9-6-2024

CSA 0666 - Design And Analysis of Algorithm

P. Oviya
192324086

1. If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then $t_1(n)+t_2(n) \in O(\max\{g_1(n), g_2(n)\})$. Prove the assertions.

$f_1(n) \le c_1 g_1(n)$ for all $n \ge n_0$

$f_2(n) \le c_2 g_2(n)$ for all $n \ge n_0$

Adding

$f_1(n) + f_2(n) \le c_1 g_1(n) + c_2 g_2(n)$

Since

$\max\{g_1(n), g_2(n)\} \ge g_1(n)$

$\max\{g_1(n), g_2(n)\} \ge g_2(n)$

$f_1(n) + f_2(n) \le c_1 \max\{g_1(n), g_2(n)\} + c_2 \max\{g_1(n), g_2(n)\}$

$\le (c_1+c_2) \max\{g_1(n), g_2(n)\}$

let $c = c_1 + c_2$

$f_1(n) + f_2(n) \le c\max\{g_1(n), g(2)\}$ for all $n \ge n_0$

$\therefore f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$

2 Find the time complexity of the bellow recurrence equation

$$T(n) = \begin{cases} 2T(n/2) + 1 & \text{if } n > 1 \\ 1 & \text{otherwise} \end{cases}$$

$T(n) = aT(n/b) + f(n)$

if $f(n) = O(n\log^a_b - c)$

then $T(n) = \Theta(n \log^a_b)$

if $f(n) = \Theta(n \log^a_b \ \log^k n)$

then $T(n) = \Theta(n \log_b^a \log^{k+1} n)$

if $f(n) = \Omega(n \log_b^{a} + c)$

then $T(n) = \Theta(f(n))$

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

$a = 2$

$b = 2$ $k = 1$, $p = 1$

$\log_a^b = \log_2^2 = 1$

$\log_a^b = k$ $p \geq -1$ $\Theta(n^k \log^{p+1} n)$

4 $T(n) = \begin{cases} 2T(n-1) & \text{if } n > 0 \\ 1 & \text{otherwise} \end{cases}$

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

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

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

$T(n) = 2^2[2T(n-3)]$

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

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

$n - k = 0$, $n = k$.

$T(0) = 1$

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

5 Big O Notation: ST $f(n) = n^2 + 3n + 5$ is $O(n^2)$

$f(n) = n^2 + 3n + 5$

$f(n) \leq c \cdot n^2$

for all $n > n_0$

$f(n) = n^2 + 3n + 5$

$= n^2 + 3n + 5$

$f(n) = n^2 + 3n + 5 \leq c\, n^2$

$n^2 + 3n + 5 \leq c \cdot n^2$

$3n + 5 \leq c \cdot n^2$

$\therefore$ where n is close to 0, $3n + 5 \leq c \cdot n^2$ can be (-ve)

$f(n) = O(n^2)$

6 Big Omega Notation : Prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$

$g(n) = n^3 + 2n^2 + 4n$

$g(n) \geq c \cdot n^3$

$g(n) = n^3 + 2n^2 + 4n$

$= n^2(n+2) + 4n$

$g(n) \leq n^3$

$g(n) = n^2(n+2) + 4n \geq c \cdot n^3$

$n^2(n+2) + 4n \geq c \cdot n^3$

$n^2(n+2) + 4n - c \cdot n^3 \geq 0$

$n^2(n+2) + 4n - c \cdot n^3 \geq 0$

$\therefore$ This inequality is not alway true when n is close to 0. $n^2(n+2)4n - c \cdot n^3$ can be (-ve)

$\therefore g(n) \neq \Omega(n^3)$

7 Big theta notation : Determine whether $h(n) = 4n^2 + 3n$ is $\Theta(n^2)$ or not

$h(n) = 4n^2 + 3n$

first, we need to find the constant c such that $h(n) \geq c \cdot n^2$ for large enough n.

$h(n) = 4n^2 + 3n$

$= n^2(4 + 3/n)$

$h(n) = n^2(4 + 3/n) \geq c \cdot n^2$

$\Rightarrow \cancel{n^2}\left(4+\frac{3}{2}\right) \geq c\cdot\cancel{n^2}$

$\Rightarrow 4 + 3/n \geq c$

This in equality to hold for all n, we need

$4 + 3/n \geq c$ for all n.

This inequality is not always true when n is close 0. $4+3/n$ can be less than c.

$\therefore$ We can't find a constant c such that

$h(n) \geq c\cdot n^2$

$\therefore h(n) \neq \Theta(n^2)$

8 Let $f(n) = n^3 - 2n^2 + n$ and $g(n) - n^2$ Show whether $f(n) = \Omega(g(n))$ is true or false and justify your answer

$f(n) = n^3 - 2n^2 + n$

$g(n) = n^2$

$f(n) \geq c\cdot g(n)$

$f(n) = n^3 - 2n^2 + n$

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

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

Compare $f(n)$ & $g(n)$:

$f(n) = n^2 \quad (n-2+1/n) \geq c\cdot n^2$.

$n^2(n-2+1/n) \geq c\cdot n^2$

$n^2(n-2+1/n) + c\cdot n^2 \geq 0$

$n^2(n-2+\frac{1}{n}+c) \geq 0.$

$n - 2 + 1/n + C \geq 0$

This inequality is not always true for example, when n is close to 2. $n - 2 + 1/n + C$ can be neg

$\therefore f(n) \neq \Omega(g(n))$

9 Determine whether $h(n) = n\log + n$ is in $\Theta(n\log n)$ prove a rigorous proof for your conclusion

$h(n) = n\log n + n$

$c_1 \cdot n\log n \leq n\log n + n \leq c_2 \cdot n\log n$

Upper bond :

$n\log n + n \leq c_2 \cdot n\log n$

$n\log n + n \leq n\log n + n\log n = 2n\log n$

$c_2 = 2$

$n\log n + n \leq 2n\log n$

Lower bond :

$c_1 \cdot n\log n \leq n\log n + n$

$c_1 \cdot n\log n \leq n\log n + n$

divide both sides by (n)

$c_1 \cdot \log n \leq \log n + 1$

$1/2 \log n \leq \log n$

$c_1 \cdot n\log n \leq n\log n + n \leq c_2 \cdot n\log n$

$\therefore h(n) = n\log n + n \in \Theta(n\log n)$

10 Solve the following recurrence relations and find the order of growth for solutions.

$T(n) = 4T(n/2) + n^2, \quad T(1) = 1$

$T(n) = 4T(n/2) + n^2 \quad T(1) = 1$

By master's theorem

$a = 4 \quad k = 2$

$b = 2 \quad f(n) = n^2 \quad P = 0$

$\log_b^a = \log_2^4 = \log_{2^1}^{2^2} = \log_1^2 = 2$

$\therefore \log_b^a = k$

$P > -1$, so,

$= \Theta(n^k \log_n^{P+1})$

$= \Theta(n^2 \log^1 n)$

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

11 Given an array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers, find the maximum and minimum product that can be obtained by multiplying two integers from the array.

[4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9]

Maximum product

2 largest no's : 11, 10

2 smallest (-ve no's) : -9, -8

products

$11 \times 10 = 110$

$-9 \times -8 = +72$

Max product = 110

Min product

$11 \times -9 = -99$

$10 \times -9 = -90$

$\therefore$ Min product = -99

2 Demonstrate Binary Search method to search key = 23, form the array arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}

arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

key = 23.

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	23	38	56	72	91

$M = \frac{l+h}{2} = \frac{0+9}{2} = 4.5 = 5.$

0	1	2	3	4	5	6	7	8	9
2	5	8	12	16	$\boxed{23}$	38	56	72	91
					MID				

arr[mid] = 23

arr[mid] = key

23 = 23

∴ key is found.

13 Apply merge sort and order of list of 8 elements. Data d = {45, 67, -12, 5, 22, 30, 50, 20}. Set up a recurrence relation for the number of key comparisons made by merge sort.

	0	1	2	3	4	5	6	7
d =	45	67	-12	5	22	30	50	20

$M = \frac{0+7}{2} = 4.$

0	1	2	3	4	5	6	7
45	67	-12	5	22 |	30	50	20.

$M = \frac{0+4}{2} = 2.$

0	1	2	3	4	5	6	7
45	67	-12 |	5	22 |	30	50 |	20

$M = \frac{0+2}{2} = 1$

0	1	2	3	4	5	6	7
45	67	-12	5	22	30	50	20

$M = \frac{0+1}{2} = 0.5$

0	1	2	3	4	5	6	7
45	67	-12	5	22	30	50	20

| 45 | 67 | | -12 | 5 | | 22 | | 30 | 50 | | 20 |

| -12 | 5 | 45 | 67 | | 22 | | 20 | 30 | 50 |

| -12 | 5 | 22 | 45 | 67 | | 20 | 30 | 50 |

| -12 | 5 | 20 | 22 | 30 | 45 | 50 | 67 |

Sorted

recurrence relation

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

$a = 2$ $\quad$ $k = 1$

$b = 2$ $\quad$ $p = 1$

$\log_b^a = \log_2^2 = 1$

$\Rightarrow \log_b^a = k.$

$\therefore \Theta(n^k \log_n^{p+1})$

$\Theta(n^1 \log^2_n)$

$\therefore \Theta(n \log n).$

4 Find the no of times to perform Swapping for selection Sort. Also estimate the time Complexity for the order of notation set S (12, 7, 5, -2, 18, 6, 13, 4)

S = 12, 7, 5, -2, 18, 6, 13, 4.

1] 12, 7 5 -2 18 6 13 4

Start Min

2] -2 7 5 12 18 6 13 14

Start min

3] -2 5 7 12 18 6 13 14.

Start Min.

4] -2 5 6 12 18 7 13 14

Start. min

5] -2 5 6 7 18 12 13 14

Start min

6] -2 5 6 7 12 18 13 14

Start min

7] -2 5 6 7 12 13 18 14

Start Min

8]

-2	5	6	7	12	13	14	18

⇒ Sorted

time complexity. Best $O(n^2)$

Space complexity ~~Avg~~ - $O(1)$

Total no of Swaps 6

15 Find the index of the target value 10 using binary search from the following list of elements
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

Given

0	1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18	20

$M = \frac{l+h}{2} = \frac{0+9}{2} = 4.5 \simeq 5$ (or) 4.

target = 10.

0	1	2	3	4	5	6	7	8	9.
2	4	6	8	$\boxed{10}$ Mid.	12	14	16	18	20

target = 10
a[mid] = targed

10 = 10

The target element is found.

16 Sort the following elements using merge sort divide & conquer strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyze complexity of the algorithm.

0	1	2	3	4	5	6	7	8	9	10	11
38	27	43	3	9	82	10	15	88	52	60	5.

$M = \frac{l+h}{2} = \frac{0+11}{2} = 5.5 \simeq 6$

0	1	2	3	4	5	6	7	8	9	10	11
38	27	43	3	9	82	10	15	88	52	60	5

$M = \frac{l+h}{2} = \frac{0+6}{2} = 3$

$M = \frac{l+h}{2} = \frac{7+11}{2} = 9$

0 1 2 3 | 4 5 6 | 7 8 9 | 10 11

38 27 43 3 | 9 82 10 | 15 88 52 | 60 5.

$M = \frac{l+h}{2} = \frac{0+3}{2} = 2$ $\quad M = \frac{l+h}{2} = \frac{4+6}{2}$ $\quad M = \frac{7+9}{2} = 8$ $\quad M = \frac{21}{2} = 10.$

0 1 2 | 3 | 4 5 | 6 | 7 8 | 9 | 10 | 11

(5 = 5)

38 27 43 | 3 | 9 82 | 10 | 15 88 | 52 | 60 | 5

$M = \frac{0+2}{2} = 1$

38 27 | 43 | 3 | 9 | 82 | 10 | 15 | 88 | 52 | 60 | 5.

$M = 0$

38 | 27 | 43 | 3 | 9 | 82 | 10 | 15 | 88 | 52 | 60 | 5.

| 27 | 38 | | 43 | | 3 | | 9 | 82 | 10 | | 15 | 88 | | 52 | | 5 | 60 |

| 27 | 38 | 43 | | 3 | | 9 | 10 | 82 | | 15 | 52 | 88 | | 5 | 60 |

| 3 | 27 | 38 | 43 | | 9 | 10 | 82 | | 5 | 15 | 52 | 60 | 88 |

| 3 | 9 | 27 | 38 | 43 | 82 | | 5 | 15 | 52 | 60 | 88 |

| 3 | 9 | 27 | 38 | 43 | 82 | 5 | 15 | 52 | 60 | 88 | ⇒

3 5 9 15 27 38 43 52 60 82 88 ⇒ Sorted

time complexity ⇒ $O(n^2)$

17. Sort the array 64 , 25 , 34 , 12 , 22 , 11 , 90 using bubble sort.
What is the time complexity in the best worst & average case?

Iteration 1

64 34 25 12 22 11 90
i j

34 64 25 12 32 11 90
i j

34 25 64 12 22 11 90
i j

34 25 12 64 22 11 90
i j

34 25 12 22 64 11 90.
i j

34 25 12 22 64 11 90
i j

34 25 12 22 11 64 90
i j

Iteration 2.

34 25 12 22 11 64 90
i j

25 34 12 22 11 64 90
i j

25 12 34 22 11 64 90
i j

25 12 22 34 11 64 90
i j

25 12 22 11 34 64 90
i j

iteration -3.

25 12 22 11 34 64 90
i j

12 25 22 11 34 64 90

12 22 25 11 34 64 90

i j

12 22 11 25 34 64 90

i j

12 22 11 25 34 64 90

i j

12 22 11 25 34 64 90

i j

Iteration 4

12 22 11 25 34 64 90

i j

12 22 11 25 34 64 90

i j

12 11 22 25 34 64 90

i j

12 11 22 25 34 64 90

i j

12 11 22 25 34 64 90

i j

12 11 22 25 34 64 90

i j

Iteration 5

12 11 22 25 34 64 90

i j

11 12 22 25 34 64 90

i j

11 12 22 25 34 64 90

i j

11 12 22 25 34 64 90

i j

11 12 22 25 34 64 90

i j

11 12 22 25 34 64 90

i j

11 12 22 25 34 64 90

Sorted

time complexity

Best – O(n)

Avg – O(n2)

Worst – O(n²)

18 Sort the array 64, 25, 12, 11 using selection sort. What is the time complexity of selection sort in best, worst, average case?

0	1	2	3	4
$\boxed{64}$	25	12	22	$\boxed{11}$
Start				Min

11 | $\boxed{25}$ $\boxed{12}$ 22 64

Sorted | Start Min

11 12 | $\boxed{25}$ $\boxed{22}$ 64

Sorted | Start Min

11 12 22 | $\boxed{25}$ 64. So, no swapping.

Start Min

11 12 22 25 64 ⇒ Sorted.

time complexity

Best case = O(n2)

Avg case : O(n2)

Worst : O(n²)

Sort the following elements using insertion sort using bruteforce approach strategy. [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyze complexity of algorithm.

	38	27	43	3	9	82	10	15	88	52	60	5
	i	j										
1)	27	38	43	3	9	82	10	15	88	52	60	5.
2)	27	38	43	3	9	82	10	15	88	52	60	5.
3)	27	38	3	43	9	82	10	15	88	52	60	5.
4)	3	27	38	43	9	82	10	15	88	52	60	5
5)	3	9	27	38	43	82	10	15	88	52	60	5.
6)	3	9	10	27	38	43	82	15	88	52	60	5.
7)	3	9	10	15	27	38	43	82	88	52	60	5.
8)	3	9	10	15	27	38	43	82	88	52	60	5.
9)	3	9	10	15	27	38	43	52	82	88	60	5
10)	3	9	10	15	27	38	43	52	82	60	88	5.
11)	3	9	10	15	27	38	43	52	60	82	88	5
12)	3	9	10,	15	27	38	43	52	60	82	88	5
13)	3	5	9	10	15,	27	38	43	52	60	82	88.

Sorted //

Time complexity =
Best = $O(n)$ this occurs when the array is already sorted. The inner loop will run only once
Avg : $O(n^2)$ - The list is randomly ordered.
Worst - $O(n^2)$ if the list is in reverse.
Space complexity = $O(1)$ - insertion sort.

20 Given an array [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers, sort the following elements using insertion sort using brute force approach strategy analyze complexity of the algorithm.

arr = [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9].

4 -2 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9.

i j

swap.

-2 4 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9.

i j

shift

-2 4 5 3 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 -9

i j

swap.

-2 4 3 5 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 9

i j

-2 3 4 5 10 -5 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 9

i j

-2 3 4 5 -5 10 2 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 9

i j swap

-5 -2 3 4 5 2 10 8 -3 6 7 -4 1 9 -1 0 -6 -8 11 9.

i p j

-5 -2 3 4 5 2 8 10 -3 6 7 -4 1 9 -1 0 6 -8 11 -9.

i j

-5 -2 2 3 4 5 8 10 -3 6 7 -4 1 9 -1 0 6 -8 11 -9.

swap.

-5 -3 -2 2 3 4 5 8 10 6 7 -4 1 9 -1 0 6 -8 11 -9.

i j

-5 -3 -2 2 3 4 5 8 6 10 7 -4 1 9 -1 0 6 -8 11 -9.

i j

-5 -3 -2 2 3 4 5 6 7 8 10 -4 1 9 -1 0 -6 -8 11 -9

i j

-4

-5 -3 -2 2 3 4 5 6 7 8 10 1 9 -1 0 -6 -8 11 -9

i j

-5 -3 -4 -2 1 2 3 4 5 6 7 8 10 9 -1 0 -6 -8 11 -9.

i j

-5 -3 -4 -2 1 2 3 4 5 6 7 8 9 10 -1 0 -6 -8 11 -9

i j

-5 -3 -4 -2 -1 1 2 3 4 5 6 7 8 9 10 0 -6 -8 11 -9

i j

-5 -3 -4 -2 -1 0 1 2 3 4 5 6 7 8 9 10 -6 -8 11 -9

-6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 -8 11 -9

-8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 -9

-9 -8 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11

Sorted

time complexity.

Best O(n) - This occurs when the array is already sorted. The inner loop will run only once for each element.

Average case : O(n2) = The list is randomly order

Worst case : O(n2) = If the list is in reverse order.

Scanned with OKEN Scanner