.1 A useful recurrence

Generalized Algorithm: T(n) =

$$\left\{ \begin{array}{c} \Theta(1), if n <= 1 \\ \textbf{aT(n/b) + f(n) otherwise} \end{array} \right\}$$

1. **Karatsuba's Algorithm:**
$$T(n) = 3T(n/2) + \Theta(n)$$

2. **Strassen's Algorithm:**
$$T(n) = 7T(n/2) + \Theta(n^2)$$

3. **Fast Fourier Transform:**
$$T(n) = 2T(n/2) + \Theta(n)$$

.2 How the recurrence unfolds

$$T(n) = aT(n/b) + f(n)$$

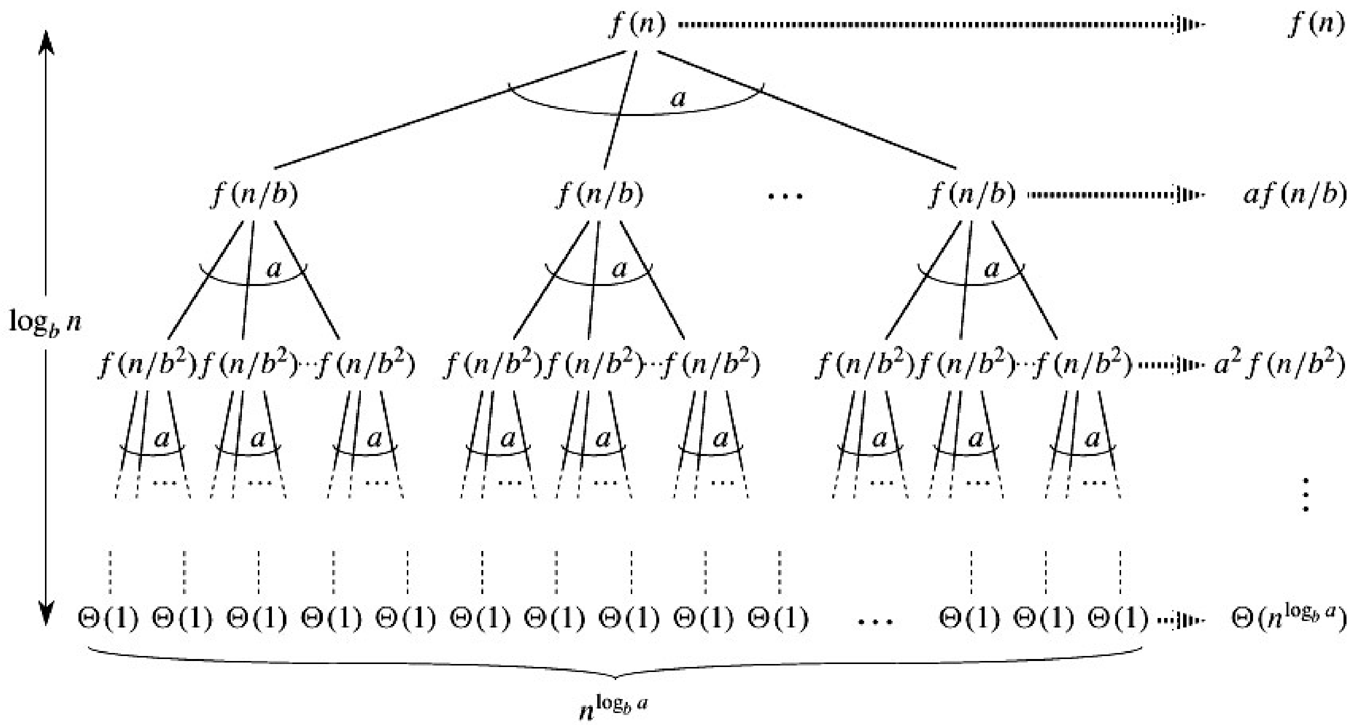

Figure .2.1: Unfolding of a recurrence algorithm

How the recurrence unfolds Case 1:

$$f(n) = \mathcal{O}\left(() \, n^{log_b a} - \epsilon\right)$$

for some constant

$$\epsilon > 0$$

Final answer:

$$T(n) = \theta(n^{log_b a})$$

Procedure:

Given:

$$T(n) = \theta(n^{log_b a - \epsilon}) + \sum_{j=0}^{log_b a - 1} a^j f(n/b^j)$$

$$f(n) = \theta(n^{log_b a - \epsilon}), for \epsilon > 0$$

Working:

$$g(n) = \sum_{j=0}^{log_b a - 1} a^j f(n/b^j)$$

After substituting the value of f(n), in the equation of g(n), we get:

$$g(n) = \sum_{j=0}^{log_b a - 1} a^j * \theta(((n/b)^j)^{log_b a - \epsilon})$$

We see that,

$$n^{log_b a - \epsilon} = Constant$$

Hence, removing the constant outside. We get,

$$n^{log_b a - \epsilon} * \theta(\sum_{j=0}^{log_b a - 1} a^j * 1/(b^j)^{log_b a - \epsilon})$$

$$n^{log_b a - \epsilon} * \theta(\sum_{j=0}^{log_b a - 1} a^j * (b^{j\epsilon})/(a^j))$$

After applying the infinite series formula. We get,

$$n^{log_b a - \epsilon} * \theta(((b^\epsilon)^{log_b a} - 1)/((b^\epsilon) - 1))$$

Here, we can neglect (-1), and after simplifying the equation. We get,

$$n^{log_b a - \epsilon} * n^\epsilon$$

Hence, we can conclude that:

$$T(n) = \theta(n^{log_b a})$$

How the recurrence unfolds Case 2:

$$f(n) = \Theta(n^{log_b a} * log^k n)$$

for k ¿=0
Final answer:

$$T(n) = \theta(n^{log_b a} * log^{(k+1)} n)$$

Procedure:
Given:

$$T(n) = \theta(n^{log_b a - \epsilon}) + \sum_{j=0}^{log_b a - 1} a^j f(n/b^j)$$

$$f(n) = \Theta(n^{log_b a} * log^k n)$$

Working:

$$\theta(\sum_{j=0}^{log_b a - 1} a^j * (n/b^j)^{log_b a} * log^k(n/b^j))$$

Removing the constants outside, we get:

$$\theta(n^{(log_b a)} * \sum_{j=0}^{log_b a - 1} a^j * (1/a^j) * log^k(n/b^j))$$

After simplifying, we get:

$$\theta(n^{(log_b a)} * \sum_{j=0}^{log_b a - 1} *(logn - j(logb))^k$$

Sub case 1: Upper bound

$$<= \theta(n^{(log_b a)} * \sum_{j=0}^{log_b a - 1} *(log^k n))$$

Now, in order to remove the summation. We can write the above equation as:

$$\theta(n^{(log_b a)} * (log^k n * logn))$$

Hence, the upper bound is:

$$\mathcal{O}\left(n^{(log_b a)} * log^{(k+1)} n\right)$$

Sub case 2: Lower bound

$$>= \sum_{j=0}^{1/2(log_2 n)} *(logn - j(logb))^k$$

Substituting the highest value of j from the summation, we get:

$$>= (logn - 1/2(logn))^k$$

$$>= \sum_{j=0}^{1/2(log_2 n)} (1/2) * (logn)^k$$

Hence, we can conclude that:

$$T(n) = \theta(n^{log_b a} * log^{(k+1)} n)$$

How the recurrence unfolds Case 3:

$$f(n) = \Omega(n^{log_b a + \epsilon})$$

and

$$a * f(n/b) <= c * f(n)$$

for constants

$$\epsilon > 0 c < 1$$

Final answer:

$$T(n) = \theta(f(n))$$

Procedure:
Given:

$$f(n) = \Omega(n^{log_b a + \epsilon})$$

$$a * f(n/b) <= c * f(n)$$

Working:

$$a * f(n/b) = a(n/b)^{(log_b a+\epsilon)}$$

$$= a * (n^{log_b a+\epsilon}/b^{log_b a+\epsilon})$$

$$= a * f(n)/a * b^{\epsilon}$$

$$= f(n)/b^{\epsilon}$$

Considering:

$$c < b^{(-\epsilon)}$$

We get:

$$< c * f(n)$$

for

$$b^{(-\epsilon)} < c < 1$$

Substituting n=n/b, we get:

$$f(n/b^2) <= c/a * f(n/b) <= (c/a)^2 * f(n)$$

$$a^2 * f(n/b^2) <= c^2 * f(n)$$

General problem:

$$a^j * f(n/b^j) < c^j * f(n)$$

$$g(n) = \sum_{j=0}^{log_b a-1} (a^j) * f(n/b^j)$$

$$<= \sum_{j=0}^{log_b a-1} c^j * f(n)$$

$$<= f(n) * \sum_{j=0}^{log_b a-1} c^j$$

$$<= f(n) * \sum_{j=0}^{\infty} c^j$$

$$<= f(n) * (1/(1-c))$$

The above term is a constant

Hence, we can conclude that:

$$g(n) = \mathcal{O}\left((f(n))\right)$$