Notes on Chapter 10 - Some Simple Algorithms and Data Structures

Swarup Tripathy [*]

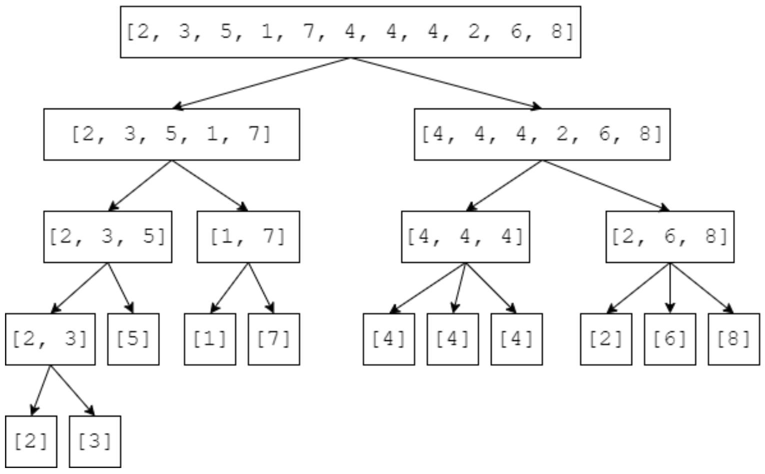
May 2022

A curated list of important points for my reference.

1. Introduction to Algorithms, by Cormen, Leiserson, Rivest, and Stein, is an excellent source for thous of you not intimidated by a fair amount of mathematics.
2. The key to efficiency is a good algorithm, not clever coding tricks.
3. **SEARCH ALGORITHMS**
   (a) a method for finding an item or group of items with specific properties within a collection of items.
4. **SORTING ALGORITHMS**
   - The standard implementation of sorting in most Python implementations runs in roughly O(n*log(n)) time, where n is the length of the list.
   - In most cases, the right thing to do is to use either Python's built in sort method *L.sort()* or its built in function sorted *sorted(L)*
   - *SELECTION SORT*
     - given list [4,2,6,1]
     - a 'while loop' over individual elements
     - a 'for loop' under 'while loop' traversing over individual elements
     - if the 'for loop' element is less than the 'while loop' element, swap it
     - Here the complexity of the algorithm will be quadratic in the length of L.
   - *MERGE SORT*
     - Also known as Divide and Conquer Algorithm.
     - Breaks down problem into multiple subproblems recursively until they become simple to solve.

[*]John V Guttag

- Solutions are combined to solve original problem
- O(n*log(n)) is the running time, optimal running time for comparison based algorithms.
- General Principle
  - Split array in half
  - Call mergeSort on each half to sort them recursively
  - Merge both sorted halves into one sorted array.
  - We continue this until we get arrays of size 1, since arrays of size 1 are always sorted.



  - At the bottom nodes, you can observe arrays of size 1

- The sorting algorithm used in most Python Implementation is called **timsort.**
- Timsort's worst case performance is the same as merge sort's, but on average it performs considerablty better.