# Tutorial 1

Solve the following recurrences using the master method:

- $T(n) = 7T(n/2) + \Theta(n^2)$
  (Strassen's algorithm for matrix multiplication, read Section 4.2 if you are interested)
  $n^2/n^{\log_2 7} = O(n^{-\epsilon}) \Rightarrow$ Case 1: $T(n) = \Theta(n^{\log_2 7}) \approx \Theta(n^{2.81})$.
- $T(n) = 143640T(n/70) + \Theta(n^2)$ (Pan's algorithm for matrix multiplication)
  $n^2/n^{\log_{70} 143640} = O(n^{-\epsilon}) \Rightarrow$ Case 1: $T(n) = \Theta(n^{\log_{70} 143640}) \approx \Theta(n^{2.79})$.
- $T(n) = 2T(n/2) + \Theta(n)$ (Merge-sort): $n/n = 1 = \log^0 n \Rightarrow$ Case 2: $T(n) = \Theta(n \log n)$
- $T(n) = 4T(n/2) + \Theta(n^3)$
  $n^3/n^{\log 4} = n = \Omega(n^\epsilon)$ and $4(n/2)^3 = n^3/2 < n^3 \Rightarrow$ Case 3: $T(n) = \Theta(n^3)$

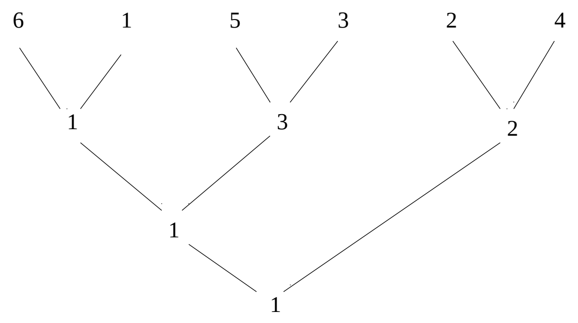
**Stooge sort**: recursively sort first two-thirds, last two-thirds, and first two-thirds again.

$$T(n) = \begin{cases} 1 & \text{if } n \le 2 \\ 3\,T(2n/3) + 1 & \text{if } n > 2 \end{cases}$$

Master method: $f(n)/n^{\log_b a} = O(1/n^{\log_{3/2} 3}) = O(n^{-\epsilon}) \Rightarrow$ Case 1: $T(n) = \Theta(n^{\log_{3/2} 3}) \approx \Theta(n^{2.7})$.

**Problem 9.1-1** (215): Find the second smallest of $n$ elements in $n + \lceil \log n \rceil - 2$ comparisons.

**Solution:** The smallest element is found in $n - 1$ comparisons through a cup tournament where the least of two compared elements advance to the next round. After $\lceil \log n \rceil$ rounds the smallest element remains. The second smallest is any of the $\lceil \log n \rceil$ elements that has been compared to the smallest. By $\lceil \log n \rceil - 1$ further comparisons we get the answer. Example:



With 6 elements we get 3 candidates to be the second smallest. The total number of comparisons is $(6-1) + (3-1) = 7$.

**Problem 9-1** (224): Give the $i$ largest numbers (distinct), in sorted order.

**Solution:** (a) Sort and output the $i$ largest, which takes $O(n \log n) + O(i) = O(n \log n)$ time.

(b) Build a max-heap, for instance, and do $i$ EXTRACT-MAX, which require $O(n) + O(i \, \log n)$ time.

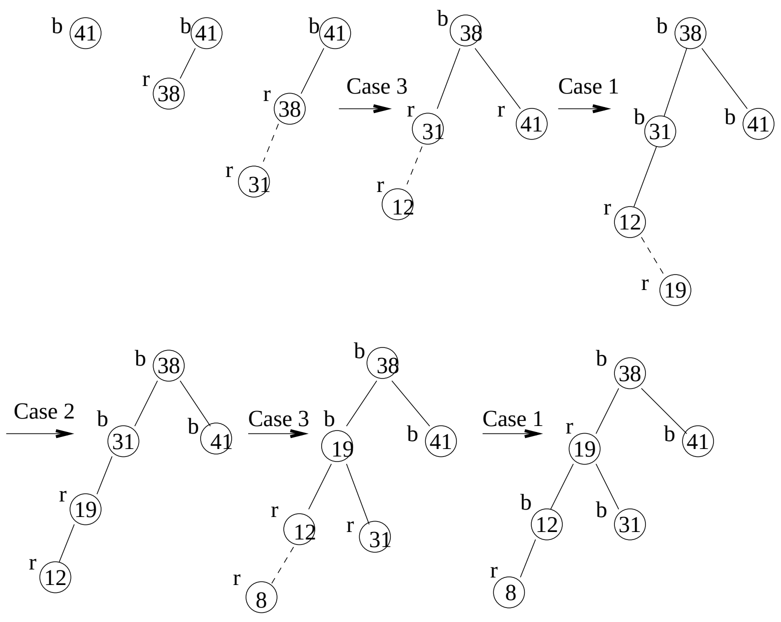
(c) Compute the $(n-i)$th smallest number using SELECT in $O(n)$ time. SELECT partitions the numbers in two sets, those smaller than the $(n-i)$th and those larger. Sort the $i$ larger in $O(i \, \log i)$ time.

**Problem 13.1-5** (312): Show that the longest path to a leaf $\leq 2\cdot$ shortest path in a red-black tree.

**Solution:** In the longest path at least every other node is black, in the shortest path every node can be black. Since two paths to leaf contains an equal number of black nodes this means that the length of a longest path is at most twice the length of a shortest.

**Problem 13.3-2** (322): Insert 41, 38, 31, 12, 19, 8 into an empty red-black tree.

**Solution:**



## Discussion of assignment 1

1. Remember to motivate the $O(n)$ time complexity in (a) and (c).
2. The non-trivial operation is `Balance`.

Strive for clear and concise solutions on all of the assignments. Note that solutions shall be worked out individually. You may be asked by the teacher to explain your submitted solution.

- Justify that your solution gives the correct answer to the problem.
- State and motivate the time complexity of the algorithms to justify their efficiency, which is with regards to **worst case time** unless otherwise stated.

## Digressions

Feynman's problem solving method: (1) Write down the problem. (2) Think very hard.[1] (3) Write down the answer.

H.L. Mencken: For every complex problem there is a solution that is simple, elegant, and wrong.

Hoare's law for big problems: Inside every big problem there is a small problem struggling to get out.

[1]Feynman talked of getting the *inside track* on solving a problem. Viewing it from a new angle could provide a solution.