Submitted to

Subroto Nag Pinku

Daffodil Int. University

Submitted by

Faraz Ahmed

Id:191-15-12948

Sec:O14

ASSIGNMENT 1

1. What is optimization?

ANS: In computer science optimization is referred to software optimization. Software optimization is to make that software faster and bug free for that specific word.

Optimization also can be referred for code or system optimization. Code optimization mainly depends on choosing the correct algorithm to make it faster and correct data structure etc.

2. What are the different algorithms you know?

ANS: Basic sorting and searching algorithms like- merge sort, selection sort, binary search etc.

3. Why are you learning so many algorithms?

ANS: We are learning so many algorithms to use those basic methods

to give more efficient solution for codes or complex problems we do every day.

4. Show analysis of a recursive algorithm.

```
Factorial (n)
{
If(n==0)
Return 1;
Else
n*Factorial(n-1)
}
```

Time complexity analysis:

Let's assume ==,*,- operator's cost constant 1 time.

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

= T(n-1)+3

=T(n-2)+6

=T(n-3)+8

=T(n-k)+3k

T(0)=1

In terms of T(0)

n-k=0

n=k

T(n)= T(0)+3n

T(n)=3n+1

Time complexity in the worst case: O(n)

5. Design an iterative and recursive algorithm and prove that your algorithm works

Recursive Fibonacci vs. iterative method

```
int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}
```

Iterative method

```
int fib(int n)
{

    int f[n + 2];
```

```
    int i;


    f[0] = 0;
    f[1] = 1;

    for(i = 2; i <= n; i++)
    {


       f[i] = f[i - 1] + f[i - 2];
    }
    return f[n];
    }
};
```

Time complexity analysis:

Recursive method:

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

We can assume that $T(n-2) = T(n-1)$**.**

Substituting the value of $T(n-1) = T(n-2)$ **into our relation** $T(n)$**, we get:**

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

$T(n) = 2*[2*T(n-2) + 1] + 1 = 4*T(n-2) + 3$

Next, we can substitute in $T(n-2) = 2*T(n-3) + 1$**:**

$T(n) = 2*[2*[2*T(n-3) + 1] + 1] + 1 = 8*T(n-3) + 7$

And again for $T(n-3) = 2*T(n-4) + 1$**:**

$T(n) = 2*[2*[2*[2*T(n-4) + 1]+ 1] + 1] + 1 = 16*T(n-4) + 15$

We can see a pattern starting to emerge here, so let's attempt to form a general solution for $T(n)$. It appears to stand that:

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

In this function 2^k is the term that has the highest rate of change for different values .That's why we can assume that

Time complexity is $O(2^n)$

While the iterative method is O(n)