

## СAA – Упражнение 9

### Бързо сортиране

Идеята на бързото сортиране е да се разменят елементи, които са максимално отдалечени един от друг – така се намалява броя на сравненията.

Принцип на действие:

- 1) Избира се "главен" или "водещ" елемент от масива.
- 2) Масивът се разделя на две части: елементи, по-малки от "главния" и елементи, по-големи от него.
- 3) Рекурсивно се повтарят горните стъпки, докато размерът на частта, която се сортира стане 1.

В най-лошия случай алгоритъмът има сложност  $O(n^2)$ . В средния случай алгоритъмът има сложност  $O(n \log n)$ , такава е и минималната сложност. Най-тежкият случай е когато на всеки етап водещият елемент е минималният за сортирания участък и следователно той се разделя на една част с размер 1, а останалите елементи са в другата част.

### Програмна реализация:

```
void quicksort(float A[], int start, int stop)
{
    //инициализация на необходимите променливи
    // определяне на медианата като среден по индекс елемент

    m=A[(start+stop)/2];
    do
    {
        // търсене на елемент, по-голям от медианата, обхождането на масива е от
        // началото към края му

        while (A[i]<m) i++;

        // търсене на елемент, по-малък от медианата, обхождането на масива е от
        // края към началото

        while (A[j]>m) j--;

        // ако двата показалеца не са се разминали се прави размяна на //елементите A[i] и A[j]

        if(i<=j)
        {
            размяна на елементите A[i] и A[j]
            i++; j--;
        }
    }while (i<=j);
}
```

```

// разделяне на масива на два подмасива – от началото до j-та позиция и от i-та
// позиция до края – за тези два подмасива се извиква рекурсивно функцията
// quicksort

if (start < j)
    quicksort(A, start, j);
if (i < stop)
    quicksort(A, i, stop);
}

```

### Цифрова сортировка

Цифровата сортировка се прилага за сортиране на цели числа в определен диапазон. Бива два вида: 1) по ключ; и 2) по ключ и номер.

#### Цифрова сортировка по ключ

Нека е даден е едномерен масив от цели числа в диапазона [0, 15].

A[i]	15	7	0	13	3	4	3	8	9
------	----	---	---	----	---	---	---	---	---

За сортирането му се използва помощен масив, чиито индекси са от минималния до максималния елемент в масив А – т.е. от 0 до 15. Индекси в помощния масив са елементите на масив А. Първоначално елементите на масив Р се нулират.

к	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Р [к]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Обхожда се масив А и за всеки негов елемент  $k=A[i]$  в масив Р елементът Р [к] се увеличава с 1. Така след обхождане на масив А всеки елемент Р [к] на помощния масив Р ще съдържа броя на елементите в А със стойност к. За примера Р ще има вида:

к	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Р [к]	1	0	0	2	1	0	0	1	1	1	0	0	0	1	0	1

Следва обхождане на масив Р и за всеки негов елемент  $P[k] > 0$  в масив А се записват Р [к] на брой елемента със стойност к.

След сортирането масив А добива следния вид:

A[i]	0	3	3	4	7	8	9	13	15
------	---	---	---	---	---	---	---	----	----

Сложността на описания алгоритъм е  $O(n+m)$ , където  $n$  е дължината на масива (брой елементи), а  $m$  е дължината на диапазона от цели числа. Когато диапазона на числата, които ще се сортират не е голям това е най-бързият алгоритъм за сортиране на цели числа (с линейна сложност).

### Реализация на алгоритъма

// нулиране на масив Р

```
for (i = min; i <= max; i++)  
    P[i] = 0;
```

// обхождане на масив А и попълване на масив Р с броя на срещнатите елементи.

```
for (j = 0; j < n; j++)  
    P[A[j]]++;
```

// обхождане на масив Р и попълване на стойностите за елементите на масив А вече // в сортиран вид

```
j = 0;  
for (i = min; i <= max; i++)  
{  
    for (k = 0; k < P[j]; k++)  
        A[j++] = i;  
}
```

### Задачи:

1. Реализирайте алгоритъма за бързото сортиране с отделна функция. В main функцията въведете елементите на масива, който ще бъде сортиран, извикайте функцията за бързо сортиране, след което разпечатайте елементите на сортирания вече масив. Масивът, който ще бъде сортиран да съдържа реални числа (тип float).
2. При така описаната реализация на бързото сортиране определете броя на рекурсивните обръщения във функцията quicksort в ситуацията, при която:  
А) водещият елемент на всяка стъпка е минималният за сортирания участък.  
Б) водещият елемент на всяка стъпка е максималният за сортирания участък.  
Обосновете отговорите си.
3. Реализирайте алгоритъма за цифрова сортировка с отделна функция. В main функцията въведете елементите на масива, който ще бъде сортиран, извикайте функцията за цифрова сортировка, след което разпечатайте елементите на сортирания вече масив. Използвайте предварително зададени стойности за минимален и максимален елемент.