CSA0666 - Design & Analysis of Algorithm.

192377055
Jacinth Priscilla

Analytical Assignment -2

1) If $T_1(n) \in O(g_1(n))$ and $T_2(n) \in O(g_2(n))$ then $T_1(n)+T_2(n) \in O(\max\{g(n), g_2(n)\})$. Proove the assertions.

=> for any real numbers $a_1, b_1, a_2 b_2$, such that $a_1 \le b_1$ and $a_2 \le b_2$, we have $a_1+a_2 \le 2 \max\{b_1, b_2\}$.

Given:-

$T_1(n) \in O(g_1(n))$ for all non-negative integers.

n_1 & c_1

$T_1(n) \le c_1 g_1(n)$ for all $n \ge n$.

also since $T_2(n) \in (g_2(n))$ then there is constant c_2 & non negative integer n_2

$T_2(n) \le c_2 g_2(n)$ for all $n \ge n_2$.

$c_3 = \max\{c_1, c_2\}$ & $n_0 = \max\{n_1, n_2\}$.

$T_1(n) + T_2(n) \le c_1 g(n) + c_2 g_2(n)$

$\le c_3 g_1(n) + c_3 g_2(n)$

$= c_3\{g_1(n) + g_2(n)\}$

$\le \max\{g_1(n) + g_2(n_2)\}$.

$T_1(n) + T_2(n) \in O(\max\{g_1(n) + g_2(n)\}$

Hence Proved....

Time Complexity

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

Given,

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

using Mater's theorem

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

$a=2$; $b=2$; $f(n)=1$; $K=0$.

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

$\log_b^a > K$, $T(n) = \Theta(n^{\log_b^a})$

$= \Theta(n^1)$

$= \Theta(n)$

Time Complexity $T(n) = \Theta(n)$

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

Given:

$T(n) = 2T(n-1) \rightarrow (1)$. using Backward substitution.

$n = n-1$

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

Sub (2) in (1)

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

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

Sub $n=n-2$.

$T(n-2) = 2T(n-2-1) = 2T(n-3)$ → (4)

Sub (4) in (3)

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

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

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

Stop when, $n=k \Rightarrow n-k=0$

$T(n) = 2^n T(0)$

$T(0) = 1$

So $T(n) = 2^n \cdot 1$

$= 2^n$

} Time complexity = $O(2^n)$

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

Given:- $f(n) = n^2+3n+5$.

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

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

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

$n \geq 1$ $n^2+3n+5 \leq n^2+3n^2+5n^2$.

$n^2+3n+5 \leq (1+3+5)n^2$.

$n^2+3n+5 \leq 9n^2$.

Now $c=9$, $n_0 = 1$

$f(n)$ is in $O(n^2)$ WRT $c=9$

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

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

$f(n) = n$

Ω condition.

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

When, $n=1 \quad 1 \geq 1+2+4$

$1 \geq 7$ - No.

$n^4 = n^3+2n^2+4n.$

When,

$3^4 = 81$

$8^1 = 27+18+12$

$81 = 37$

$f(n) = n^4$ with $n \geq 3$ $g(n)$ is $\Omega(n^3)$

7) Big Theta Notation: determine whether $n(n) = 4n^2+3n$ is $\Theta(n^2)$ or not

Given, $n(n) = 4n^2+3n$ is $\Theta(n^2)$ or not

for theta, condition is $C_1 g(n) \leq f(n) \leq C_2(g(n))$

$C_1n^2 \leq 4n^2+3n \leq C_2n^2.$

Upper Bound $O(n^2)$

$4n^2+3n \leq C_2n^2.$

$n \geq 1 \quad 4n^2+3n \leq 4n^2+3n^2 = 7n^2$

$4n^2+3n \leq 7n^2$

Lower Bound

$4n^2+3n \geq c_1 n^2$

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

$c_1 = 4$

$n(n) = 4n^2+3n$ is $\Theta(n^2)$ with $c_1=4$ $c_2=7$ $n_0=1$

$n(n) = 4n^2+3n \in \Theta(n^2)$

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

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

$g(n) = n^2$.

for $\Omega(g(n))$, $f(n) \geq c \cdot g(n)$.

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

$n=1$ $1-2+1 \geq c \cdot 1$

$n=2$ $8-8+8 \geq 1 \cdot 4$

$8 \geq 4$

for the above answer for $n \geq 2$, $f(n) = \Omega(g(n))$ is true since it satisfies the Big Omega Condition.

9) determine whether $n(n) = n\log n + n$ is in $\Theta(n \log n)$ Prove a rigorous proof for your condition.

$c_1 n \log n \leq n(n) \leq c_2 \cdot n \log n$

Upper Bound

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

for $n \geq 1$, $\log n$ is positive

$n \log n + n \leq n\log n + n \log n$.

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

$c_2 = 2$,

$h(n) = n\log n + n \leq 2n\log n$ ∴ hence proved

Lower Bound

for $n \geq 1$, $\log n$ is positive

$n + n \log n \geq n \log n$.

$c_1 = 1$; $h(n) = n\log n + n \geq n\log n$.

Thus,

$n\log n \leq n \log n \leq n\log n + n \leq 2n\log n$ for all $n \geq 1$

$c_1 = 1$, $c_2 = 2$ & $n_0 = 1$

Hence $n(n) = n\log n + n$ is in $\Theta(n\log n)$

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

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

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

Master theorem.

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

here, $a = 4, b = 2, f(n) = n^2, k = 2, P = 1$

Now,

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

$\log_b^a = k$

Comparing with P,

$P \geq -1$, so, $\Theta(n^k \log_n^{P+1})$

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

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

Order of growth of $T(n) = 4T(n/2) + n^2$ is $\Theta(n^2 \log n)$.

Give array of [4, -2, 5, 3, 10, -5, 2, 8, -3, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] Multiplying two Integers from array

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

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

Max product

* Product of 2 largest (+ve) number
* Product of two smallest (negative) number

Min product

* Product of two largest +ve numbers.
* Product of two Smallest (-ve numbers).
* Product of Smaller negative number and.

Largest +ve

Max product,

↳ 10×11 = 110

↳ (-9) × (-8) = 72

Max product = 110

Min product,

↳ 10 × 11 = 110.

↳ -9 × -8 = 72

↳ -9 × 11 = -99

↳ Second Smallest -ve & second Largest positive -8 × 10 = -80.

Min product = -99

Max product = 110, Min product = -99

12) Demonstrate Binary search Method to search Key = 23 from array arr[] = {2,5,8,12,16,23,38,56,72,91}

Given:-

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

=> low = 0 arr[mid] = arr[4] = 16

high = 9

Mid = $\frac{0+9}{2}$

= 4

=> 16 < 23, search in right } 1st Iteration

now, low = mid + 1 = 5

=> Mid = 5+9/2 = 7 arr[mid] = arr[7] = 56 } 2nd Iteration

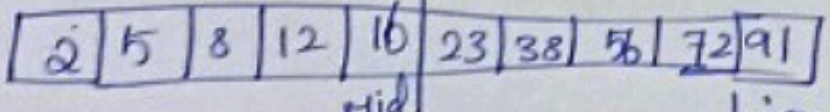
now, 56 > 23

high = Mid - 1 = 6.

=> Mid = 5+6/2 = 5 arr[mid] = arr[5] = 23 } 3rd iteration

23 = 23, key found at index 5

=>



=> 2 5 8 12 16 | 23 38 56 | 72 91

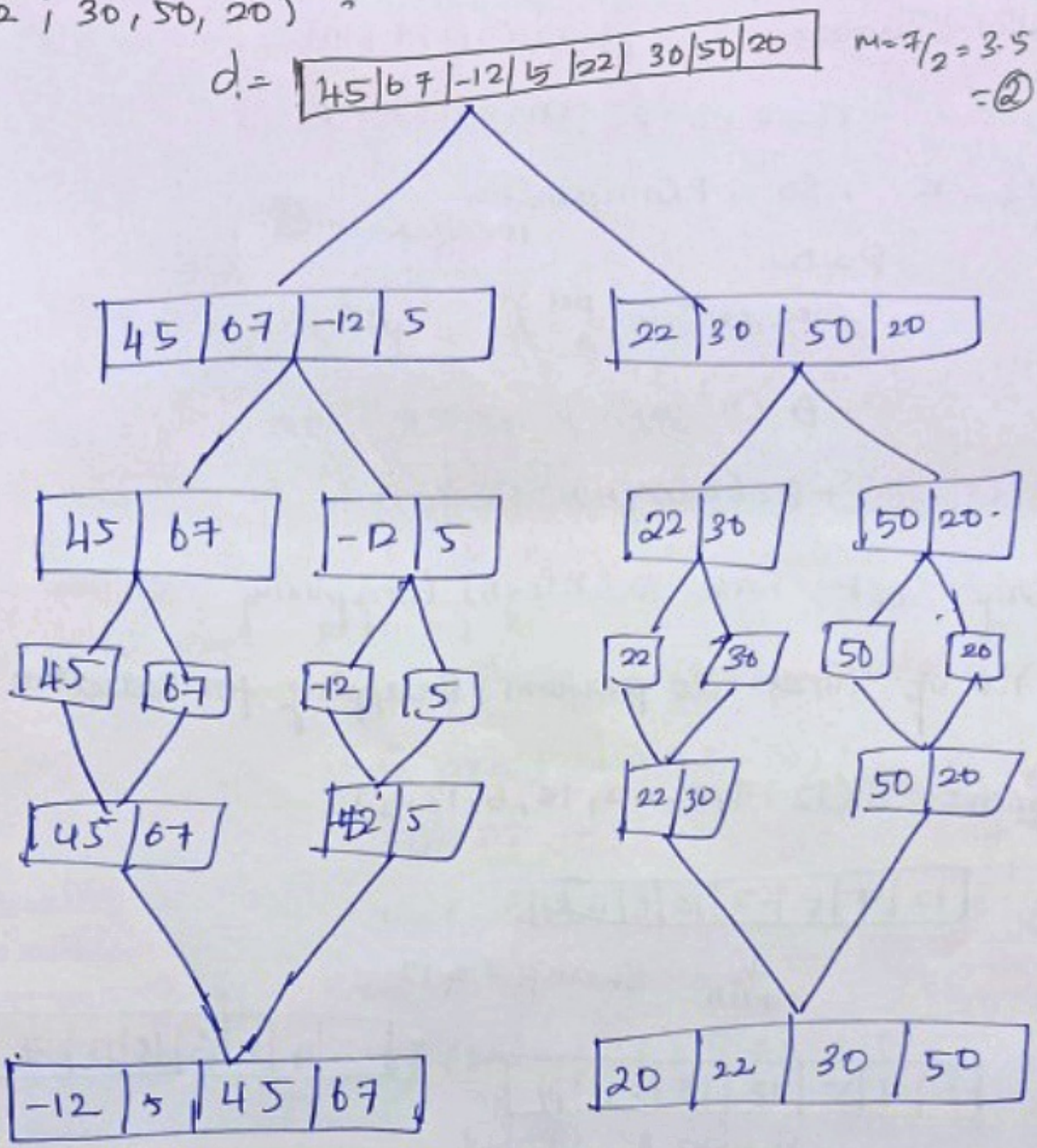
Mid high

=> 2 5 8 12 16 | 23 38 56 72 91

low Mid high

23 = 23 found at index 5

(B) Apply merge sort and order the list d = (45, 67, -12, 5, 22, 30, 50, 20).



Ordered List } -12 5 45 67 20 22 30 50

Recurrence Relation; Solving

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

Master Theorem $T(n) = aT(n/2) + f(n)$

$a = 2, b = 2, f(n) = n, K = 1$

$\log_b^a = K$, So, P Comparision

$P = 0$

$= \theta(n^K \log_n^{P+1})$

$= \theta(n^1 \log^1 n)$

$= \theta(n \log n)$

=> Merge Sort have $O(n \log n)$ Complexity

14) find no of Times to perform Swapping for selection Sort

Given = S(12, 7, 5, -2, 18, 6, 13, 4)

12	7	5	-2	18	6	13	4

↓ Min · Swap 2 & 12

-2	4	5	6	18	13	7

-2	7	5	12	18	6	13	4

Swap 4 & 7 ↑ Min

-2	4	5	12	18	6	13	7

↑ Min No swap.

-2	4	5	12	18	6	13	7

↓ Min
Swap 6 & 12

15) find index value of target value 10 using binary search from following list for elements [2,4,6,8,10,12,12,16,15]

Given,

arr[2,4,6,8,10,12,16,15,20]

Target = 10.

by using binary search

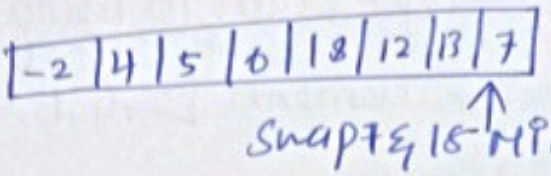
2	4	6	8	10	12	14	16	18	20

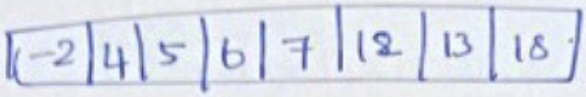
Mid = $\frac{0+9}{2}$ = 4.

Now, arr[mid] = = 10, target is found at index 4

Algorithm:-

```
low=0
high=9
While low ≤ high
    Mid = low + high/2
    if arr[mid] < target
        low=mid+1
    If arr[mid] > target
        high = mid-1
return -1
```





So swap = 4

Time Complexity

```
arr : array of items
    n : size
  for (i=0; i<n; i++  ---> n
  min = i
    for (j=i+1; j<n; j++) ---> n x n
    if arr[j] < arr[min] then -1
    Min = j  -1
    end if -1
    end for -1
    if (min != i) then -1
        Swap arr[min] and arr[i
        endif
        endfor
T(n) = O(n^2)
```