CSA0672- DAA

M. Shashanka
192311083.

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 that assertions.

Sol. The proof extends to orders of growth the following simple fact about four arbitrary real numbers a_1, b_1, a_2, b_2: if $a_2 \le b_1$ and $a_2 \le b_2$, then $a_1 + a_2 \le 2 \max\{b_1, b_2\}$.

Since $t_1(n) \in O(g_1(n))$, there exist some positive constant c_1 and some non negative integer n_1 such that

$$t_1(n) \le c_1 g(n) \text{ for all } n \ge n_1.$$

Similarly, since $t_2(n) \in O(g_2(n))$:

$$t_2(n) \le c_2 g_2(n) \text{ for all } n \ge n_2.$$

Let us denote $c_3 = \max\{c_1, c_2\}$ and consider $n \ge \max\{n_1, n_2\}$ so that we can use both inequalities. Adding them yields the following

$$\begin{aligned} t_1(n) + t_2(n) &\le c_1 g_1(n) + c_2 g_2(n) \\ &\le c_3 g_1(n) + c_3 g_2(n) \\ &\le c_3 [g_1(n) + g_2(n)] \\ &\le c_3 \, 2 \max\{g_1(n), g_2(n)\} \end{aligned}$$

Hence, $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n), g_2(n)\})$ the constants c and n_0 required by the definition O being $2c_3 = 2 \max\{c_1, c_2\}$ and $\max\{n_1, n_2\}$.

The property implies that the algorithm's overall efficiency will be determined by the part with a higher order of growth, i.e. its least efficient part

$\therefore$ $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g_2(n))$, then,

$t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$.

② Find the Time complexity of the below recurrence equation.

③ $T(n) = \begin{cases} 2T(n/2)+1 & \text{if } n>1 \\ 1 & \text{otherwise.} \end{cases}$

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

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

$T(n) = \begin{cases} 2T(n-1) \\ 1 \end{cases}$

$T(n) = aT(n/b) + f(n)$ [∵ from masters theorem]

Here, $a=2$, $b=2$, $f(n)=1$, $k=0$

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

∴ $\log_b^a > k$ then $T(n) = O(n^{\log_b^a})$

$= O(n^{\log_2^2}) = O(n)$ //.

$\Rightarrow T(n) = 2^n T(n-n)$ (∵ $T(0)=1$)

$= 2^n T(0)$

$T(n) = 2^n$.

∴ The time complexity is $T(n) = O(2^n)$

⑤ Big O notation show that $f(n) = n^2 + 3n + 5$ is $O(n^2)$

⇒ Given $f(n) = n^2 + 3n + 5$

A function $f(n)$ is $O(g(n))$ if there exist constant $c>0$ and n_0 such that for all $n > n_0$:

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

$n^2 + 3n + 5 \leq c \cdot n^2$ (∵ divide both sides with n^2)

$\Rightarrow 1 + 3/n + 5/n^2 \leq c$

$\Rightarrow$ Let $c=2$, then

$n = n_0$

⇒ 1 ≤ 3/n + 5/n² ≤ 2

⇒ 3/n + 5/n² ≤ 2 - 1

⇒ 3/n + 5/n² ≤ 1

⇒ 3/n + 5/n² ≤ 1/2 + 1/2

⇒ 3/n ≤ 1/2 & 5/n² ≤ 1/2

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

To solve this problem, we will use iteration method

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

$n = n-1 \Rightarrow -①$

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

$\Rightarrow 2T(n-2) - ②$

② in ①

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

$\Rightarrow 2^2 T(n-2) - ③$

$\rightarrow n = n-2 \Rightarrow -④$

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

$= 2T(n-3)$

④ in ③

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

$= 2^3 T[n-3] - ⑤$

continue the process

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

$n-k = 0$

$k = n$

$n \geq 6$ & $n^2 \geq 10$ $\boxed{\therefore n = 6}$

$n \geq \sqrt{10} \approx 3.16$

$1 + 3/n + 5/n^2 \leq 1 + 1/2 + 1/2 = 2$

Thus, for $\boxed{c = 2}$

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

$\therefore f(n) = n^2 + 3n + 5 \in O(n^2)$

Big Omega notation: prove that $g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$.

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

A function $g(n)$ is $\Omega(f(n))$ if there exist positive constants c and n_0 such that for all $n > n_0$:

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

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

$1 + 2/n + 4/n^2 \geq c$

$\boxed{\text{let } c = 1}$

$1 + 2/n + 4/n^2 \geq 1$

Thus, for $\boxed{c = 1}$

$n^3 + 2n^2 + 4n \geq 1 \cdot n^3$

$\therefore n^3 + 2n^2 + 4n \geq n^3$

$g(n) = n^3 + 2n^2 + 4n$ is $\Omega(n^3)$.

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

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

A function $f(n)$ is $\Theta(g(n))$ if there exist constants $c > 0$ and n_0 such that for all $n \geq n_0$:

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

$4n^2 + 3n \leq c \cdot n^2$ [Divide with n^2]

$4 + 3/n \leq c$

Let $c = 5$, then

$4 + 3/n \leq 5$

This is true for all $n \geq 1$. Therefore, we can take $c = 5$ and $n_0 = 1$.

Thus, $h(n) = 4n^2 + 3n$ is $O(n^2)$.

(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.

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

& $g(n) = n^2$

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

$n^3 - 2n^2 + n \geq c \cdot n^2$

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

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

Let $c = 1$, then

$n^3 - (2+1) n^2 + n \geq 0$

$n^3 - 3n^2 + n \geq 0$

$n^3 - 2n^2 + n \geq c \cdot n^2$

thus $c = 1$

$n^3 - 2n^2 + n \geq n^2$

$\therefore f(n) = n^3 - 2n^2 + n \in \Omega(n^2)$ //.

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

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

$f(n) \leq c \cdot \log n$

$n \log n + n \leq c \cdot n \log n$

Let $c = 2$, then.

$\log n+1 \leq 2 \log n$.

therefore, $h(n) = n\log n + n$ is $O(n\log n)$.

(10) Solve the following recurrence relations and find the order of growth for solutions. $T(n) = 4T(n/2) + n^2$ $T(1) = 1$.

A

$T(n) = 4T(n/2) + n^2$, $T(0) = 1$

By masters theorem.

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

Here $a = 4$, $b = 2$ | $f(n) = n^2$, $k = 2$ $\therefore \log_b^a = k$

$\log_b^a = \log_2^4 = 2 = k$

$p > -1$ then.

$\Rightarrow O(n^k \log_n^{p+1})$

$\Rightarrow O(n^2 \log n)$.

(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.

Given an array is

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

maximum product.

(1) product of two largest +ve numbers,

(2) product of two negative numbers.

minimum product

product of largest +ve and most -ve numbers

product of two Smallest -ve numbers.

(14)

Sorting the array :-

Sorted array = {-9, -8, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

Maximum product calculation :-

product of two largest +ve numbers = 10 x 11 = 110

product of two -ve numbers = -9 x -8 = 72

Thus, the maximum product is 110.

Minimum product calculation :-

① product of largest +ve & most -ve numbers -99.

② product of two smallest -ve numbers = 72

∴ max = 110, min = -99.

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

Given array[] :

{2, 5, 8, 12, 16, 23, 38, 56, 72, 91}

⇒

l				h
2	5	8	12	16
0	1	2	3	4

23	38	56	72	91
5	6	7	8	9

mid = $\frac{l+h}{2}$ = 0+9/2 ⇒ 5

key = 23 > arr[4] = 16

mid = $\frac{l+h}{2}$ = $\frac{5+9}{2}$ = 7

arr[7] = 56

key = 23 < arr[7] = 56

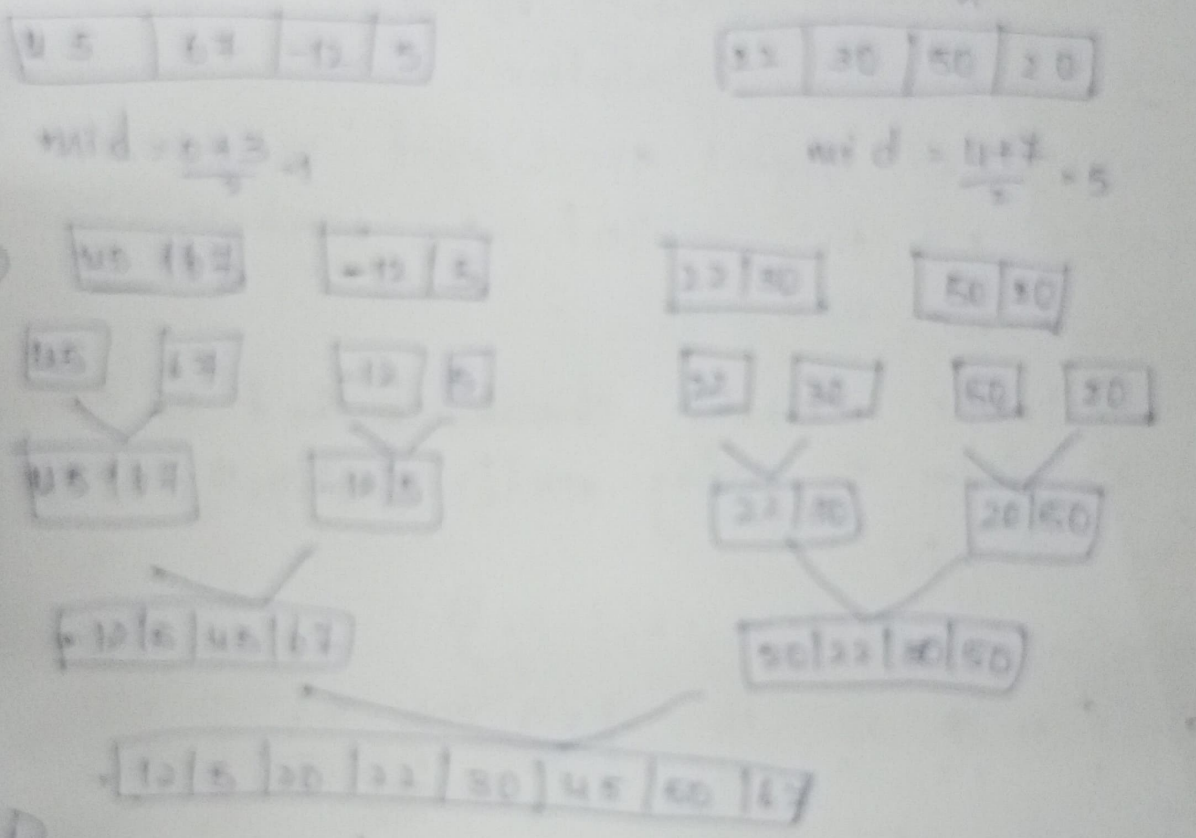
mid = $\frac{5+6}{2}$ = 5 ⇒ arr[5] = 23

key = 23 = arr[5] = 23

Thus key = 23 = arr[5] is found at index 5 in array using the Binary Search method.

Apply merge sort and order the list of Solomon data A = (45, 67, -12, 5, 33, 30, 50, 20) Set up a recurrence relation for the number of key comparisons made by mergesort.

Given A: (45, 67, -12, 5, 33, 30, 50, 20) $\frac{4+4}{2} = 4$



Recurrence relation:-

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

$T(n) = 2T(n/2) + \Theta(n)$

Here $a = 2$, $b = 2$ | $f(n) = \Theta(n)$

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

as $\log_b^a = n^1 = n$

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

∴ Time complexity is $\Theta(n \log n)$.

(14) Find the no. of times to perform swapping for selection sort. Also elements the time complexity for the order of notation set (12, 7, 5, -2, 18, 6, 13, 4)

Sol: $S = (12, 7, 5, -2, 18, 6, 13, 4)$

12	7	5	-2	18	6	13	4	unsorted

Sorted

=> | -2 | 4 | 5 | 12 | 18 | 6 | 13 | 7 |

=> | -2 | 4 | 5 | 12 | 6 | 18 | 13 | 7 |

=> | -2 | 4 | 5 | 6 | 18 | 12 | 13 | 7 |

=> | -2 | 4 | 5 | 6 | 7 | 12 | 13 | 18 |

Total swaps = 4 but as per theory, it should be $n-1 = 7$

Best case = $O(n^2)$

Average case = $O(n^2)$

Worst case = $O(n^2)$

(15) Find the index of the largest value to using binary search for the following list of elements [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]:

Given list of elements arr[] =

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

mid = l+h/2 => 0+9/2 = 4

arr(4) = 10

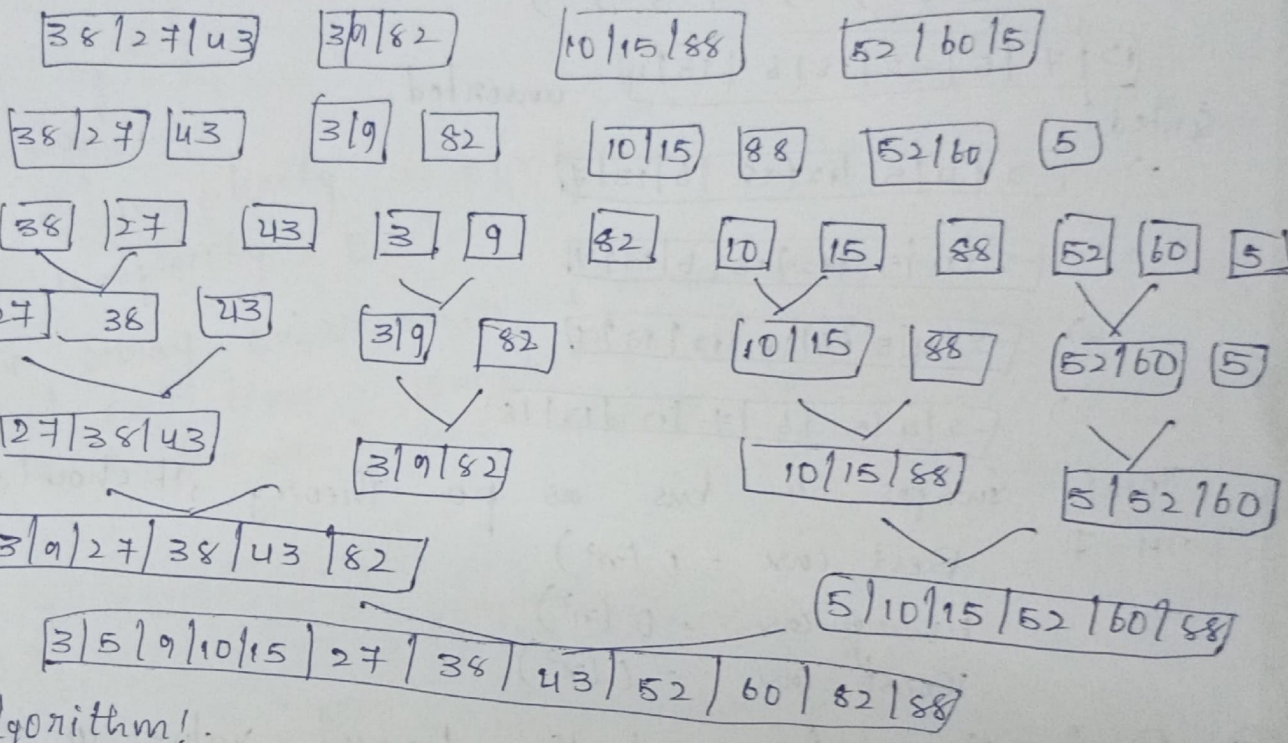
∴ The index of target value 10 in list is 4.

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

Given elements,

[38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5]

mid = l+h/2 => 0+5/2



Algorithm!.

```
mergesort (l, h)
{
  If [l<h)
  {
    mid = (l+h)/2 ;
  merge sort (l, mid)
  merge sort (mid+1, h)
  merge [l, mid, h)
}
```

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

$a = 2$, $k = 1$

$b = 2$

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

Pr-1 $O(n^k \log^{p+1} n)$

$= (n^1 \log^{1+1} n)$

$\Rightarrow n \log n^2$

∴ Time complexity $= O(n \log n)$

Space complexity $= O(n)$.

(4) Sort the array 64, 34, 25, 12, 22, 11, 90 using bubble sort. what is time complexity of selection. Sort in the best, worst and average cases?

Sol Given array

S-1 [64|34|25|12|22|11|90]

[34|64|25|12|22|11|90] ⇒ [34|25|64|12|22|11|90]

⇒ [34|25|12|64|22|11|90] ⇒ [34|25|12|22|64|11|90]

⇒ [34|25|12|22|11|64|90]

S-2 [34|25|12|22|11|64|90] ⇒ [25|34|12|22|11|64|90]

⇒ [25|12|34|22|11|64|90] ⇒ [25|12|22|34|11|64|90]

⇒ [25|12|22|11|34|64|90]

S-3

⇒ [25|12|22|11|34|64|90] ⇒ [12|25|22|11|34|64|90]

⇒ [12|22|25|11|34|64|90] ⇒ [12|22|11|25|34|64|90]

S-4

⇒ [12|22|11|25|34|64|90] ⇒ [12|11|22|25|34|64|90]

⇒ [12|11|22|25|34|64|90]

S-5 ⇒ [11|12|22|25|34|64|90]

Best case $= O(n)$

Avg case $= O(n^2)$

worst case $= O(n^2)$

(18) Sort the array 64, 25, 12, 22, 11, using Selection Sort what is time complexity of Selection Sort in the best, average and worst cases?

A- Given array.

64 | 25 | 12 | 22 | 11

=> 11 | 25 | 12 | 22 | 64

=> 11 | 12 | 25 | 22 | 64

=> 11 | 12 | 22 | 25 | 64

Time complexity

Best - $O(n)$

Average $= O(n^2)$

worst $= O(n^2)$.

(19) Sort the following elements using insertion sort using Brute force Approach strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyze complexity of the algorithm.

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

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

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

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

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

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

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

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

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

Time complexity

Best - $O(n)$

Avg - $O(n^2)$

worst case - $O(n^2)$

$O(1)$ - Insertion sort.

20 Give an array of {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 analysis complexity of algorithm

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Time complexity :-

Best :- $O(n)$

Avg :- $O(n^2)$

worst :- $O(n^2)$