ENIGCS 301110 Dhananjay Porwal

# Design & Analysis of Algorithms

## Assignment I

Q1.

Ans a) Write an algorithm to produce first 15 no.s of the series 1, 1, 2, 3, 5, 8, 13, 21 ....

Approach - from the question we can clearly see that the sequence is fibonacci series i.e. sum of two previous elements determines the next element. Therefore, we need to print first 15 element of fibonacci series.

Pseudo Code -

1. Start
2. Declare & initialise a=1, b=1, i=2, n=15, c=0
3. Print a
4. Print b
5. Repeat until i <= n:

```
c = a + b
print(c)
a = b
b = c
```

$i = i + 1$

6. Stop

Analysis - Since there is no user input, the execution doesn't depend on input size, therefore running time complexity = O(1).

b) Write an algorithm raise any no. to 3rd power.

Approach - Since any number raised to 3rd power is same as multiplication of a number to itself thrice, therefore the running time complexity of this algo. is constant. Complexity = O(1)

Pseudo Code -

1. Start
2. Declare num, result
3. Read num from user input
4. result = num * num * num
5. print (result)
6. Stop.

Analysis - Here, all the steps require constant time for execution therefore this solution results in constant time comp.

c) Write an algo to raise any number to any power.

Approach - The problem requires a result of any number n, raised to the power k. Therefore a solution can be produce a result which is equivalent to multiplication of n to itself for k times.

Pseudo Code -

1. Start
2. Declare num, power, Result
3. Read num, power from user Input
4. Result = 1
5. Repeat until power >= 0:
   result = result * num
   power -= 1
6. Print (num)
7. Stop

Analysis - Since the execution of this program depend upon power (k) linearly, therefore the solution has linear time complexity = O(k)

d) Write an algo. to compute no. of days before Christmas

Approach - We need to find no. of days before Christmas We have to calculate no. of days after 1st Jan for the given date & for ~~Christ~~ Christmas & by sub. of these two numbers, we get the solution.

Pseudo Code -

1. Start
2. Declare givend, givenm, isleap, givend, xmasd
3. Read isleap
4. Read givend, givenm
5. Declare & initialise array, months = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
6. Check isleap == True:
   months[1] = 29
7. givend = Sum(months[givenm - 2]) + givend
8. xmasd = 25 + months[10]
9. Print (xmasd - givend)
10. Stop.

Analysis - Since this algorithm finds sum of month days before given month which can never exceed 12, therefore this algo. has constant time complexity.

e) Algo. to determine type of parallelogram.

Approach - We know that when the sides are given, we can determine a parallelogram as a square if all sides are equal & as a rectangle if 2 pairs of sides are equal.

Pseudo code -

1. Start
2. Declare $l_1, l_2, b_1, b_2$
3. Read $l_1, l_2, b_1, b_2$ from user input
4. Check if $l_1 \neq b_1$ & $l_2 \neq b_2$:
   print ("Rectangle")
   else:
   print ("Square")
5. Stop

Analysis -

Since we are only comparing the length of sides of the parallelogram, therefore this algo. takes constant time for execution. Hence, the time complexity = O(1)

1) Determine the type of triangle by given sides.

Approach - from the input we can get the length of sides of the triangle & we know that if all the sides of a triangle are equal then it is equilateral triangle, if two sides are equal then it is isosceles triangle & if all the sides are unequal then it is scalene triangle.

Pseudo Code -

1. Start
2. Declare $S_1, S_2, S_3$
3. Read $S_1, S_2, S_3$ from user input
4. Check if $S_1 == S_2$ && $S_2 == S_3$:
   print ("Equilateral triangle")
   else if $S_1 == S_2$ or $S_1 == S_3$ or $S_2 == S_3$:
   print ("Isosceles triangle")
   else:
   print ("Scalene triangle")
5. Stop.

Analysis - Since the algo is performing comparision only, therefore it is a constant complexity algo. time com. = O(1)

9) Algo. to find remainder of divisor of 2 nos.

Approach - Since the remainder is the left-off no. remainder after performing division of 2 no. we can find out the result by sub. the product of quotient & divisor from the dividend to find the remainder.

Pseudo code -

1. Start
2. Declare num1, num2, rem, quo
3. Read num1, num2, from user input
4. quo = num1 / num2
5. rem = num1 - (num2 × quo)
6. print (rem)
7. Stop.

Analysis -

The algo. only computes & results into mathematical operations, therefore its independent of input size. Time complexity = O(1)

Q2

Ans To prove this, we have to show that there exists constants $c_1, c_2, n_0 > 0$ such that $\forall\ n \geq n_0$

$$0 \leq c_1 (f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2 (f(n) + g(n))$$

As the fun^n are asymptotically non-ve, we can assume that for some $n_0 \geq 0$, $f(n) \geq 0$ & $g(n) \geq 0$ for $n \geq n_0$.

$$f(n) + g(n) \geq \max(f(n), g(n))$$

Also note that $f(n) \leq \max(f(n), g(n))$ &
$g(n) \leq \max(f(n), g(n))$

$$f(n) + g(n) \leq 2 \max(f(n), g(n))$$

$$\Rightarrow \frac{1}{2}(f(n) + g(n)) \leq \max(f(n), g(n))$$

So, $0 \leq \frac{1}{2}(f(n) + g(n)) \leq \max(f(n), g(n)) \leq (f(n) + g(n))\ \forall\ n \geq$

So, max $(f(n), g(n)) = f(n) + g(n)$ because there exists $c_1 = 0.5$ & $c_2 = 1$

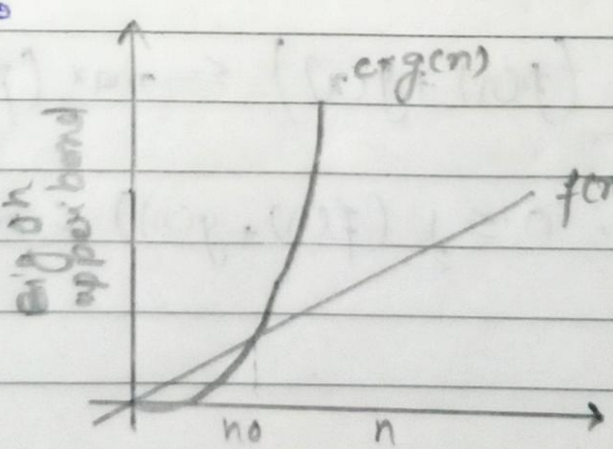
Qs

Ans Asymptotic notation describes the algo. efficiency & performance in a meaningful way. It describes the behaviour of the time or space complexity for large instance characteristics.

The asymptotic running time of an algo. is defined in terms of fun.

a). Big Oh notation (O) - It is asymptotic upper bound.

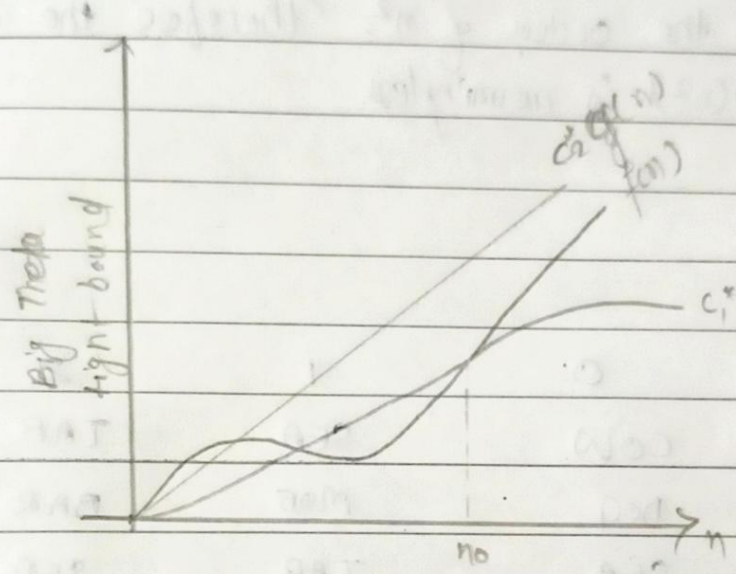
$f(n) = O(g(n))$ exist if, $f(n) \leq c^* g(n)$ where c is a ~~pos~~ +ve constant for all $n \geq n_0$



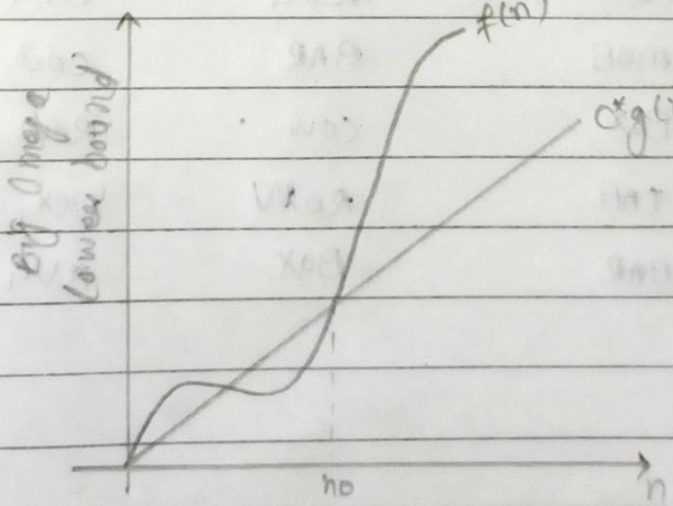
b) Big Theta notation ($\Theta$) - Tight bound

$f(n) = \Theta(g(n))$ if, $c_1 g(n) \leq f(n) \leq c_2 g(n)$

for all $n \geq n_0$




c) Big Omega notation ($\Omega$) - Lower bound



Let $T(n)$ be running time for algo. A $f(n) = O(n^2)$
The statement says $T(n)$ is at least $O(n^2)$.
While $O(n^2)$ describes that the fun$^n$ is upper bound to order of $n^2$ which means its complexity is atmost $n^2$. It lies anywhere less than or equal to the order of $n^2$. Therefore the statement atleast $O(n^2)$ is meaningless.

Q.4

Ans

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| COW | SEA | TAB | BAR |
| DOG | MOB | BAR | BOX |
| SEA | TAB | SEA | COW |
| RUG | DOG | MOB | DOG |
| ROW | RUG | DOG | MOB |
| MOB | BAR | COW | ROW |
| BOX | COW | ROW | RUG |
| TAB | ROW | BOX | SEA |
| BAR | BOX | RUG | TAB |

Q5

Ans Given: $T(n) = 7T(n/2) + n^2$

wkt,

$$T(n) = \Theta(n^{\log_4 49})$$

Q.

$$T'(n) = aT(n/4) + n^2$$

$a = \alpha$, $b = 4$, $f(n) = n^2$

$n^{\log_b a} = n^{\log_4 \alpha}$

$f(n) = O(n^{\log_e a})$ where $e > 0$

for $a > 16$ : $T'(n) = \Theta(n^{\log_4 a})$

$\therefore \log_4 a < \log_4 49 \Rightarrow a < 49$

Consequently the value of **a** so that x is asymptotically fastee then A will be value of 48