Lab2 continued

1. Prove by induction that for all n > 4, Fn > . Then use this result to explain the approximate asymptotic running time of the recursive algorithm for computing the Fibonacci numbers. Is the recursive Fibonacci algorithm fast or slow? Why? Hint #1. For the base case of your proof, the following table of values may be useful:

Solu:

Base case: From the table, we know F(4) > , F(3) > , F(2) > .

Induction step: For F(n), if n>4

F(n+1) – = F(n) + F(n-1) > + = (1 + ) = > 0

Conclusion: for all n > 4, F(n) > .

.

T(n) = T(n-1) + T(n-2) + 1 = T(n-2) + T(n-3) + 1 + T(n-3) + T(n-4) + 1 + 1 = T(n-3) + T(n-4) + 1 + T(n-4) + T(n-5) + 1 + T(n-4) + T(n-5) + 1 + T(n-5) + T(n-6) + 1 =…

In each step we call T twice, thus will conclude below:  
T(n) = 2⋅2⋅...⋅2 + c = 2ⁿ + c;

T(n) = O(2ⁿ) is very slow.

2. More big-oh: (Work with someone who is familiar with limits)

a. True or false: is O(). Prove your answer.

b. True or false: log n is Θ(log3 n). Prove your answer.

c. True or false: (n/2) log(n/2) is Θ(nlog n). Prove your answer.

Solu:

1. False.

For each c, while n>logc, we have >\*c

So is not O().

1. True.

 = =

So log n is O(log3 n),

And  = =

So log3 n is O (log n),

So log n is Θ(log3 n).

1. True.

==1/2

So (n/2) log(n/2) is O(nlog n),

 =2

So nlog n is O( (n/2) log(n/2)),

So (n/2) log(n/2) is Θ(nlog n),

P3:

* T(1) = 1
* T(2) = T(1) + 1 = 2
* T(3) = T(2) + 1 = 3
* T(4) = T(3) + 1 = 4
* ...
* T(n) = n

the worst-case asymptotic running time of this algorithm O(n)

Induction step:

T(1) = 1, T(n) = n,

T(n+1) = T(n) + 1 = n+1;

P4:

T(n) = n + 1 = O(n); calculate step is included in the Java code

P5:

asymptotic running time :T(n) = O(log n)

If n > 1

T(n) = T(n/2) + n = T(n/4) + n/2 + n = T(n/8) + n/4 + n/2 + n =… = T(1) + 1 + 2 + 4

+…+ n;

Obviously, the plus time is log n, so T(n) = O(log n)

P6:

The worst-case: all elements are 0s or all elements are 1s

T(n) = T(n/2) + T(n/4) +… + T(1) = O(log n) = o(n)