Name: Tandra Rani Karmakar

ID:191-15-12134

Sec: o14

Assignment-1

1.Optimization:
In computer science optimization is the process of modifying a software system to make some aspect of it work more efficiently or use fewer resources (CPU, Memory) and deliver high speed. Optimization can be categorized broadly into two types: machine independent and machine dependent.

2. Different Algorithms that I know:
Selection sort, bubble sort, heapsort, insertion, quicksort, Complexity and running time etc.

3. Why I am learning so many algorithms:
I'm learning so many algorithms Cause I want to know that my code which I'm doing is based on ideas that solve the problem and that I'm using resources efficiently. And I want to know that my solution is correct and efficient for all possible situations, or at least to know that the cases in which my any algorithm fails to meet these criteria are rare.

4. 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.Recursive Fibonacci vs. iterative method:
intfib(intn) {
if(n <= 1)
returnn; returnfib(n-1) + fib(n-2); } Iterative method
intfib(intn) {
intf[n + 2]; inti;
f[0] = 0; f[1] = 1;
for(i = 2; i <= n; i++) {
f[i] = f[i - 1] + f[i - 2]; } returnf[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)