Name - Shanjal Gupta
Sec - 15
Rollno - 35
Stu id - 2002489

Date
Page No.

Tutorial - 2

Soln 1. When while loop executes -

At first Pass $i = 1$

2nd Pass $i = 1+2$

3rd pass $i = 1+2+3$

Similarly 4th $i = 1+2+3+4$

. . . . . nth $i = 1+2+3+\dots+n$

for i^{th} time $i = (1+2+3+4+\dots+i) < n$

$$= \frac{i(i+1)}{2} < n$$

$$= \left(\frac{i^2}{2} + \frac{i}{2}\right) < n$$

ignoring $\frac{i}{2}$ & $\frac{1}{2}$

After neglecting we left with

$$= i^2 < n$$

$$= i < \sqrt{n}$$

Hence the time complexity is $O(\sqrt{n})$

Soln 2.

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

Time complexity:-

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

when $n=0$ & $n=1$

i.e., $T(0) = T(1) = 0$

for $T(n) = ?$

Here $T(n-2) \approx T(n-1)$

On substituting the value of $T(n-1) = T(n-2)$ into $T(n)$,

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

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

on substituting of

$T(n) = 2 \times [2 \times T(n-2) + 1] + 1$

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

$T(n-2) = 2T(n-3) + 1$

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

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

$T(n) = 16 * T(n-4) + 15$

Similarly for k^{th} term

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

Take $n-k = 0$

$n = k$

Hence, $T(n) = 2^n * T(0) + (2^n - 1)$

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

So, time complexity is $O(2^n)$

Space complexity:-
Here n are the no. of entries in a stack & for each function call one,
So space complexity for each case (call) is 1, i.e. O(1)
& for n no. of cases, <= n
i.e O(n)

Soln 3. For (int i=0; i<n; i++) {
for (int j=0; j<n; j=j*2)
{
O(1) // statements.
}
}
O(n(logn))

For (int i=0; i<n; i++){
for (int j=0; j<n; j++) {
for (int k=0; k<n; k++){
O(1) // (Statements)
}
}
}
O(n^3)

```
for(int i=0; i<n; i=i/2){
  for(int j=0; j<n; j=j*2)
  {
    O(1)}
  }O(1)}
}
}
```

Statements

$O(\log(\log n))$

4. $T(n) = T(n/4) + T(n/2) + cn^2$

On removing $T(n/4)$ as smaller term

$T(n) = T(\frac{n}{2}) + cn^2$

On applying Master's theorem on R/H/S

~~T~~ $a=0, b=2$ $k=2$, $p=0$.

$\log_b^a = \log_2^0 = 0.$

$0 < 2$ i.e $\log_b^a < k$

& $p \geq 0$

$O(n^k \log^p n)$

so, $O(n^2 \log^0 n)$

$O(n^2)$

Ans

Date
Page No.

Ans/ time complexity of the function fun( ) is O(n log n)

For i = 1, inner loop Executed n times.
For i = 2, inner loop executed n/2 times.
for i = 3, inner loop executed n/3 times.
for i = n, inner loop executed n/n times.

So, Complexity is like as.

$$n\left(n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}\right)$$

$$n\left(1/1 + \frac{1}{2} + \frac{1}{3} + \cdots + 1/n\right)$$

which becomes like ↓

Forms h.p $\rightarrow \left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$ —

Particular term time complexity is (log n)

So for total 2 loops time complexity is O(n(log n)).

Sol/Ans. for(int i = 2; i < n; i = pow(i, k)){

// O(1) — Expressions.

}

k is constant.

i takes the value like $2, 2^k, 2^{k^2}, 2^{k^3}, \ldots, 2^{k\log_k(\log n)}$

last term must be less than or equal to n.

$O(\log_k(\log(n)))$

Date
Page No.

Soln

8. (a) ~~$1 < n < n! < \log n < \log(n!) < \log(\log n) < n \log n < \log^2 n < 2^n < 4^n < 2^{(2^n)} < n^2$~~

(a) $1 < \log n < \log(n!) < \log(\log n) < n < n! < n \log n < \log^2 n < 2^n < 4^n < 2^{(2^n)} < n^2$

(b) ~~$1 < \log(n) < \log(n!) < \log(\log n) <$~~

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

(c) $96 < \log_8(n) < \log_2(n) < \log(n!) < n! < n \log_6(n) < n \log_2(n) < 5n < 8n^2 < 8^{(2n)} < 7n^3$