CSA-0669 192372007
M. SURYA NARAYANAN

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

(i) $t_1(n) \in O(g_1(n))$ means there exist constants $C_1 > 0$ and n_1 such that for all $n \geq n_1$; $t_1(n) \leq C_1 g_1(n)$

(ii) $t_2(n) \in O(g_2(n))$ means there exist constants $C_2 > 0$ and n_2 such that for all $n > n_2$: $t_2(n) \leq C_2 g_2(n)$.

We need to show that $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$.

Proof:

Consider: $t_1(n) + t_2(n)$ and the definition of $\max\{g_1(n), g_2(n)\}$ let $g(n) = \max\{g_1(n), g_2(n)\}$.

By definition of maximum, for all n:

$$g(n) = \max\{g_1(n), g_2(n)\} \geq g_1(n)$$

$$g(n) = \max\{g_1(n), g_2(n)\} \geq g_2(n)$$

Given the bounds of $t_1(n)$ and $t_2(n)$:

$t_1(n) \leq C_1 g_1(n) \leq C_1 g(n)$ for all $n \geq n_1$

$t_2(n) \leq C_2 g_2(n) \leq C_2 g(n)$ for all $n \geq n_2$

To bound $t_1(n) + t_2(n_1)$ consider

$t_1(n) + t_2(n)$

For $n \geq \max(n_1, n_2)$:

$t_1(n) + t_2(n) \leq C_1 g(n) + C_2 g(n) = (C_1 + C_2) g(n)$

Thus, there exists a constant $c = c_1 + c_2$ and $n_0 = \boxed{\Sigma}$
such that for all $n \geq n_0$: $t_1(n) + t_2(n) \leq c\,g(n)$
By the definition of Big O notation, this implies:
$t_1(n) + t_2(n) \in O(c\,g(n))$
Since $g(n) = \max\{g_1(n), g_2(n)\}$, we have:
$t_1(n) + t_2(n) \in O(\max\{g_1(n); g_2(n)\})$
Hence, we have proven the assertion.

---

2)(i) $T(n) = 2T(n/2) + 1$ if $n > 1$

$2T(n/2) + 1$ if $n > 1$

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

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

$\log_a b = \log_2 2 = \boxed{1}$

$f(n) = 1 \Rightarrow n^1 = n^k \Rightarrow \boxed{k = 1}$

$\log_a b = k$

Case 2:

$T(n) = \Theta(n^k \log_a b) = \boxed{\Theta(n)}$

$T(n) = 2T(n-1)$ if $n > 0$

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

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

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

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

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

let $k = n$

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

$T(0) = 1 \Rightarrow T(n) = \boxed{O(2^n)}$

6.) S.T : $f(n) = n^2 + 3n + 5$ is $O(n^2)$

$O \rightarrow f(n) < c \cdot g(n)$

$f(n) = n^2 + 3n + 5,\ g(n) = n^4$

$f(n) = 9 \quad > \quad g(n) = 1$

$f(n) = 21 \quad > \quad 16$

$f(n) = 23 < 71$

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

$\therefore f(n) \leq g(n)$

$\therefore$ Big O is proved.

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

for $\Omega(n^3)$

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

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

$n=1$ $g(n) = 7$ $c \cdot (n) = 1$

$n=1$ $g(n) = 24$ $c \cdot (n) = 8$

$n=3$ $g(n) = 57$ $c \cdot (n) = 27$

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

$\therefore$ $\Omega(n^3)$ is satisfied :

7) Determine $h(n) = 4n^2 + 5n$ is $O(n^2)$.

for $O(n^2)$

$$h(n) \leq c(n^2)$$

$$4n^2 + 5n \leq c(n^2)$$

$n=1$ $h(n) = 9$ $c(n^2) = 1$

$n=2$ $h(n) = 26$ $c \cdot (n^2) = 4$

$$h(n) > c(n^2)$$

$\therefore$ $\Omega(n^2)$ is possible

hence $h(n)$ is both $O(n^2)$ and

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

S.T $f(n) = \Omega(g(n))$

To prove: if $f(n) \geq c \cdot g(n)$

$n=1$ $f(n) = 0$ $g(n) = 1$

$n=2$ $f(n) = 2$ $g(n) = -4$

$n=3$ $f(n) = 12$ $g(n) = -9$

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

$\therefore \Omega(n)$ is Proved.

9.) Determine $h(n) = n\log n + n$ is $O(n\log n)$

$h(n) = n\log n + n$ to be $O(n\log n)$

Then $h(n) \leq c\,(n\log n)$

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

$n\log n + n = n(\log n + 1)$

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

$\div$ by n

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

$\div$ by $\log n$

$$\frac{\log n + 1}{\log n} \leq c_1$$

$$1 + \frac{1}{\log n} = c_1 = 2$$

To Check
$n\log n + n \leq 2(n\log n)$

$C(n\log n) \geq 0$

$n=1 \quad h(n)=1 \qquad C(n\log n)=4\log 2$

$n=2 \quad h(n)=2\log 2+2 \qquad C(n\log n)=6\log 3$

$n=3 \quad h(n)=3\log 3+3$

$\therefore h(n) < c \cdot g(n)\log n$

$\therefore$ we have proved
$h(n)$ is $O(n\log n)$.

(c) Find order of growth.

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

$a=4, \quad b=2$

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

$f(n) = n^k \log n^P = n^2$

$\boxed{K=2n}$

$\log_a b = \log_4 2 = 2$

$\log_a b = K$

Case 2 ::

$P=1$

$P>-1$

$\therefore T(n) = n^k \log^{P+1} n \quad \boxed{n^2 \log^1 n}$

$\therefore T(n) = \boxed{O(n^2 \log n)}$

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

Sorting the array:

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

Candidate for maxi product.

=> $10 \times 11 = 110$

=> $-9 \times -8 = 72$

Candidate for min product.

=> $-9 \times -8 = 72$

=> $-9 \times 11 = -99$

=> $-8 \times 11 = -88$

Max product = 110

Min product = -99

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

```
def binary_s(arr, key):
    s = 0
    end = len(arr)-1
    while start <= end:
        mid = (start+ end)//2
        if arr[mid] == key
            return mid
        elif arr[mid] < key
            start = mid+1
        else
            end = mid-1
```

(i) start = 0 , end = 9

mid = 4

16 < 23 , searches in the right half

(ii) 23, 38, 56, 72, 91

start = 5 , end = 9

mid = 7

56 > 23 , search in the left half

(iii) 23, 38

start = 5, end = 6

mid = 5

23 = 23 key

found

Q: [45, 67, -12, 5, 22, 30, 50, 20]

[45, 67, -12, 5, 22, 30, 50, 20]

[45, 67, -12, 5]   [22, 30, 50, 20]

[45, 67] [-12, 5]   [22, 30] [50, 20]

Sorted   Sorted   Sorted   [20, 50]

[-12, 5, 45, 67]   [20, 22, 30, 50]

[-12, 5, 20, 22, 30, 45, 50, 67]

Recurrence Relation:

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

$\log_a b = \log_2 2 = 1$

$f(n) = n = n^k \Rightarrow k = 1, \ p = 1$

$\log_a b = k \Rightarrow$ Case 1 $\Rightarrow p > -1$

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

$= \boxed{T(n) = \Theta(n \log n)}$

14) Find the no of times to perform ... selection sort also estimate the time complexity for the order of notation set S = [12, 7, 5, -2, 18, 6, 13, 4]

(i) [-2, 7, 5, 12, 18, 6, 13, 4]
(ii) [-2, 4, 5, 12, 18, 6, 13, 7]
(iii) [-2, 4, 5, 6, 18, 12, 13, 7]
(iv) [-2, 4, 5, 6, 7, 12, 13, 18]

Total no of swaps = 4

Best case = $O(n^2)$ worst case is also $O(n^2)$

$\therefore$ ~~...~~ Time complexity
$T(n) = O(n^2)$

---

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

Initially => start = 0, end = 9

$$\text{Mid} = \frac{\text{start} + \text{end}}{2} = \frac{0+9}{2} = 4$$

list [mid] = list [4] = 10

$\therefore$ Target is found in Index [4]

... elements using merge sort divide and ~~Conquer~~ Conquer strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 82, 52, 60, 5] and analyse complexity of algorithm.

Sol

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

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

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

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

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

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

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

Recurrence Relation:

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

$\log_b a = \log_2 2 = 1$

$f(n) = n \Rightarrow n^k, k = 1, p = 1$

$\log_b a = k =$ case - 1

$P > 1 : O(n^k \log_n^{P+1})$

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

17.) Sort the array [64, 34, 25, 12, 22, 11, 92] using bubble sort, what is the time complexity of selection sort in the best, worst, and average case.

1st pass → [34, 64, 25, 12, 22, 11, 92]

[34, 25, 64, 12, 22, 11, 92]

[34, 25, 12, 64, 22, 11, 92]

[34, 25, 12, 22, 64, 11, 92]

[34, 25, 12, 22, 11, 64, 92]

[34, 25, 12, 22, 11, 64, 92]

2nd pass = [25, 34, 12, 22, 11, 64, 92]

[25, 12, 34, 22, 11, 64, 92]

[25, 12, 22, 34, 11, 64, 92]

[25, 12, 22, 11, 34, 64, 92]

3rd pass = [12, 25, 22, 11, 34, 64, 92]

[12, 22, 25, 11, 34, 64, 92]

[12, 22, 11, 25, 34, 64, 92]

4th pass → {12, 11, 22, 25, 34, 64, 90}

5th pass → {11, 12, 22, 25, 34, 64, 90}

Best case : $O(n^2)$

Worst case : $O(n^2)$

Average case : $O(n^2)$

) Sort the array [64, 25, 12, 22, 11] using Selection Sort. What is the time complexity in the best, worst and average cases

1st pass → [11, 25, 12, 22, 64]

~~2nd pass [11, 12, 25, 22, 25, 64]~~

2nd pass → [11, 12, 25, 22, 64]

3rd pass → [11, 12, 22, 25, 64]

~~4 the pass → [11, 12,~~

Sorted ⇒ [11, 12, 22, 25, 64]

Best case : $O(n^2)$

Worst case : $O(n^2)$

avg case : $O(n^2)$

14) Sort the elements using Insertion Sort using Brute Force approach strategy [38, 27, 43, 3, 9, 82, 10, 15, 88, 52, 60, 5] and analyse complexity of the algo.

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

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

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

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

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

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

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

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

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

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

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

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

using
3, 9, -8, 2
of the

case : O(n)
case : O(n²)
case : O(n²)

Given an array [4, -2, 5, 3, 10, -5, 2, 8, -5, 6, 7, -4, 1, 9, -1, 0, -6, -8, 11, -9] integers, sort the following elements using Insertion Sort using brute force Approach Strategy analyse Complexity of the algorithm.

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

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

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

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

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

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

Worst case -> $O(n^2)$

Best Case -> $O(n)$

Average Case -> $O(n)^2$