Design and Analysis of Algorithm (DAA)

Recurrences

[Module 1]

Dr. Dayal Kumar Behera

School of Computer Engineering
KIIT Deemed to be University, Bhubaneswar, India

Recurrences

- When an algorithm contains recursive call to itself, its running time can be described by recurrence.

- Using recurrence, solution to a given problem can be expressed in terms of solution to smaller inputs of the same problem.

- Iterative algorithms are easy to analyse. But it's difficult to analyse Recurrence algorithm.

- For analysis of recursive algorithms, we need to formulate recurrence equation.

- **Recurrences are used to analyze the computational complexity of divide-and-conquer algorithms**.

  - Example: Merge Sort, Binary search are divide-and-conquer algorithms.

Example

Recurrence equation to find factorial

```
Algorithm  Factorial (n)
If n==1 or 0
       return 1
else
       return n*Factorial(n-1)
```

Factorial (n), how many times is factorial() called?

T (1) = 1
T (2) = T(1) + 1 = 2
T (3) = T(2) + 1 = 3
.
.
T (n) = T(n-1) + 1 = n

Analysis of Recursive Algorithm

To analyse recursive algorithm one need to

1. Formulating recurrence equation for given recursive problem
2. Solving the recurrence equation to understand the behaviour of the program.

Whether Recurrence Relation and Recurrence Equation are same??

A recurrence Relation is basically a definition of a function in terms of itself. For Example

$n! = n \times (n - 1)!$ With base condition as $1! = 1$

Where as a recurrence equation is a specific form of a recurrence relation.

For factorial

$T(n) = T(n-1) + 1$ or $t_n = t_{n-1} + 1$ where $t_1 = 1$

1 here indicates one additional multiplication is required..

A recurrence equation is also known as difference equation

Example

$T(n) = T(n-1) + 2$

$T(0) = 0;$

Can also be written as

$$T(n) = \begin{cases} T(n-1) + 2 & where\ n \geq 1 \\ 0 & where\ n = 0 \end{cases}$$

We can use a simple notation

$t_n = t_{n-1} + 2$

$t_0 = 0$

Substitute n=1, 2, 3…

t1 = t0 + 2 = 2

t2 = t1 + 2 = 4

t3 = t2 + 2 = 6

. . .

Even number sequence

But depends upon **Initial Condition**

Solving Recurrences/Recurrence Equations

- Recursion Tree method
- Master method
- Substitution method
- Iterative method/Iteration method

Solving Recurrences using Master Method

Theorem 4.1 (Master theorem)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $\boxed{af(n/b) \leq cf(n)}$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Regularity Condition

Master Method: Case-2, General Form

Case 2: $f(n) = \Theta(n^{\log_b a} \lg^k n)$, where $k \geq 0$.

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

Simple case: $k = 0 \Rightarrow f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \lg n)$

Master Method: Examples

Solve the following recurrence using master method

- $T(n) = 4\,T\left(\frac{n}{2}\right) + n$
- $T(n) = 4\,T\left(\frac{n}{2}\right) + n^2$
- $T(n) = 4\,T\left(\frac{n}{2}\right) + n^3$
- $T(n) = 7\,T\left(\frac{n}{2}\right) + n^2$
- $T(n) = 2\,T\left(\frac{n}{4}\right) + \sqrt{n}$
- $T(n) = 16\,T\left(\frac{n}{4}\right) + n^2$
- $T(n) = 2\,T\left(\frac{n}{2}\right) + n$
- $T(n) = 2\,T\left(\frac{n}{2}\right) + n^3$

Master Method: Limitations

- It is worth noting that the three cases of Master Theorem do not cover all the possibilities for f(n)
  - There is a gap between cases 1 and 2 when f(n) is smaller than $n^{\log_b a}$ but not polynomially smaller
  - Similarly, there is a gap between cases 2 and 3 when f(n) is larger than $n^{\log_b a}$, but not polynomially larger
- If the function f(n) falls into one of the above-mentioned gaps, or if the regularity condition in case 3 fails to hold, we cannot apply the Master method to solve the recurrence.
- However, any other method may be applied for solving the same.


Each of your
actions will
have an
impact on your
future.


Once you know
who is walking
with you on your path.
you will never
be afraid.

Thank you