**Merge sort**

“Fools ignore complexity.

Pragmatists suffer it.

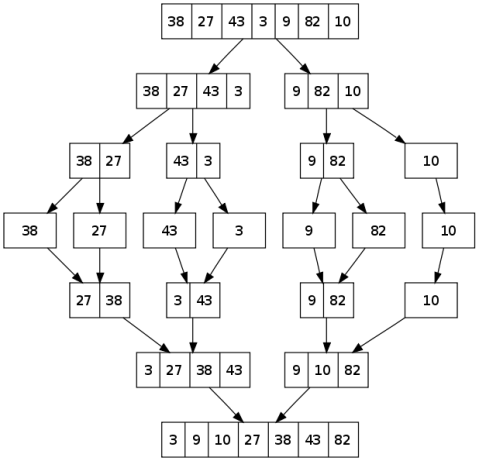
Some can avoid it.

Geniuses remove it.”

~ Epigrams in Programming

Merge sort е част от алгоритми, наречени Divide and conquer. При този тип алгоритми основната идея е разделяне на сложната задача на няколко по-прости, които се решават по-лесно и начина за тяхното решение може да се приложи при по-сложната задача. Това решение се основава на рекурсията и, както ще видите при този алгоритъм, ще ползваме точно рекурсия. Ако някой се интересува от тези алгоритми, може да погледне в библията на Наков там има цял дял, посветен на тях. Важно е да се отбележи, че не всяка задача може да бъде решена по метода Divide and conquer, както не всяка задача може да бъде решена с динамично или графи.

Основната функция на този алгоритъм е да слее два сортирани масива в един сортиран масив. Ако например имаме а[] = {3,4} и b[] = {5,6}, когато слеем двата масива в c[] = {3,4,5,6}, получаваме един по-голям сортиран масив. Как всъщност се извършва това сливане? Взимаме първите елементи от двата масива и ги сравняваме, по-малкият от тях го слагаме в третия масив и местим индекса на масива, от който сме взели елемента, с една позиция напред. Правим това, докато единия или двата масива не се изпразнят. Ако след сливането са останали елементи в някой от двата първоначални масива, просто ги прехвърляме в края на изходния масив, защото знаем, тези елементи са със сигурност по-големи от другите елементи в масива. Като е важно да се отбележи, че ако двата масива НЕ са сортирани предварително, този алгоритъм няма да работи. Какво правим в случаите, когато имаме 1 масив, който не е сортиран? Например масива a[] = {38,27,43,3,9,82,10}



Първо трябва да разделим масива на 2 по-малки масива, след това новите масива трябва да бъдат разделени на 2. Това се повтаря, докато накрая не получим масиви от по 1 елемент. Защо точно масиви от по 1 елемент ли? Защото така винаги е вярно твърдението, че масива от 1 елемент е сортиран. След това прилагаме обратната процедура, с която сливаме масивите от по 1 елемент в по-големи масиви. Сливаме ги по принципа, обяснен по-горе. За разлика от преди, този път вече елементите на новите по-големи масиви ще са сортирани. Извършваме това сливане, докато не получим отново един масив, в който вече стойностите са сортирани. Предимствата на този алгоритъм са, че сложността му винаги е O(nlog(n)), независимо от това какъв масив подаваме, и освен това се справя със същата скорост, дори и когато имаме масив с числа, които са еднакви. Това го казвам, защото heep sort-a, който отново е част от D&C алгоритмите, трудно се справя с масив с еднакви числа и често може сложността да се изроди в линейна в такива случаи. Това няма как да се случи при merge sort-a. Освен това merge sort-a е лесен за писане и доста пъти скоростта му не е много по–ниска от тази на quick sort. Един от недостатъците на това сортиране е, че иска допълнителна памет по време на изпълнение на програмата за допълнителни масиви, а ако имплементацията е реализирана чрез указатели - и за тях. Това сортиране е стабилно. Има се предвид, че запазва относителната подредба на ключовете на еднаквите елементи, което не се среща често при другите сортирания. Подобно на повечето съвременни методи за сортиране, сортирането чрез сливане се държи лошо при малък брой елементи. Както и при бързото сортиране, добри резултати се получават при комбинирането му с други методи. Стандартен подход е, когато броят на елементите на разглежданото множество падне под 15-20 (този брой е примерен), да се използва сортиране чрез вмъкване.