8.1 Master Theorem 8

$$T(n) = \begin{cases} \Theta(1), & n \leq 1, \\ aT(\frac{n}{b}) + f(n), & otherwise(a \geq 1 \; and \; b > 1). \end{cases}$$

case 1 : If f(n) = O($n^{log_b a - \epsilon}$) for some constant $\epsilon > 0$, then *T(n)* = $\Theta(n^{log_b a})$
case 2 : If f(n) = $\Theta(n^{log_b a} lg^k n)$, then *T(n)* = $\Theta(n^{log_b a} lg^{k+1} n)$
case 3 : If f(n) = $\Omega(n^{log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $a f\left(\frac{n}{b}\right) \leq cf(n)$ for some constant *(*c) < 1 and all sufficiently large *(*n), then *T(n)* = Θ*(f(n)))*

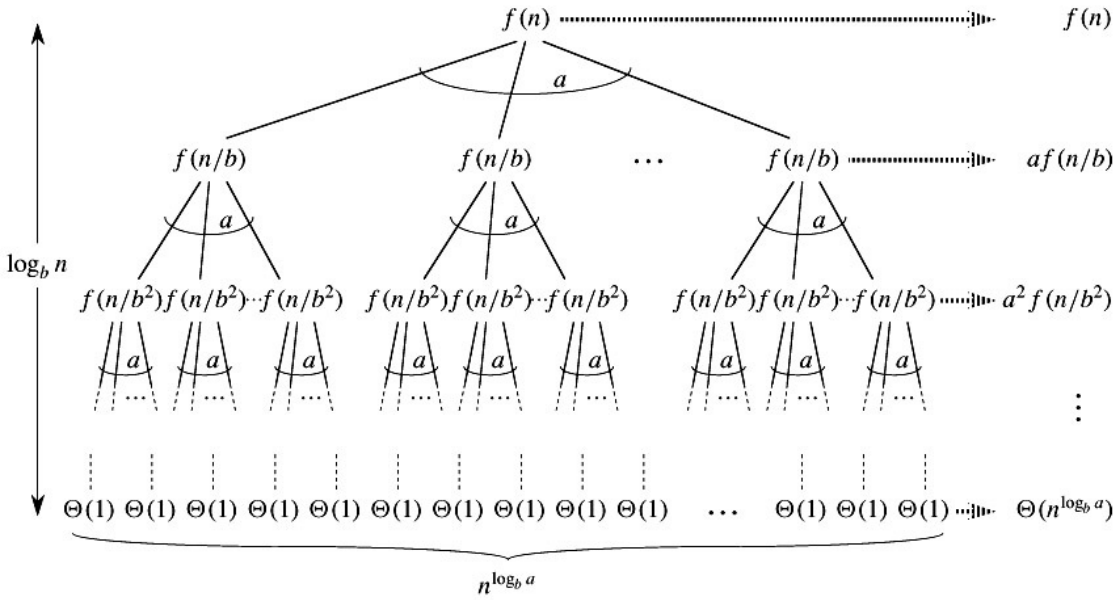

Figure 8.1.1: *Recurrence Tree*

8.1.1 Recurrences not solvable using the Master theorem

Example 1: T(n) = $\sqrt{n}$ T $\left(\frac{n}{2}\right)$ + n According to the Master's Theorem, here a = $\sqrt{n}$ is not a constant

Example 2: T(n) = 2T$\left(\frac{n}{logn}\right)$ + n^2 In this example, $b = logn$ is not a constant

Example 3: T(n) = $\frac{1}{2}$T$\left(\frac{n}{2}\right)$ + n^2 In this example, a(number of subproblems should be atleast 1) = $\frac{1}{2}$ is not ≥ 1

Example 4: T(n) = 2T$\left(\frac{4n}{3}\right)$ + n Here, b(subproblem size) = $\frac{3}{4}$ is not > 1

Example 5: T(n) = 3T$\left(\frac{n}{2}\right)$ - n Here, f(n) = -n is not positive.

Example 6: T(n) = 2T$\left(\frac{n}{2}\right)$ + $n^2 \sin n$ Here, f(n) = $n^2 \sin n$ and $\sin n$ range varies from -1 to 1 for any given n. Hence violates the regularity condition of case 3.

Example 7: T(n) = 2T$\left(\frac{n}{2}\right)$ + $\left(\frac{n}{logn}\right)$ In the above example,
No. of leaves, $n^{log_b a}$ = $n^{log_2 2}$ = n.
With case 1, f(n) = O($n^{log_b a}$) = $\left(\frac{n}{logn}\right)$, but $\neq$ O($n^{log_b a - \epsilon}$) for any constant $\epsilon > 0$.
With case 2, f(n) = $\Theta(n^{log_b a} lg^k n)$ Here, f(n) = $nlog^-1n$ where k = -1 is not ≥ 0 to solve.
With case 3, f(n) = $\Omega(n^{log_b a + \epsilon})$ for any constant $\epsilon > 0$, which infers that if f(n) is polynomially lesser than the number of leaves we can solve it but the given f(n) value doesn't indicate any specific range as it is a logarithmic function over some value n.
Hence, this cannot be solved by master's theorem.

Example 8: T(n) = T$\left(\frac{n}{2}\right)$ + 2T$\left(\frac{n}{4}\right)$ + n. Master's theorem can be applied for only one set of 'a' and 'b' values (Means a definitive set of branching) Here, a and b values are not fixed and hence cannot be solved by Master's theorem.

Thus we realized that many divide and conquer problems cannot be solved by Master's theorem. Thus we will learn an extension which is a generalization of Master's theorem called the Akra-Bazzi recurrence solution.

Before learning Akra-Bazzi recurrences, to provide an intuition of the Master's theorem, lets see if Master's theorem can be written in a different form.

Let's consider Master Theorem,

$$T(n) = \Theta(n^{\log_a b}) + \sum_{j=0}^{\log_b n - 1} a^j f\left(\frac{n}{b^j}\right)$$

Here,

$n^{log_b a}$ = Number of Leaves in all the levels.
g(n) = $\sum_{j=0}^{\log_b n-1} a^j f\left(\frac{n}{b^j}\right)$ = cost of division and merging of all the levels combined. i.e., from 0 to (n-1) levels.
a^j is the number of subproblems at that level
$f\left(\frac{n}{b^j}\right)$ is the function that denotes the cost of dividing and merging of subproblems at each level.

Proof: $T(n) = (n^{\log_a b}) + \sum_{j=0}^{\log_b n-1} a^j f\left(\frac{n}{b^j}\right)$

Let us assume that p = $\log_a b$
Input size at level j, $n_j = \frac{n}{b^j}$

Now, $T(n) = (n^p) + \sum_{j=0}^{\log_b n-1} a^j f(n_j)$

Let us consider $\sum_{j=0}^{\log_b n-1} a^j f(n_j)$ which we can write in different form to generalize,

Since p = $\log_a b$,

$$a = b^p$$
$$a^j = (b^p)^j = (b^j)^p$$
$$a^j = \left(\frac{n}{\frac{n}{b^j}}\right)^p$$
$$a^j = \left(\frac{n}{n^j}\right)^p$$

This represents the number of subproblems at level j.

$$\text{Now, } \sum_{j=0}^{\log_b n-1} a^j f(n_j) = \sum_{j=0}^{\log_b n-1} \left(\frac{n}{n^j}\right)^p f(n_j)$$

Here, n_j takes log n different values based on j
For every recursion level(lets say 'm') which varies from nth level to the last level, lets say 1(since merging and computation starts from that level)

$$\text{Now, } \sum_{j=0}^{\log_b n-1} \left(\frac{n}{n^j}\right)^p f(n_j) = \sum_{m\epsilon\{b,b^2,...,\frac{n}{b^2},\frac{n}{b},n\}} \left(\frac{n}{m}\right)^p f(m)$$

Since, n^p is not changing with respect to the summation,

$$\sum_{m\epsilon\{b,b^2,...,\frac{n}{b^2},\frac{n}{b},n\}} \left(\frac{n}{m}\right)^p f(m) = n^p \sum_{m\epsilon\{b,b^2,...,\frac{n}{b^2},\frac{n}{b},n\}} \frac{f(m)}{m^p}$$

For some recursion level m, lets consider the function f(m),
In each level, $\{b, b^2, ..., \frac{n}{b^2}, \frac{n}{b}, n\}$ n changes by a constant factor b, and hence f(m) also changes by a constant factor.

where, f(m) =$\theta(f(n))$

$$\text{For one level, } \sum_{m=n} \frac{f(m)}{m^p} \rightarrow \sum_{m=\frac{n}{b}+1} \frac{f(m)}{m^p}$$

$$\text{From the above inference, } \sum_{m=\frac{n}{b}+1}^{n} \frac{f(m)}{m^p} = \theta\left(\sum_{m=\frac{n}{b}+1} \frac{f(n)}{n^p}\right)$$

Here, the factor $\frac{f(n)}{n^p}$ is a constant and not dependent of change factor m.

Therefore, $\theta\left(\sum_{m=\frac{n}{b}+1} \frac{f(n)}{n^p}\right) = \theta\left((n - \frac{n}{b} - 1 + 1\frac{f(n)}{n^p}\right)$

$= \theta\left(n(1 - \frac{1}{b})\frac{f(n)}{n^p}\right)$

For large values of b, $(1 - \frac{1}{b})$ will be 1

This implies, $\sum_{m=\frac{n}{b}+1}^{n} \frac{f(m)}{m^p} = \theta\left(n\frac{f(n)}{n^p}\right)$

From the above inference $\sum_{m=n} \frac{f(m)}{m^p} \rightarrow \sum_{m=\frac{n}{b}+1} \frac{f(m)}{m^p}$,

$\sum_{m=\frac{n}{b}+1}^{n} \frac{f(m)}{m^p} = \theta\left(n\sum_{m=n} \frac{f(m)}{m^p}\right)$

which can be written as,

$\sum_{m=n} \frac{f(m)}{m^p} = \theta\left(\frac{1}{n}\sum_{m=\frac{n}{b}+1} \frac{f(m)}{m^p}\right)$

$= \theta\left(\sum_{m=\frac{n}{b}+1} \frac{f(m)}{m^{(p+1)}}\right)$

Thus, $\sum_{j=0}^{\log_b n-1} \left(\frac{n}{n^j}\right)^p f\left(n_j\right) = \theta\left((n^p)\sum_{m=1}^{n} \frac{f(m)}{m^{(p+1)}}\right)$

■

Thus we came to a conclusion that in a complicated recurrence equation which has multiple a and b values can be solved in the above way which is very similar to akra-bazzi reference solution.

8.2 Akra-Bazzi Recurrences

By Master's theorem, T(n) = $\theta(n^p)$ for some value p which is $log_b a$

Let us consider a complex recurrence equation which has multiple a and b and does not have f(n),

T(n) = $a_1 T\left(\frac{n}{b_1}\right) + a_2 T\left(\frac{n}{b_2}\right) + a_3 T\left(\frac{n}{b_3}\right)$

T(n) = $a_1\left(\frac{n}{b_1}\right)^p + a_2\left(\frac{n}{b_2}\right)^p + a_3\left(\frac{n}{b_3}\right)^p$

T(n) = $\left(\frac{a_1}{b_1} + \frac{a_2}{b_2} + \frac{a_3}{b_3}\right) * (n)^p$

By Master's theorem, $\left(\frac{a_1}{b_1} + \frac{a_2}{b_2} + \frac{a_3}{b_3}\right) = 1$

This complex equation can be solved by finding a value of p for which $\left(\frac{a_1}{b_1} + \frac{a_2}{b_2} + \frac{a_3}{b_3}\right) = 1$

But if there is a cost of dividing and merging, i.e., f(n) in the equation then the above solution doesn't work.

8.2.1 Deterministic select

Input: An array A[q:r] of distinct elements, and integer $k\epsilon[1, r - q + 1]$

Output: An element x of A[q:r] such that rank(x,A[q:r]) = k

We want to find the kth smallest number here.

Algorithm:

- Let $n \leq 140$ some constant, for reference lets consider some value 140. Then sort the numbers of the array less than that constant. Here, it takes time complexity of $\theta(1)$.
- Otherwise if n is very large. Divide the array into groups of 5. Each group is of size similar to $\frac{n}{5}$.
- Then find the median of all those 5 groups.
- Now find the median of those 5 medians. Use the algorithm recursively to find the median of 5 medians.
- Do the recursive solving with the bi-partition median method.

This select method reduces the time complexity to solve the problem to a factor of n.

$SELECT(A, k)$: Given an unsorted set A of $n(= |A|)$ items, find the k^{th} smallest item in the set.

- All the items are arranged according to the values of the medians in the groups.
- x is the median of medians of the groups.
- At least half of the group is to the left of x and other to the right.
- Amongst the left half atleast half(indicated in green) are smaller than x and the other half of right half (indicated in blue) are larger than x. Cant say anything for the other elements outside the blue and green groups.
- Intuitively the items definitely smaller than x is $\geq 3\left((\frac{1}{2})(\frac{n}{5}) - 1\right) \geq \left(\frac{3n}{10} - 6\right)$
- the items definitely larger than x is $\geq 3\left((\frac{1}{2})(\frac{n}{5}) - 1\right) \geq \left(\frac{3n}{10} - 6\right)$
- Thus the items in any recursive call $\leq n - \left(\frac{3n}{10} - 6\right)$
- $n - \left(\frac{3n}{10} - 6\right)$ = $\left(\frac{7n}{10} + 6\right)$ which are the maximum number of elements on the other side.
- Thus the level of computations reduce by an order of 7.

Consider the following recurrence which describes the worst-case running time of the deterministic selection algorithm.

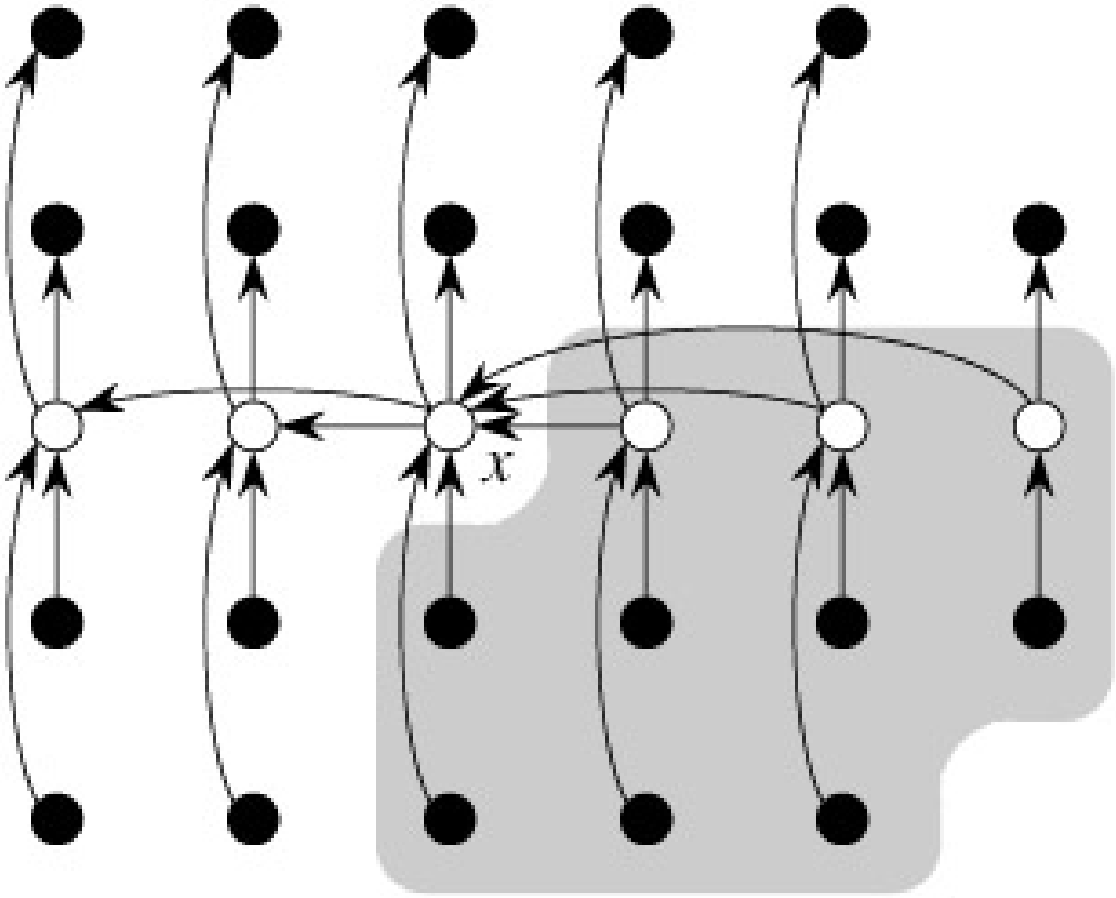

Figure 8.2.2: *SELECT(A,k)*

$$T(n) \leq \begin{cases} \theta 1, & if \quad n < 140, \\ T(\lceil \frac{n}{5} \rceil) + T\left(\frac{7n}{10} + 6\right) + \theta(n), & if \quad n \geq 140. \end{cases}$$

Here, $T(\lceil \frac{n}{5} \rceil)$ is the Time complexity of one median amongst a group of 5.

For a large value of n, $\frac{8n}{10} \geq \left(\frac{7n}{10} + 6\right)$

By simplifying, $n \geq 60$.

This assumption is based on the only case where n is sufficiently larger.

Thus we obtain the following upper bound for T(n),

$$T'(n) = \begin{cases} \theta 1, & if \quad n < 140, \\ T'(\frac{n}{5}) + T'\left(\frac{8n}{10}\right) + \theta(n), & if \quad n \geq 140. \end{cases}$$

If we consider, $\frac{7.5n}{10} \geq \left(\frac{7n}{10} + 6\right)$

By simplifying, $n \geq 120$.

Thus we obtain the following upper bound for T(n),

$$T''(n) = \begin{cases} \theta 1, & if \quad n < 140, \\ T''(\frac{n}{5}) + T'"\left(\frac{7.5n}{10}\right) + \theta(n), & if \quad n \geq 140. \end{cases}$$

Thus we need Akra-Bazzi recurrence solution for solving complex recurrences like the above.