Recurrence

Defⁿ - Recurrence eqⁿ recursively defines a sequence of fⁿ with different argument.
- Behaviour of recursive algorithm is better represented using recurrence eqⁿ.

1. The Substitution Method

→ **Forward substitution**
- This method finds the solⁿ of the smallest problem using base condition.
  A solution of the bigger problem is obtained using the previously computed solⁿ of the smaller problem.
  This process is repeated until the solⁿ for problem n is achieved.

→ **Backward Substitution**
- This method substitutes the value of n by n-1 recursively, to solve, smaller and smaller problem.
  It works exactly in reverse order of forward substitution

2. Master Method

- The master method concerns relations of the form:
  $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ where $a \geq 1, b > 1$
- In the application to the analysis of a recursive algorithm, the constants and function take on the following significance:
  - n is the size of the problem.
  - a is the no of subproblems in the recursion
  - n/b is the size of each subproblem
  - f(n) is the cost of work done outside the recursive calls, which includes the cost of dividing the problem and the cost of merging the solutions to the subproblems

Case 1:

Generic Form

If $f(n) \in O(n^c)$ where $c \backslash$ It $\log \frac{a}{b}$ (using big O notation) then:

$T(n) \in \Theta(n^{\log_b a})$

Example:

$a = 8,\ b = 2,\ f(n) = 1000n^2$

$f(n) \in O(n^c)$ where $c = 2$

Next, we see if we satisfy case 1 condition

$\log_b a = \log_2 8 = 3 > c$

It follows from the 1st case of the master thm.

$T(n) \in \Theta(n^{\log_b a}) = \Theta(n^3)$

(The exact solⁿ of the recurrence relⁿ is assuming.)

$T(n) = 1001n^3 - 1000n^2$; assuming $T(1) = 1$

Case 2:

Generic Form

If it is true, for some constant $K \geq 0$

$f(n) \in \Theta(n^c \log^k n)$ where $c = \log_b a$

then -

$T(n) \in \Theta(n^c \log^{k+1} n)$

Example:

$T(n) = 2T\left(\frac{n}{2}\right) + 10n$

$a = 2,\ b = 2,\ f(n) = 10n$

$f(n)\ \Theta(n^c \log^k n)$ where $c = 1,\ K = 0$

Next, we see if we satisfy case 2 condition

$\log_b a = \log_2 2 = 1$ and therefore yes, $c = \log_b a$

It follows from the 2nd case of the master theorem

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

Thus, the given recurrence relation T(n) was in $\Theta(n \log n)$.

(This result is confirmed by the exact solⁿ of the recurrence relation) →

$T(n) = n + 10n \log_2 2$ assuming $T(1) = 1$

Case 3:

Generic Form

If it is True that

$f(n) \in \omega(n^c)$ where $c > \log_b a$

and if it is also true that

$af\left(\frac{n}{b}\right) \leq kf(n)$ for some constant $k < 1$

and sufficiently large n (often called the regularity condition) then:

$T(n) \in \Theta(f(n))$

Example:

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

$a = 2,\ b = 2,\ f(n) = n^2$. so

$f(n)\ \omega(n^c)$ where $c = 2$

Next, we see if we satisfy case 3 condition

$\log_b a = \log_2 2 = 1$ and therefore yes

$c > \log_b a$

The regularity condition also holds

$2\left(\frac{n^2}{4}\right) \leq kn^2$ choosing $k = 1/2$

It follows from the 3rd case of the master thm

$T(n) = \Theta(f(n)) = \Theta(n^2)$

Thus the given recurrence relation T(n) was in $\Theta(n^2)$ that complies with the f(n) of the original formula

(This result is confirmed by the exact solⁿ of the recurrence relation which is, assuming $T(1) = 1$)

General format

$T(n) = a \cdot T(n/b) + f(n)$

$a \geq 1$

$b \geq 1$

$f(n) = +ve$

Cases

1) $f(n) < n^{\log_b a}$
   $T(n) = \Theta(n^{\log_b a})$
2) $f(n) = n^{\log_b a}$
   $T(n) = \Theta(n^{\log_b a} \cdot \log n)$
3) $f(n) > n^{\log_b a}$
   $T(n) = \Theta(f(n))$