Solving Recurrence (Part 1)

CS-6th

Instructor: Dr. Ayesha Enayet

Definition

- A recurrence relation for the sequence a0,a1,a2,…a_n is an equation that relates a_n to certain of its predecessors $a_0, a_1, \ldots a_{n-1}$.
- Initial condition for the sequence $a_0, a_1, \ldots a_{n-1}$ are explicitly given values for a finite number of the terms of the sequence.
- A recurrence relation and initial conditions can be used to define sequence.
- We often use a recurrence relation to describe the time required by the algorithm, especially recursive algorithm.

Reference: Algorithms By Richard Johnsonbaugh, Marcus Schaefer

Example (Fibonacci sequence)

$f_n=f_{n-1}+f_{n-2}$, $n>=3$,

Initial condition:

$f_1=f_2=1$ (for series 1,1,2,3,5….)

A more formal way…

Fibonacci numbers

We define the ***Fibonacci numbers*** F_i, for $i \geq 0$, as follows:

$$F_i = \begin{cases} 0 & \text{if } i = 0\,, \\ 1 & \text{if } i = 1\,, \\ F_{i-1} + F_{i-2} & \text{if } i \geq 2\,. \end{cases} \tag{3.31}$$

Thus, after the first two, each Fibonacci number is the sum of the two previous ones, yielding the sequence

0,1,1,2,3,5,8,13,21,34,55,. . ..

Fibonacci series: Time Complexity

Fibonacci(n)

If n=0 or n=1, return n

return Fibonacci(n-1) + Fibonacci(n-2)

1. $T(n)=T(n-1)+T(n-2)+c$ where c is a time taken by operations other than recursive calls
2. $T(0)=T(1)=1$
3. $T(n-1)\approx T(n-2)$
4. $T(n)=2.T(n-1)+c$

Time Complexity (substitution)

$T(n)=2.T(n-1)+c$------$\rightarrow$eq1

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

$T(n-1)=2.T(n-2)+c$-----$\rightarrow$eq2

Substitute eq2 in eq1

$T(n)=2(2.T(n-2)+c)+c$

$T(n)=2(2(2.T(n-3)+c)+c)+c$

$T(n)=2^3.T(n-3)+4c+2c+c$-$\rightarrow$eq3

$T(n)=2^k.T(n-k)+c(2^k-1)$

For n=k:

- $T(n)=2^n.T(n-n)+c(2^n-1)$
- $T(n)=2^n.T(0)+c(2^n-1)$

  Where $T(0)=1$
- $T(n)=2^n+c(2^n-1)$
- $T(n)=O(2^n)$

generic form for eq3

- $T(n)=2^3.T(n-3)+4c+2c+c \text{-}\rightarrow eq3$
- $A(n)=b^3.A(n-3)+b^2.f(n)+b^1.f(n)+b^0.f(n)$
- $A(n)=b^n.A(n-n)+b^{n-1}.f(n)+b^{n-2}.f(n)+\ldots+b^0.f(n)$
- $A(n)=b^n.c+\sum_{j=0}^{n-1} f(n)$