

САА – Упражнение 6

Методи за съставяне на алгоритми. Метод „Разделяй и владей“. Рекурсия

Метод “Разделяй и владей”. Сортиране чрез сливане

Прилагането на подход 'разделяй и владей' при съставянето на алгоритми протича в три стъпки:

Стъпка 1: **Разделяй** – Задачата, която трябва да бъде решена се разделя на няколко подобни подзадачи (най-често две) с по-малък размер.

Стъпка 2: Рекурсивно се решава всяка от подзадачите поотделно. Ако размерността на подзадачата е достатъчно малка и подзадачата е вече достатъчно опростена, се намира директно нейното решение.

Стъпка 3: **Владей** – Комбинират се решенията на подзадачите в едно общо решение на изходната задача.

Пример за приложение на метода 'разделяй и владей'

Нека да разгледаме алгоритъм за сортиране на масив чрез сливане. Той се състои от следните три стъпки:

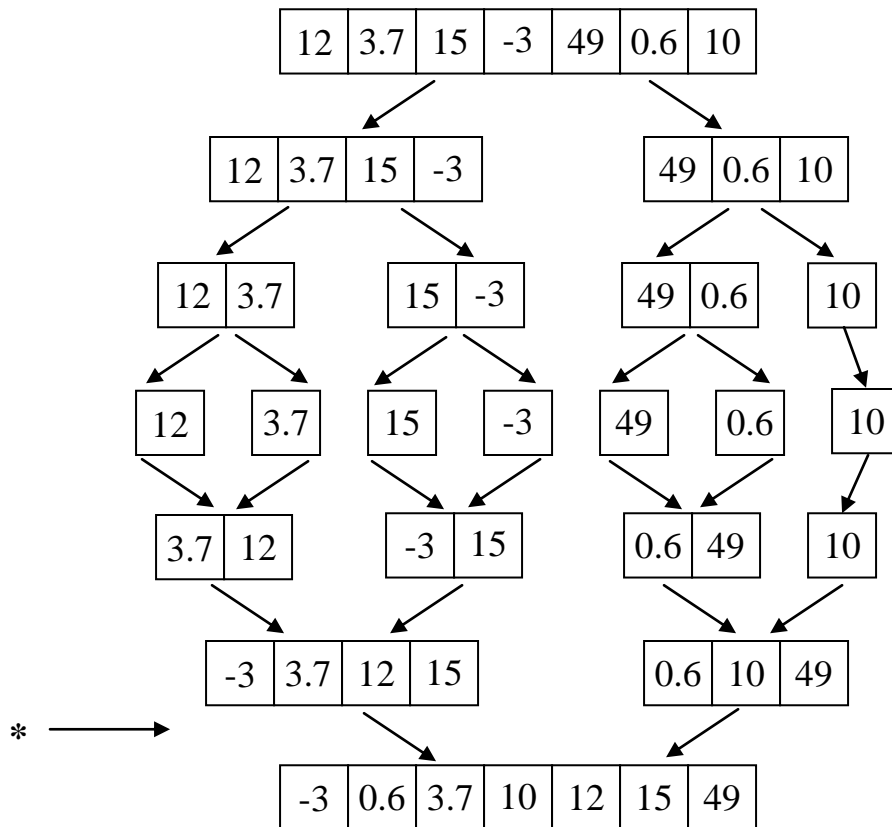
- 1) Разделяме масива, който ще сортираме, на два подмасива с равен (или почти равен) брой елементи.
- 2) Сортираме двата подмасива рекурсивно.
- 3) Сливаме двата сортирани подмасива в общ масив, също сортиран.

Пример за сортиране на масив чрез сливане (от върха към дъното) е показан на Фигура 1.1

Програмната реализация на разглеждания алгоритъм за сортиране е представена по-надолу.

- 1) рекурсивна функция за сортиране чрез сливане

```
void merge_sort (int l, int r, float A[])
{
    int m;
    if (l < r)
    {
        m = (l + r)/2;
        merge_sort (l, m, A);
        merge_sort (m+1, r, A);
        merge (l, m, r, A);
    }
}
```



Фигура 1.1. Пример за сортиране чрез сливане

2) фрагменти от функцията, реализираща сливане на два сортирани масива в един общ сортиран масив. Нека, $i1$ и $i2$ са съответно първи и последен индекс на първия масив, а $j1$ и $j2$ са първи и последен индекс на втория масив, n е сумата от броя на елементите на двата масива, а T е помощен масив, за начален индекс на който е зададено $k = i1$.

```

while ((i1<=i2)&&(j1<=j2))
{
    if (A[i1]<A[j1])
    {
        T[k]=A[i1]; i1++;
    }
    else
    {
        T[k]=A[j1]; j1++;
    }
    k++;
}
while (i1<=i2)

```

```

{
    T[k]=A[i1]; i1++; k++;
}
while (j1<=j2)
{
    T[k]=A[j1]; j1++; k++;
}
for (k=0; k<n; k++)
{
    j1--; A[j1]=T[j1];
}

```

Сложността на описания алгоритъм за сортиране чрез сливане е $O(n \log_2 n)$, където n е броят на елементите, които ще бъдат сортирани.

Рекурсия

Рекурсивен алгоритъм е този, който при изпълнението си се обръща към себе си пряко или косвено поне един път. При създаване на рекурсивни функции следва да обърнем внимание на няколко важни условия:

- 1) Възможност за декомпозиране на общия случай на по-прости: задачата, която се решава, трябва да може да се разбие на подзадачи с по-малка размерност, за решаването на които да може да се приложи рекурсивно същия алгоритъм;
- 2) Дефиниране на базови случаи: следва да се определят един или няколко (краен брой) базови случаи, които съответстват на най-ниската размерност на задачата и се решават директно. Прието е тези базови случаи да се наричат **дъно на рекурсията**;
- 3) Определяне на параметрите на задачата: стойността на параметъра (един или няколко), определящ размерността на задачата трябва да намалява при всяко следващо рекурсивно извикване. Така достигането до дъното на рекурсията следва да бъде гарантирано след краен брой стъпки.

При изпълнението на рекурсивен алгоритъм се поддържа програмен стек, в който се пазят стойностите на всички локални променливи и фактически параметри.

Пример за съставяне на рекурсивен алгоритъм - числа на Фибоначи

Числата на Фибоначи съставят следната числова редица:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ...

Всеки член от тази редица се получава като сума от предходните два.

Рекурсивна функция за определяне на n-тото число на Фибоначи:

```

int fibonacci (int n)
{
    if (n <= 1) return 1;
    return fibonacci (n - 1) + fibonacci (n - 2);
}

```

Итеративно решение за намиране на n -тото число на Фибоначи:

```
int fibonacci_1 (int n)
{
    int i, f1 = 1, f2 = 1, f;
    for (i = 2; i <= n; i++)
    {
        f = f1 + f2; f2 = f1; f1 = f;
    }
    return f;
}
```

Задачи:

1. Напишете програма, която реализира алгоритъма за сортиране на масив чрез сливане.
2. За какъв тип числа може да бъде приложен алгоритъма за сортиране чрез сливане?
3. Сравнете времето за изпълнение на рекурсивната и итеративната функция за пресмятане на числата на Фибоначи. По какъв начин входните данни влияят върху времето за изпълнение на всяка от двете функции?
4. * Реализирайте алгоритъма за сортиране чрез сливане на списъци.