

Merge Sort

Лесников Юрий, seagest

1 Сортировка слиянием (Merge Sort)

Замечание 1.1. Если у нас есть два отсортированных массива, то можно их слить в третий, который будет тоже отсортирован за $O(n + m)$, где n и m – размеры массивов. Это достигается методом двух указателей.

Сортировка слиянием

1. Разбиваем массив на две примерно равные части.
2. Решаем все подзадачи, применяя рекурсивно шаги 1 и 3, пока не дойдем до базы, в которой все тривиально (разбиваем массивы и дальше пополам, пока не дойдем до массива из одного элемента).
3. Объединяем решения подзадач (сливаем два отсортированных массива в один большой).

Algorithm 1 Merge Sort

```
1: function MergeSort( $A$ )
2: if  $A.len = 1$  then
3:   return  $A$ 
4: end if
5:  $A_{left} \leftarrow \text{MergeSort}(A[0 \dots \lfloor \frac{n}{2} \rfloor - 1])$ 
6:  $A_{right} \leftarrow \text{MergeSort}(A[\lfloor \frac{n}{2} \rfloor \dots n - 1])$ 
7:  $i_{left} \leftarrow 0$ 
8:  $i_{right} \leftarrow 0$ 
9:  $i \leftarrow 0$ 
10:  $A_{res} \leftarrow []$ 
11: while  $i_{left} < A_{left}.len$  and  $i_{right} < A_{right}.len$  do
12:   if  $A_{left}[i_{left}] \leq A_{right}[i_{right}]$  then
13:      $A_{res}[i] \leftarrow A_{left}[i_{left}]$ 
14:      $i_{left} \leftarrow i_{left} + 1$ 
15:   else
16:      $A_{res}[i] \leftarrow A_{right}[i_{right}]$ 
17:      $i_{right} \leftarrow i_{right} + 1$ 
18:   end if
19:    $i \leftarrow i + 1$ 
20: end while
21: while  $i_{left} < A_{left}.len$  do
22:    $A_{res}[i] \leftarrow A_{left}[i_{left}]$ 
23:    $i_{left} \leftarrow i_{left} + 1$ 
24:    $i \leftarrow i + 1$ 
25: end while
26: while  $i_{right} < A_{right}.len$  do
27:    $A_{res}[i] \leftarrow A_{right}[i_{right}]$ 
28:    $i_{right} \leftarrow i_{right} + 1$ 
29:    $i \leftarrow i + 1$ 
30: end while
31: return  $A_{res}$ 
```

1.1 Анализ сортировки слиянием.

- Сортировку слиянием можно описать рекуррентным соотношением: $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$. Тогда из мастер теоремы о рекурсии получаем, что $T(n) = \Theta(n \log n)$. Значит, время работы алгоритма составит $\Theta(n \log n)$.
- Потребляемая память составит $\mathcal{O}(n)$, поскольку необходимо завести массив для слияний. Также алгоритму нужно $\mathcal{O}(\log n)$ стековой памяти для хранения параметров рекурсивных вызовов.
- К счастью, алгоритм можно реализовать итеративно и без рекурсии, начав решать задачу снизу-вверх: сначала сливаем соседние массивы размеров 1, затем размеров 2, затем 4, и так далее.

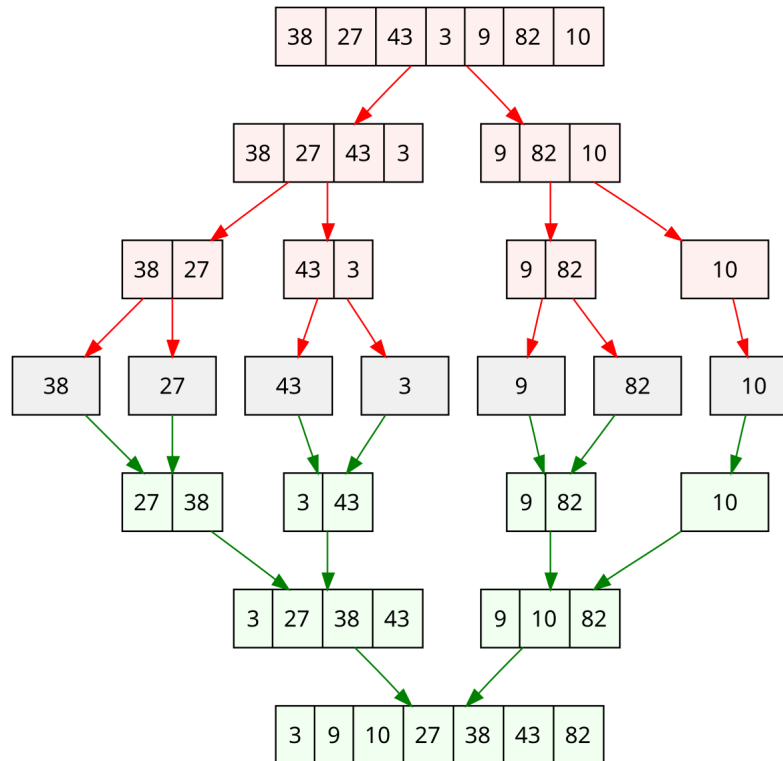


Рис. 1: Иллюстрация работы Merge Sort