1. Занятие

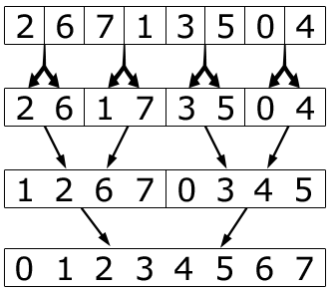
Алгоритм сортировки слиянием основан на идее, что два отсортированных списка можно слить в один отсортированный список за время, равное суммарной длине этих списков.

Напишем функцию merge.
Создадим результирующий список. Далее сравним первые элементы данных списков. Тот элемент, который меньше, скопируем в конец результирующего списка (который первоначально пуст) и в этом списке перейдем к следующему элементу. Будем повторять этот процесс (выбираем из начала двух списков наименьший элемент, копируем его в результирующий список), пока один из исходных списков не кончится. После этого оставшиеся элементы (один из двух исходных списков будет непуст) скопируем в результирующий список.
Для того, чтобы не удалять начальные элементы из списков, заведем два индекса i и j, указывающие на текущие элементы в каждом списке. Вместо удаления элементов будем передвигать эти индексы. В конце, добавим к результирующему списку оставшиеся элементы из двух исходных списков A и B (один из этих массивов уже полностью обработан, это не должно нас смущать).

Заметим, что сложность работы функции merge — линейная от суммарных длин списков A и B, так как каждый элемент обрабатывается ровно один раз за $O(n)$.

Теперь можно реализовать сортировку слиянием. Напишем функцию merge_sort.

Разберем алгоритм сортировки слиянием на следующем примере. Имеется неупорядоченная последовательность чисел: 2, 6, 7, 1, 3, 5, 0, 4. После разбивки данной последовательности на единичные массивы, процесс сортирующего слияния (по возрастанию) будет выглядеть так:



Массив был разделен на единичные массивы, которые алгоритм сливает попарно до тех пор, пока не получится один массив, все элементы которого стоят на своих позициях.

Это — рекурсивная функция, которая получает на вход исходный список и список, составленный из тех же элементов, но отсортированный. Если длина исходного списка равна 1 или 0, то он уже отсортирован и сортировать его не надо. Если же длина списка больше 1, то разобьем его на две части равной (или почти равной, если длина исходного списка — нечетная). Отсортируем обе эти части, затем сольем их вместе при помощи функции merge.

Оценим сложность этого алгоритма. Пусть массив содержит n элементов. Тогда за O(n) его можно разделить на две части и после сортировки слить их вместе. Каждая из этих двух частей имеет размер n/2, и за O(n) шагов каждую из них можно поделить на две части размером n/4 и затем после сортировки слить их вместе. Аналогично, четыре части размером n/4 за суммарное O(n) шагов делятся на части размером n/8 и сливаются вместе. Этот процесс «в глубину» продолжается столько раз, сколько раз можно число n делить на 2, до тех пор, пока размер части не станет равен 1, то есть log2n. Итого, общая сложность этого алгоритма равна O(nlog2n).

Одним из недостатков сортировки слиянием является тот факт, что он требует много вспомогательной памяти (столько же, каков размер исходного массива) для реализации.