Mohammad Aves
13 CST

Tutorial II

Ans(1) Time complexity - O(sqrt n)

1st time $i = 1$

2nd time $i = 3$ $(i = 1+2)$

3rd time $i = 6$ $(i = 1+2+3)$

⋮

n^{th} time $i = \frac{x(x+1)}{2} = \frac{x^2}{x} + \frac{x}{2} = x^2 < n$

$x = sqrt(n)$

Ans(2) $fib(n) = fib(n-1) + fib(n-2)$

Let $T(0) = 1$

```
fib(n)
    if n<=1
        return 1
    return fib(n-1) + fib(n-2)
```

(Time complexity)

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

$= 2T(n-2) + c$ $(\text{as } T(n-1) \simeq T(n-2))$

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

$= 2*(2T(n-4)+2C)+C$

$= 4T(n-4)+3C$

$T(n-4) = 2*(4T(n-4)+3C)+C$

$= 8T(n-3)+7C$

$= 2^k xT(n-3k)+(2^k-1)C$

$(n-k=0 \Rightarrow n=k \quad , \boxed{k=n})$

$T(n) = 2^n * T(0) + 2^n C - C$

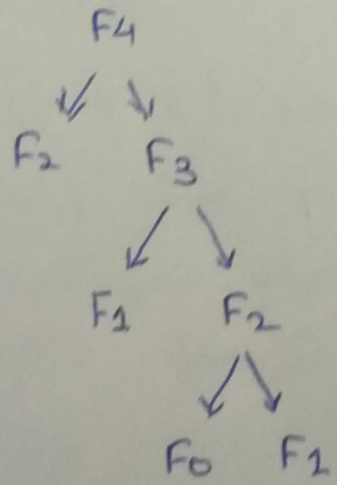
$= 2^n * 1 + 2^n C - C$

$= 2^n(1+C) - C$

$T(n) \simeq 2^n$ // Constant can be ignored

$= O(2^n)$

Space complexity The space is proportional to the maximum depth of the recursion tree



Hence the space complexity of Fibonacci recursive is $O(N)$

Ans(3) Merge sort = $n(\log(n))$

* For time complexity = n^3

we can use three nested loops = $O(n^3)$

```
For(int i=0; i<n; i++)
  {
   For (int j=0; j<n; j++)
   {
     For(int k=0; k<n; k++)
     {
       Some O(1) expression
     }
   }
  }
```

* For time complexity - $\log(\log(n))$

we use function

```
For(int i=2; i<n; i=pow(i,c)
{
  // Some O(1) expressions
}
```

∀ k is constant

* For time complexity $n \log n$

we can use function

```
int fun(int n)
{
  For (i=1; i<=n; i++)
  {
```

```
For( j=1 ; j<=n ; j+=i)
{
  some O(1) expressions
}
}
```

Ans (4) $T(n) = 2T(n/2) + cn^2$ // $T(\frac{n}{2}) \geq T(n/4)$

using masters method

$T(n) = \Theta T(n/b) + f(n)$

$a \geq 1$, $b > 1$, $c = \log_b a$ comparing n^c & $f(n)$

we get $c = \log_2 2 = 1$

$f(n) > n^c$

$T(n) = \Theta(f(n))$

$T(n) = \Theta(n^2)$

Ans (5) For $i = 1 \rightarrow j = 1, 2, 3 - - n$ (sum for n times)

For $i = 2 \rightarrow j = 1, 3, 5 - - n$ (sum for $n/2$ times)

For $i = 3 \rightarrow j = 1, 4, 7 - - - -$ (sum for $n/3$ times)

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

$= n(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + - - -)$

$= n \int_1^n \frac{1}{x}$

$\Rightarrow n \int_1^n \frac{dx}{x}$

Nice

$\Rightarrow [\log x]_1^n \cdot n$

$\Rightarrow (\log(n) - \log(1))\, n$

$\Rightarrow n \log(n)$

The time complexity following function is $n \log n$.

Ans(6) For first iteration $i = 2$

2nd iteration $i = 2^{\wedge K}$

3rd iteration $i = (2^k)^k = 2^{k^2}$

⋮

h^{th} iteration $i = 2^{k^i}$ (loop ends at $2^{k^i} = n$

(apply log)

$\log(n) = \log 2^{k^i}$

$k^i = \log(n)$

(again apply log)

$\log(k^i) = \log(n)$

$i = \log_k(\log(n))$

Ans (8)

(a) $100 < \log(\log n) < \log(n!) < \log^2 n < \sqrt{n} < n$
$< n\log n < n^2 < 2^n < 4^n < 2^{2^n} < \log(n!) < n!$

(b) $1 < \log(\log(n)) < \overline{1}\log(n) < \log(n) < \log 2N <$
$2\log(n) < n < 2n < 4n < n\log(n) < n^2$
$< \log(n!) < n! < 2(2^n)$

(c) $96 < \log_8(n) < \log_2(n) < 5n < n\log_c n$
$< n\log_2 n < n! < \log(n!) < \theta^{2n}.$