Maneesh Cheema
ML 42
2015251
DAA Assignment 01

(1) Asymptotic notations are used to representation execution time of an algorithm as a function of input size.

(a) Big theta (Θ) : Gives tight upper and lower bound

(b) Big oh (O) : Gives tight upper bound (worst case)

(c) Big omega (Ω) : Gives tight lower bound (best case)

(d) Small oh (o) : Gives upper bound

(e) Small omega (ω) : Gives lower bound.

(2)

```
for (i=1 to n)
    i = i * 2
```

$\Rightarrow$ 1, 2, 4, 8, 16 ... n

{GP}

$a = 1, r = 2$ $\quad t_k = a r^{k-1}$

$n = 1 \cdot 2^{k-1}$

$n = 2^{k-1}$

$\log n = (k-1) \log 2$

$k = \log n + 1$

$\Rightarrow 1 + 1 + (\log_2 n + 1) + (\log_2 n + 1) \ldots$

$\Rightarrow O(\log_2 n)$

(3) $T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0 \\ 1 & \text{else} \end{cases}$

$T(n) = 3T(n-1)$ — (1)

putting $n = n-1$

$T(n-1) = 3T(n-2)$ put in (1)

$T(n) = 3[3T(n-2)] = 9T(n-2)$ — (2)

put $n = n-2$ in (1)

$T(n-2) = 3T(n-3)$ putting in (2)

$T(n) = 9[3T(n-3)] = 27T(n-3)$

$\vdots$

$T(n) = 3^k T(n-k)$

For $T(0) = 1$, $n = 0$, $n-k = 0 \Rightarrow k = n$

$T(n) = 3^n T(n-n)$

$T(n) = 3^n T(0)$

$T(n) = 3^n$ Ans.

(4) $T(n) = 2T(n-1) - 1$ — (1)

put $n = (n-1)$ in (1)

$T(n-1) = 2T(n-2) - 1$ putting in (1)

$T(n) = 2[2T(n-2) - 1] - 1$

$\Rightarrow 4T(n-2) - 2$ — (11)

put $n = n-2$ in (1)

$T(n-2) = 2T(n-3) - 1$ put in (11)

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

$\Rightarrow 8T(n-3) - 3$

$\vdots$

$T(n) = 2^k T(n-k) - k$

For $T(0) = 1 \quad n-k=0 \quad , n=k$

$T(n) = 2^n T(n-n) - n$

$\Rightarrow 2^n T(0) - n$

$\Rightarrow 2^n - n$

$T(n) = O(2^n)$ Ans.

(5)

```
int i=1, s=1 -①
while (s<=n){
  i++; -①
  s=s+i; — k times
  printf("#"); -①
}
```

1, 3, 6, 10, 15 ..... n

1, 2, 3, 4 .... k

$\sum_{i=1}^{k} i \Rightarrow \frac{k(k-1)}{2} \Rightarrow n$

$T(n) = \frac{k^2-k}{2} = k^2 \Rightarrow n \quad k=\sqrt{n}$

$T(n) = O(\sqrt{n})$

(6)

```
void function(int n){
int i, count=0; -①
for (i=1; i*i<=n; i++) -√n times
    count++ -①
```

$T(n) = 1+\sqrt{n}+1$

$T(n) = O(\sqrt{n})$

$1+4+9+16 \ldots n$

$1+2+3+4 \ldots \sqrt{n}$.

(7) void function(int n){
int i, j, k, count = 0;
for(i=n/2; i<=n; i++)
for(j=1; j<n; j=j*2)
for(k=1; k<=n; k=k*2))
count++;

$\sum_{i=n/2}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1 \Rightarrow \sum_{i=n/2}^{n} \sum_{j=1}^{n} \log n$

(i++=1)(j=j*2) (j=j*2)

$\Rightarrow \sum_{j=n/2}^{n} (\log n)(\log n) \Rightarrow \sum_{j=n/2}^{n} \log^2 n \Rightarrow \frac{n}{2} \log^2 n$

$T(n) = O(n \log^2 n)$

(8) $\sum_{i=1}^{n} \sum_{j=1}^{n} 1 \Rightarrow \sum_{i=1}^{n} n \Rightarrow n^2$

$T(n) = T(n-3) + n^2$ — (I)

Put $n = n-3$ in (I)

$T(n-3) = T(n-6) + (n-3)^2$ put in (I)

$T(n) = T(n-6) + (n-3)^2 + n^2$ — (II)

Put $(n-6)$ in (I)

$T(n-6) = T(n-9) + (n-6)^2$ putting in (II)

$T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n^2$

$\vdots$

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

$T(1) = 0$

$n - 3k = 1$

$n - 1 = 3k$

$\frac{n-1}{3} = k$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \ldots + (n-k)^2$

$T(n) = n^2 + n^2 + n^2 \ldots$ n times

$\Rightarrow n^3$

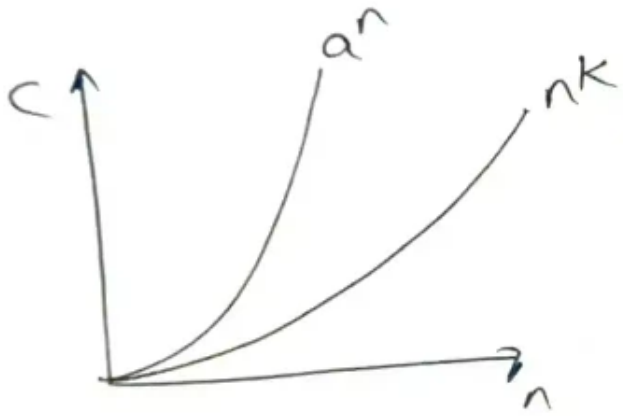
$T(n) = O(n^3)$

(9)

$\sum_{i=1}^{n} \sum_{j=1}^{n} 1 \Rightarrow \sum_{i=1}^{n} \log n \Rightarrow n \log n$

$(j += i)$

(10)



relationship holds for all values of c and $n_0 > 0$.

(11)

```
void fun(int n){
  int j=1, i=0;
  while(i<n){
    i = i+j;
    j++;
  }
}
```

$i = 0, 1, 3, 6, 10, 15 \ldots n$

$j = 1, 2, 3 \ldots \ldots$

$i_j = i_{j-1} + j$

j	i
1	0
2	0+1
3	0+1+2
4	0+1+2+3
k	0+1+2+3 ... +k

$0 + 1 + 2 + 3 \ldots k = n$

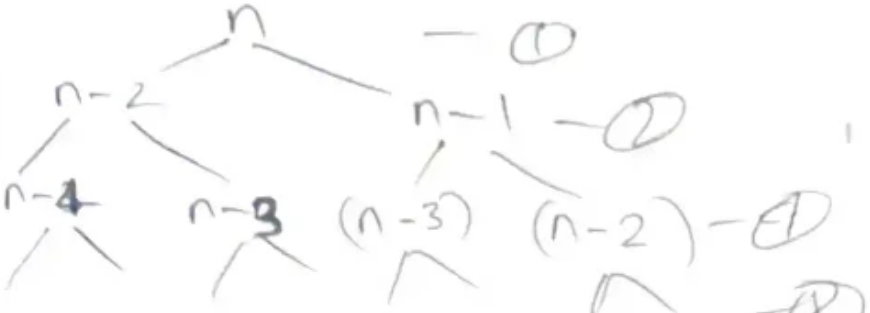
$0 + \frac{k(k+1)}{2}$

$k^2 = n$

$k = \sqrt{n}$

$T(n) = O(\sqrt{n})$ Ans

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



$T = 1 + 2 + 4 + 8 \ldots 2^n$

G.P

$a = 1$

$r = 2/1 = 2$

$S = a\left(\frac{r^n - 1}{r - 1}\right) = 1\left(\frac{2^n - 1}{2 - 1}\right)$

$\Rightarrow 2^n - 1 \quad T(n) = O(2^n)$

(13)

```
for(i=0; i<n; i*=2){
    for(j=0; j<n; j++)        n log n
    }}

for(i=0; i<n; i++){
    for(j=0; j<n; j++{        n^3
        for(k=0; k<n; k++{
    }}}

for(i=0; i*i<n; i++){         log(log n)
    }
```

(15) $\sum_{i=1}^{n} \sum_{j=1}^{n} 1$ $(j+=1)$ $\Rightarrow \sum_{i=1}^{n} \frac{n-1}{i} \Rightarrow (n-1) \sum_{i=1}^{n} \frac{1}{i}$

$\Rightarrow (n-1)\left[1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}\ldots\ldots+\frac{1}{n}\right]$

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

$T(n) = O(n\log n)$

(16) $\sum_{i=2}^{n} 1$ $i=pow(i,k)$ $\Rightarrow$ [illegible] $\Rightarrow$ [illegible]

$n = 2^{k^k}$

$\log n = k^k \log 2$

$\log n = k^k$

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

$\log_k \log_2 n = k$

$T(n) = (\log_k \log_2(n))$

(18)

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

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

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

(19)

```
for(i=0 to n-1)
    if (arr[i] = key)
        return i // key found.
return -1
```

20

```
insertion sort(arr, n){
    for i = 2 to n
        key = arr[i]
        insert arr[i] in sorted sequence
        j = i - 1
        while j > 0 and arr[j] > key
            A[j+1] = arr[j]
            j = j - 1
        A[j+1] = key
}
insertion sort(arr, n){
    if(n == 1) return
    recursion (n-1 elements) insertion sort(arr, n
    insert arr[i] in sorted sequence
        arr[0...i-1]
}
```

21

Bubble sort	n^2
Selection	n^2
merge	nlogn
insertion	nlogn
Quick	nlogn
Heap	nlogn

(22)

	Inplace	Stable	online
Bubble	✓	✓	—
Selection	✓		
Insertion	✓	✓	✓
Merge	—	✓	—
Quick	—	—	—
Heap	✓	—	—

(23)

```
Binary Search( arr , int l , int r , int x){
    while (l < r){
        int m = (l+r)/2
        if arr[m] = x ;
            return m // key found
        if arr[m] < x
                l = m+1
        else    r = m-1
    }
```

$T(n) = O(\log n)$

Space = $O(1)$

```
int BinarySearch (int arr[], int l, int r, int x){
    if (l > r) return -1;
    int m = (l+r)/2
    if (arr[m] == x) return m;
    if (arr[m] < x) BinarySearch(arr, m+1
                                 r, x)
    else BinarySearch (arr, l, m-1, x)
}
```

$T(n) = O(\log n)$

Space = $O(1)$ Except recursion stack.

(24) $T(n) = T(n/2) + 1$

$T(n)$	:	1 2 4 8 . . .
$T(n/2)$	2	$T(n) = \log n$
$T(n/1)$	4	