

//cout<<counter<<endl;

}

int main()

{

int n = 100;

int arr[n];

srand(time(0));

cout <<"arr: ";

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

arr[i] = rand()%50;

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

}

cout<<endl;

/\*bubble(arr,n);

/\*for(int i = 0; i < n; ++i)

{

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

}\*/

/\*for(int i = n - 1; i >= 0; i--){

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

}\*/

cout<<endl;

float time1 = clock();

sorted(arr, n);

float time2 = clock();

cout << "відсортований масив:";

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

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

cout<<endl;

cout<<"time1: "<<float(time2-time1)/CLOCKS\_PER\_SEC<<endl;

return 0;

}

**ДОДАТОК 2(сортування злиттям):**

#include <iostream>

using namespace std;

void merge(int \*,int, int , int );

void merge\_sort(int \*arr, int low, int high)

{

int mid;

if (low < high){

mid=(low+high)/2;

merge\_sort(arr,low,mid);

merge\_sort(arr,mid+1,high);

merge(arr,low,high,mid);

}

}

void merge(int \*arr, int low, int high, int mid)

{

int i, j, k, c[50];

i = low;

k = low;

j = mid + 1;

while (i <= mid && j <= high) {

if (arr[i] < arr[j]) {

c[k] = arr[i];

k++;

i++;

}

else {

c[k] = arr[j];

k++;

j++;

}

}

while (i <= mid) {

c[k] = arr[i];

k++;

i++;

}

while (j <= high) {

c[k] = arr[j];

k++;

j++;

}

for (i = low; i < k; i++) {

arr[i] = c[i];

}

}

int main()

{

int myarray[30], num;

cout<<"enter number of elements to be sorted:";

cin>>num;

cout<<"enter"<<num<<" elements to be sorted:";

for (int i = 0; i < num; i++) { cin>>myarray[i];

}

merge\_sort(myarray, 0, num-1);

cout<<"sorted:\n";

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

{

cout<<myarray[i]<<" ";

}

return 0;

}