Recurrence Relations

Recurrence relations can be directly derived from a recursive algorithm, but they are in a form that does not allow us to quickly determine how efficient the algorithm is. To do that we need to convert the set of recursive equations into what is called closed form by removing the recursive nature of the equations. This is done by a series of repeated substitutions until we can see the pattern that develops.

The easiest way to see this process is by a series of examples.

**Example 1:**

Test(n): ----T(n)

If n > 0 : ---1

Print(n) -----1

Test(n-1)---T(n-1)

T(n)= 1 if n=0 base case

= T(n-1)+ 2 if n>0 Recursive Case

T(n)=T(n-1)+ 2 ---Eq1

Put n=n-1

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

T(n-1)= T(n-2)+2 --- Eq2

Put n=n-2 in Eq1

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

T(n-2)=T(n-3)+2 ---Eq3

T(n)=T(n-1)+ 2 ---Eq1

Now back Substitute Eq2 in Eq1

T(n)=[T(n-2)+2]+2

T(n)=T(n-2)+4---Eq4

Again back Substitute Eq3 in Eq4

T(n)=T(n-3)+2+4

T(n)=T(n-3)+6 – Eq6

Observe Similarities in Eq 1, 4 and 6

Step 1 has n-1 and 2\*1

Step 2 has n-2 and 2\*2

Step 3 has n-3 and 2\*3

Step k will have n-k and 2\*k

So we can Generalize our Recurrence Relation to

T(n)=T(n-k)+ 2\*k

How long should we Continue?

When n=0 and T(0) = 1 base Case

So n-k =0 for us to reach T(0)

k=n

T(n)=T(n-n)+2\*n

T(n)=T(0)+2\*n

T(n)=1+2n

T(n)=O(n)

**Merge Sort / Quick Sort (Best Case) Complexity Proof:**

T(n)=2T(n/2)+n for n>1

T(1)=1 for n=1

T(n)=2T(n/2)+n -----EQ(1)

Replace n with n/2 in EQ(1)

T(n/2)=2T(n/4)+n/2 ----EQ(2)

Replace n with n/4 in EQ(2)

T(n/4)=2T(n/8)+n/4 ----EQ(3)

Back Substitute EQ(2) in EQ(1)

T(n)=2[2T(n/4)+n/2]+n

T(n)=4T(n/4)+2n

Back Substitute EQ(3) in EQ(1)

T(n)=4[2T(n/8)+n/4]+2n

T(n)=8T(n/8)+ 3n

Generalize..

T(n)=2k T(n/2k)+kn

The base case will occur when

n/2k =1

n=2k

k=log n

T(n)=n\*T(1)+log(n)\*n

T(n)=n+nlogn

T(n)=O(nlogn)

**Complexity of Quick Sort Worst Case Proof**

T(n)=T(n-1)+n for n>1--- EQ(1)

T(1)=1 for n=1

Replace n with n-1 in EQ(1)

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

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

Replace n with n-2 in EQ(1)

T(n-2)=T(n-3)+n-2 --- EQ(3)

Back Substitute EQ(2) in EQ(1)

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

T(n)=T(n-2)+2n-1 --- EQ(4)

Back Substitute EQ(3) in EQ(4)

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

T(n)=T(n-3)+3n-3

Generalize

T(n)=T(n-k)+K(n-1)

Base case is when

n-k=1

n=k

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

T(n)=n2-n

T(n)=O(n2)