Assignment 4 | Algorithms and Data Structures

Sakar Gopal Gurubacharya

Problem 4.2: *Recurrences*

Use the substitution method, the recursion tree, or the master theorem method to derive upper and lower bounds for T(n) in each of the following recurrences. Make the bounds as tight as possible. Assume that T(n) is constant for n ≤ 2.

a. $\mathbf{T(n) = 36T\left(\frac{n}{6}\right) + 2n}$

The recurrence is in the form

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

Therefore, we can use the master theorem as $\mathbf{a \geq 1}$ and $\mathbf{b > 1}$

$$a = 36$$
$$b = 6$$
$$f(n) = 2n$$

$$T(n) = n^{\log_b a}\,[u(n)]$$

We know n, a and b but we don't know u(n)

Anyways,

$$\boldsymbol{T(n) = n^{\log_6 36}\,[u(n)]}$$

$$T(n) = n^2\,[u(n)]$$

u(n) depends on h(n)

$$h(n) = \frac{f(n)}{n^{log_b a}}$$

$$h(n) = \frac{2n}{n^2}$$

$$h(n) = \frac{2}{n}$$

$$h(n) = 2n^{-1}$$

Comparing with kn^r, r = -1 which is less than 0

r < 0 == O (1)

Therefore,

$$[u(n)] = 1$$

So,

$$T(n) = n^2 \times 1$$

$$T(n) = n^2$$

$$\boldsymbol{T(n) = \theta(n^2)}$$

b. $\mathbf{T(n) = 5T\left(\frac{n}{3}\right) + 17n^{1.2}}$

The recurrence is in the form

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

Therefore, we can use the master theorem as $\mathbf{a \geq 1}$ and $\mathbf{b > 1}$

$$a = 5$$
$$b = 3$$
$$f(n) = 17n^{1.2}$$

$$T(n) = n^{\log_b a}\,[u(n)]$$

We know n, a and b but we don't know u(n)

Anyways,

$$\boldsymbol{T(n) = n^{\log_3 5}\,[u(n)]}$$

u(n) depends on h(n)

$$h(n) = \frac{f(n)}{n^{log_b a}}$$

$$h(n) = \frac{17n^{1.2}}{n^{\log_3 5}}$$

$$h(n) = \frac{17n^{1.2}}{n^{1.464973521}}$$

$$h(n) = 17n^{-0.264974}$$

Comparing with k$\boldsymbol{n^r}$, r = -0.264974 which is less than 0

r < 0 => O (1)

Therefore,

$$[u(n)] = 1$$

So,

$$T(n) = n^{\log_3 5} \times 1$$

$$T(n) = n^{\log_3 5}$$

$$T(n) = O(n^{\log_3 5})$$

$$\boldsymbol{T(n) = \theta(n^{1.46\ldots})}$$

c. $\mathbf{T(n) = 12T\left(\frac{n}{2}\right) + n^2 \lg n}$

The recurrence is in the form

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

Therefore, we can use the master theorem as $\mathbf{a \geq 1}$ and $\mathbf{b > 1}$

$$a = 12$$
$$b = 2$$
$$f(n) = n^2 \lg n$$

$$T(n) = n^{\log_b a}\,[u(n)]$$

We know n, a and b but we don't know u(n)

Anyways,

$$\boldsymbol{T(n) = n^{\log_2 12}\,[u(n)]}$$

u(n) depends on h(n)

$$h(n) = \frac{f(n)}{n^{log_b a}}$$

$$h(n) = \frac{n^2 \lg n}{n^{\log_2 12}}$$

$$h(n) = \frac{n^2 \lg n}{n^{3.584962501}}$$

$$h(n) = n^{-1.58496} \lg n$$

Comparing with k$\boldsymbol{n^r}$, r = $\mathbf{-1.58496}$ which is less than 0

r < 0 => O (1)

Therefore,

$$[u(n)] = 1$$

So,

$$T(n) = n^{\log_2 12} \times 1$$

$$T(n) = n^{\log_2 12}$$

$$T(n) = O(n^{\log_2 12})$$

$$\boldsymbol{T(n) = \theta(n^{3.58\ldots})}$$

d. $\mathbf{T(n) = 3T\left(\frac{n}{5}\right) + T\left(\frac{n}{2}\right) + 2^n}$

We know that,

$$T\left(\frac{n}{2}\right) > T\left(\frac{n}{5}\right)$$

Hence,

$$T(n) < 4T\left(\frac{n}{2}\right) + 2^n$$

The recurrence is in the form

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

Therefore, we can use the master theorem as $\mathbf{a \geq 1}$ and $\mathbf{b > 1}$

$$a = 4$$
$$b = 2$$
$$f(n) = 2^n$$

$$T(n) = n^{\log_b a}\,[u(n)]$$

We know n, a and b but we don't know u(n)

Anyways,

$$\boldsymbol{T(n) = n^{\log_2 4}\,[u(n)]}$$

u(n) depends on h(n)

$$h(n) = \frac{f(n)}{n^{log_b a}}$$

$$h(n) = \frac{2^n}{n^{\log_2 4}}$$

$$h(n) = \frac{2^n}{n^2}$$

$$h(n) = 2^n n^{-2}$$

Comparing with k$\boldsymbol{n^r}$, r = $\boldsymbol{-1.58496}$ which is less than 0

r < 0 => O (1)

Therefore,

$$[u(n)] = 2^n n^{-2}$$

So,

$$T(n) = n^{\log_2 4} \times 2^n n^{-2}$$

$$T(n) = n^2 \times 2^n \times n^{-2}$$

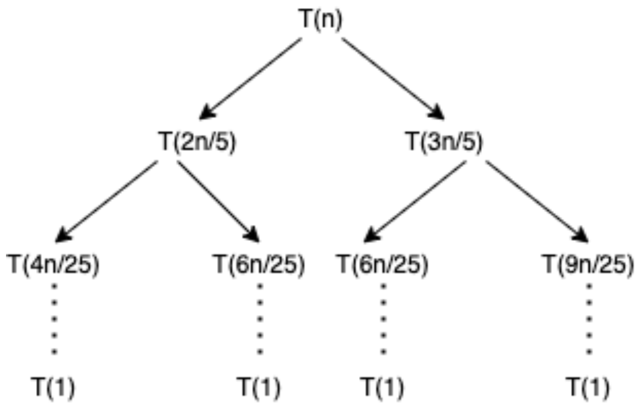
$$T(n) = O(2^n)$$

$$\boldsymbol{T(n) = \theta(2^n)}$$

This means that f(n) is polynomially larger than $n^{log_b a}$, and af(n/b) ≤ cf(n) for some constant c $<$ 1. That is why $\boldsymbol{T(n) = \theta(f(n))}$

e. $\mathbf{T(n) = T(2n/5) + T(3n/5) + \Theta(n)}$

Recursion Tree for T(n)



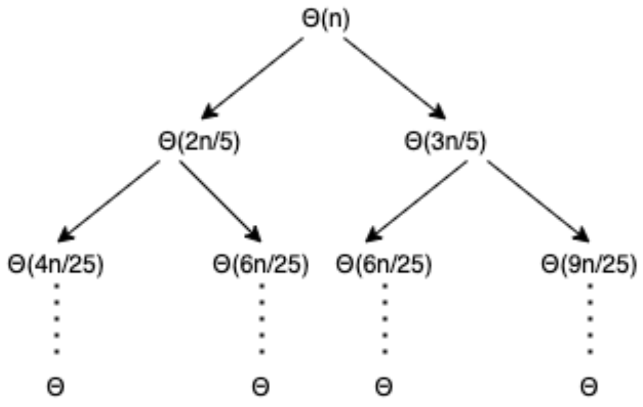
The shortest path is the most left one, because it operates on the lowest value, and the longest path is the most right one, because it operates on the highest value.

That results in the tree being unbalanced.

Shortest path can be defined as

$$\boldsymbol{n \to (\tfrac{2}{5})n \to (\tfrac{2}{5})^2 n \to \ldots \to 1}$$

Recursion Tree for Θ(n)



Sum of each complete level is equal to $\Theta(n)$

Elements from shortest path are being divided by 5, so length of this path will be equal to $\log_5 n$.

The time complexity of this algorithm is:

$$T(n) = \Theta(n) \log_5 n = \Omega(n\ lgn)$$

$$\boldsymbol{T(n) = \Omega(n\ lgn)}$$