PAGE No.
DATE

Upendra Pandit
Section - D
Roll no. - 48
University Roll no. - 20021897

Tutorial - 2

Sol 1. When while loop executes:-

At first pass $i = 1$

2nd pass $i = 1+2$

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

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

$1 \ldots\ldots$ nth $i = 1+2+3+\ldots+n$

for ith time $i = (1+2+3+4+\ldots i) < n$

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

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

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

After neglecting we are left with

$$= i^2 < n$$

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

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

Scanned with CamScanner

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 $T(n)$

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

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

On substituting

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

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

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

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

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

⋮

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

Similarly for k^{th} term

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

$n - k = 0$

$n = k$

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.

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

$O(n(\log n))$

```
for (int i=0;  i<n ; i++){
   for (int j=0;  j<n ; j++){
      for (int k=0;   k<n ; k++){

            O(1)   =   // (statement)
      }
   }
}
```

$O(n^3)$

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

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

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

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

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

on applying master's theorem on RHS.

$a = 0$, $b = 2$, $k = 2$, $p = 0$

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

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

& $p \geq 0$

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

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

$\Theta(n^2)$

Soln 5. time complexity of the function fun() is $\Theta(n \log n)$

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 $i=4$, inner loop executed n/n times

So, complexity

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

$$n\left(1/_1 + \frac{1}{2} + \frac{1}{3} \dots \frac{1}{n}\right)$$

which becomes like

forms h.p $\rightarrow$ $\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots \frac{1}{n}\right)$ -

particular form time complexity is $(\log n)$

So ~~fo~~ for total 8 loops
time complexity is $O(n(\log n))$

Soln 6. for (int i = 2; i<n ; i = pow(i,k){

// O(1) — expression

}

~~k~~ k is constant.

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

last term must be less than or equal to n

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

Soln. 8

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

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

(c) $96 < \log_8(n) < \log_2(n) < \log(n!) < n!$
$< n\log_6(n) < 6n\log_2(n) < 5n < 8n^2$
$< 8^{(2n)} < 7n^3.$