【1】

The complexity of “Fib\_Re.m” is Θ(n-2)2^n. The reason is as follow:

For simplicity, suppose we always want to calculate an even term of the progression.

Also, we define t(n) as the time needed to complete the calculation of the nth term in the progression, T(n) as the number of terminal nodes at layer n, N(n) as the number of nodes at layer n.

Then as the recursion goes on, N(n) would be like a tower:

2N n=1

2N-1 2N-2 n=2

……

2N-(N-1) …… 2N-(2N-2) n=N

……

…

2 1 n=2N-1

There are 2^(n-1) terms on each of the first n layers of the tower, i.e., N(n)=2^(n-1) when 1<=n<=N

And there are [2^n-(2^1+2^2+…+2^i)] on layer (N+i), 1<=i<=n-1, i.e., N(N+i)= [2^N-(2^1+2^2+…+2^i)] , 1<=i<=N-1

Terminal nodes only exist when n>=N, and the relation between T and N is as follow:

T(2N-1)=N(2N-1)=2

T(2N-2)=N(2N-2)-1/2N(2N-1)

T(2N-3)=N(2N-3)-1/2N(2N-2)

……

T(2N-N)=N(2N-N)-1/2N(2N-N+1)

Apparently Sum(T(N:2N-1))=N(N)+1/2Sum(N(N+1:2N-1))

=2N-1+1/2[]

=2N-1+1/2[(N-1)2N-]

=N\*2N-1-2(2N-1-1)+(N-1)

=(N-2)2N-1+N+1

Then, t(2N)=t(2N-1)+t(2N-2)

That is (n-2)\*2^n+2n+1, which isΘ(n-2)2^n.

The complexity of calculating odd terms are similar, also beingΘ(n-2)2^n.

\*:At terminal nodes (when n=2 or n-1), the number of operations needed is not 1, but it is a constant number and is not likely to affect the magnitude(complexity) when n gets very big.

【2】