INSERTION SORT AND MERGE SORT

November 2016

1 Introduction

The lecture slides explains the working of two majorly used sorting algorithms namely Merge sort and Selection sort.we"ll use illustrations to explain the algorithm.

Uses of sorting: Mp3 player sorting, Data compression,computer graphics. We"ll start with Insertion sort and then finally move to Merge sort.

2 Insertion sort

2.1 Algorithm

For i=1,2,3..n Insert A[i] into sorted array A[0...i-1] by pair wise swaps down to the correct position.

2.2 Illustration

Let's sort the integers 14,33,27,10,35,19,42,44 using insertion sort algorithm.

- take an unsorted array : 14,33,27,10,35,19,42,44.
- compare first two elements , we found out that 14,33 are in correct positions, so from now on 14 is in sored sub-list.
- Move head and compare 33,27 and swaps them to make the array as : 14,27,33,10,35,19,42,44
- Check sorted sub-list which has 14 and 27 and the swapping if required which in this case isn't.
- compare 33 and 10 and swap them : 14,27,10,33,35,19,42,44.Swapping made sorted sub-list unsorted , so swap them to get : 14,10,27,33,35,19,42,44
- Swap 14 and 10 to get : 10,14,27,33,35,19,42,44

This process goes on till all the unsorted value are covered in a sorted sub-list.

2.3 Time complexity

For each step of key movement takes O(n) time and there are n key movements. Thus total time complexity $=O(n^2)$

3 Merge sort

3.1 Algorithm

What this algorithm does is dividing the input into two halves and then merges the two sorted halves using a merging algorithm.

3.2 Illustration

The following illustration shows a complete merge sort process for array : 38,27,43,3,9,82,10. Closer inspector reveals that array splits into two parts until it's size becomes 1 and then merge routine is applied.

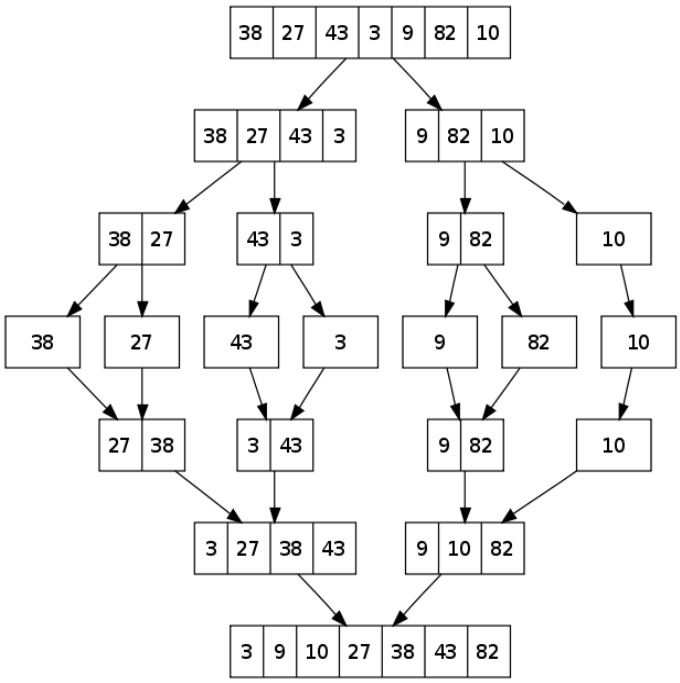

Figure 1: Implementation of Merge sort Algorithm

3.3 Time complexity analysis

It's a recursive algorithm and Time complexity can be expressed as $T(n) = 2T(n/2) + O(n)$

Solving this recurrence shows that solution for this recurrence is $O(nlog(n))$. Auxiliary space used : $O(n)$

Lecture URL: `https://youtu.be/Kg4bqzAqRBM`