Assignment

Name - Tanishka Rana   Rollno - 2015398

Section   F

Subject   - DAA

As1. Asymptotic notations are the mathematical notations used to describe running time of an algorithm when the input tends towards a particular limiting value. Therefore notations are used to determine the running time of an algo and how it grows with the amount of input.

5 types of asymptotic notations

① Big Oh(O) notation.

$$O(g(n)) = \{ f(n) : \text{there exists positive}$$

~~constant~~ ~~and~~ ~~no~~ constant condition no such that

$$0 \leq f(n) \leq g(n) \; \forall \; n \geq n_0 \}$$

② Small oh (o) notation : We
denote o notation to denote an
upper bound that is not
asymptotically tight.

$$O(g(n)) = \{ f(n) : \text{for any} $$
$$\text{positive constant } c > 0$$
$$\text{there exist a constant} $$
$$n_0 > 0 \text{ such that} $$
$$0 \leq f(n) < g(n) \, \forall \, n \geq,$$

③ Big Omega (Ω) Notation.

$$\Omega(g(n)) = \{ f(n) : \text{for any positive} $$
$$\text{constant } c \text{ and } n_0)$$
$$0 \leq g(n) \leq f(n) \, \forall \, n \geq n_0$$

④ Small omega (ω) notation

⑤ Theta notation (θ)

Q2. $O(\log n)$

Q3. $T(n) = \begin{cases} 3(T(n-1)) & n > 0 \\ 1 & n \leq 0 \end{cases}$

By using back substitution.

$$T(n) = 3T(n-1) \quad —①$$
$$T(n-1) = 3T(n-2) \quad —②$$
$$T(n-2) = 3T(n-3) \quad —③$$

Put ② & ③ in ①

$$T(n) = 3 \cdot 3 \cdot 3\, T(n-3)$$
$$T(n) = 3^k\, T(n-k)$$

let $k = n$

$$T(n) = 3^n\, T(n-n)$$
$$= 3^n$$
$$= O(3^n)$$

(Q4) $T(n) = \begin{cases} 2(T(n-1)) - 1 & n > 0 \\ 1 & n < 0 \end{cases}$

using back substitution.

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

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

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

$T(n) = 2 \cdot 2T(n-2) - 2 - 1$

$T(n) = 2 \cdot 2 \cdot 2T(n-3) - 4 - 2 - 1$

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} \cdots 1$

let $k = n$

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

$- [1 + 2 + 4 + \cdots + 2^{k-1}]$

$= 2^n T(1) - [1(2^{k-1} - 1)]$

$= 2^n - 2^{n-1} - 1$

$= O(2^n)$

α5. $O(n)$

α6. $O(\sqrt{n})$

α7. $O(n \log n \log n)$

α6. The recurrence relation is

$$T(n) = T(n-3) + n^2 \qquad n > 1 \quad —①$$
$$n <= 1$$

By back substitution,

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

Put back in eq ①

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

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

let $3k = n$
$$k = \frac{n}{3}$$

②

$$T(n) = T\left(n - 3 \times \frac{n}{3}\right) + \underbrace{n^2 + (n-3)^2 + \cdots}$$
$$+ \underbrace{\left(n - 3\left(\frac{n}{3} - 1\right)\right)^2}$$

total of k terms

$$T(n) = T(1) + n^2 + (n-3)^2 + (n-6)^2 +$$
$$\cdots + (n - n + 3)^2$$

$$T(n) = 1 + \left(n^2 + n^2 + n + \cdots\right)$$
$$+ \underbrace{\left(xn + yn + zn +\right)}$$

can be ignored

$$T(n) = 1 + kn^2 + \cdots$$
$$T(n) = 1 + \frac{n^3}{3}$$
$$= O(n^3).$$

(Q9) Sum loop runs as

$$n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n}$$

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

This sum will converge to
$\log n$.

Hence $n(\log n)$

$\Rightarrow O(n\log n)$

(Q10) $f(n) = n^{k}$; $k >= 1$
$g(n) = a^{n}$. $a > 1$

exponential function grow faster
than polynomial functions hence.

$$O(n^{k}) < O(a^{n})$$

for values of $k \geq n$ and $a \geq y$.
Calculate x & y
let $k = 2$ and $a = 2$ as well
$f(n) = n^{2}$, $g(n) = 2^{n}$.
take log on both sides

$$\log(f(n)) = 2\log_2(n)$$

$$\log(g(n)) = n\log_2 2$$

$$0(\log n) < 0(n)$$

Hence for $k \geq 2$ and $a \geq 2$ the condition sta to satisfies.

Q11 $0(\sqrt{n})$, because the values of n' go as follows: $1, 3, 6, 10, 15, 2$

Also we know that

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

for the sum of series

$$1 + 2 + 3 + 4 \dots$$

So the series $1, 3, 6, 10, 15$ will stop when a_n

becomes equal to a greater than n

$$\frac{n(n+1)}{2} = n_0$$

$$n = \sqrt{n_0}$$

Q.12) $T(n) = \begin{cases} T(n-1) + T(n-2) + 1 & n \geq 2 \\ 1 & 0 \leq n < 2 \end{cases}$

assume time taken by $T(n-2)$
$$\simeq T(n-1)$$

So $T(n) = 2T(n-2) + c$ [c is a constant]

solving we get :

$T(n) = 2.2.2T(n-2.3) + 3c + 2c + 1c$

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

$$n - 2k = 0 \qquad k = \frac{n}{2}$$

$$T(n) = 2^{n/2} T(0) + (2^{n/2} - 1)c$$

$$T(n) = O(2^{n/2}) \simeq O(2^n)$$

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

we can assume $T(n/2) \geqslant T(n/4)$

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

using Masters theorem

$a = 2, \quad b = 2$

$\log_b a = \log_2 2 = 1$

$f(n) = cn^2$

$n^k = n^2$

$k = 2$

$\log_b a < k$

$1 < 2$

$+ cn^2$

)

n

Complexity $= O(n^k)$
$= O(n^2)$

(15) Inner loop will run $1/i$ times.

$= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n}$

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

$= n \log n$

$= O(n \log n)$

(Q18) (a) $100 < \log(\log(n)) < \log(n)$

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

$< 2^n < 2^{2n} < 4^n < n!$

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

$< \log(n) < \log(2n) < 2 \log(n)$

$< \log(n!) < n \log(n) < n^2 < (2^n)2$

$< n!$

© $96 < \log 8(n) < \log_2(n) < n \log 6n$

$< n \log_2(n) < \log(n!)$

$< 5n < 8n^2 < 7n^3 < 8\, 2^n < n!$

Q19

```
for (int i=0; i<n; i++)
{
    if (arr[i] is equal to key)
        print index and break
    else
        continues
}.
```

(Q21) Quicksort : $O(n \log n)$

Merge sort : $O(n \log n)$

Bubble sort : $O(n^2)$

Selection sort : $O(n^2)$

insertion sort : $O(n^2)$

(22) Inplace : Bubblesort, Selection sort, Quicksort, Insertionsort

Stable : Bubblesort, Insertion sort,

Mergesort.

Online · Insertion sort

(24) Recurrence relation for Binary search :

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