

Data Structures

2. Giving suitable example discuss Quick sort and Bubble sort algorithm and its category (In place or out-of-place sorting algorithm). Discuss its time complexity and compare it with merge and insertion sort. Also make program of each algorithms.

Submitted By

Name : Harsh Soni

Roll No.: 11912082

Branch: IT

Q2

QUICK-SORT (arr, p, q) : [INPLACE ALGORITHM]

```
1 if p < q:  
2   r = PARTITION (arr, p, q).  
3   QUICK-SORT (arr, p, r-1)  
4   QUICK-SORT (arr, r+1, q).
```

PARTITION (arr, p, q) :

```
1 x = arr[q]  
2 i = p-1  
3 for (j ← p to q-1 :  
4   if arr[j] < x:  
5     swap (arr[j], arr[+i])  
6 arr[+i] = x  
7 return i
```



Scanned with CamScanner

A Q5

* Since, the partition funⁿ is traversing through the array once, hence the time complexity of partition = $\Theta(n)$.

$$P(n) = \Theta(n) \quad \left\{ \text{Time complexity of partition} \right\}$$

* BEST CASE TIME COMPLEXITY OF QUICK SORT.

Best case occurs when we have a balanced split:
(i.e we get lucky each time we partition).

so, the recurrence relation becomes.

$$T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n). \quad \boxed{T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n)}$$

$$T(n) \approx 2T(n/2) + \Theta(n) \quad \left\{ \begin{array}{l} T((n-1)/2) + T(\frac{n-1}{2}) \\ + \Theta(n) \end{array} \right\}$$

a) using master's theorem.

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

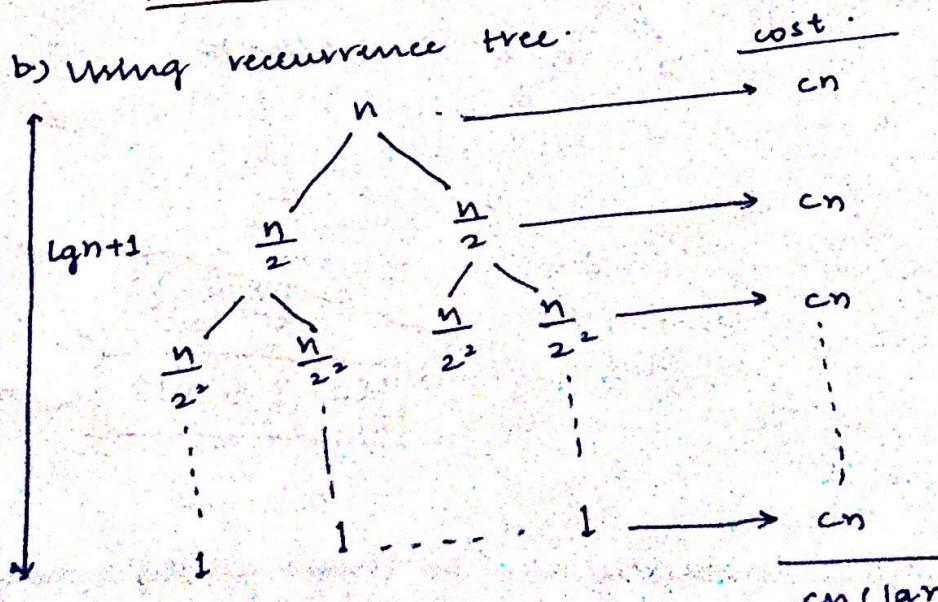
$$\text{where } a = b = 2,$$

$$f(n) = \Theta(n).$$

$$\Rightarrow T(n) = \Theta(n^{\log_2 a} \lg n).$$

$$\Rightarrow \boxed{T(n) = \Theta(n \lg n)}.$$

b) Using recurrence tree.



$$T(n) \approx \Theta(n \lg n)$$



* WORST CASE TIME COMPLEXITY OF Quick SORT.

The worst case occurs when the splits are extremely unbalanced.

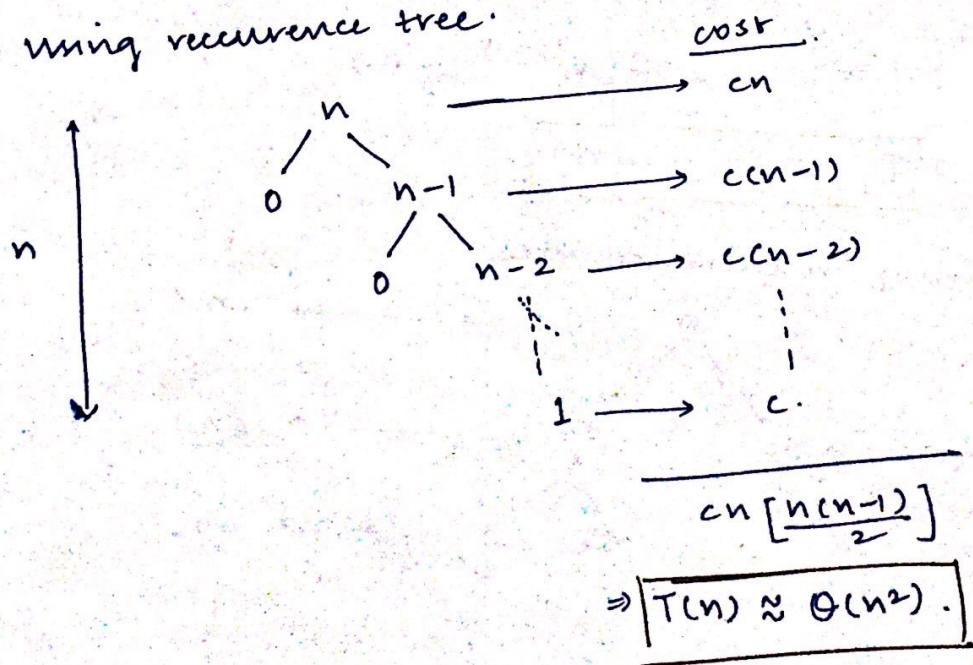
$$\begin{aligned} T(n) &= T(0) + T(n-1) + \Theta(n). \\ &= T(n-1) + \Theta(n). \end{aligned}$$

a) Using Backtracking

$$\begin{aligned} T(n) &= T(n-1) + \Theta(n) \\ &= T(n-2) + \Theta(n-1) + \Theta(n) \\ &= \sum_{k=1}^n \Theta(k) \\ &= \Theta(\sum k). \end{aligned}$$

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

b) Using recurrence tree

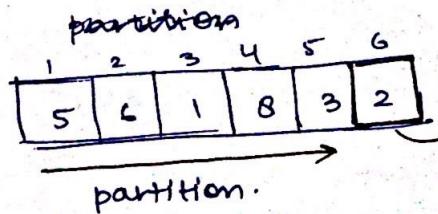
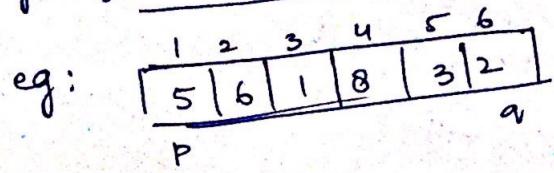


* Avg case time complexity can be found easily using above two relations

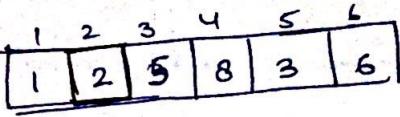
$$T(n) = O(n \lg n). \quad \{\text{Expected}\}$$



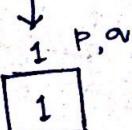
* eg of QUICK SORT



(NOTE the imbalance partitioning).

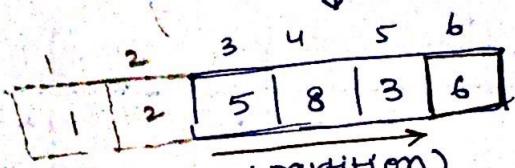


QUICK-SORT(arr, 1, 1)

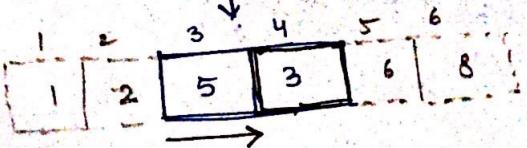


(partition)

QUICK-SORT(arr, 3, 6)

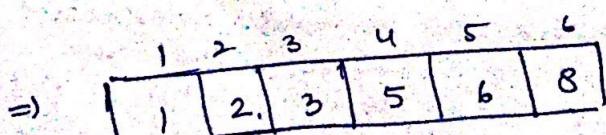
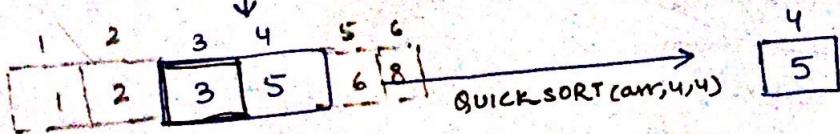


QUICK-SORT(arr, 3, 4)



partition.

QUICK-SORT (arr, 6, 6)



Scanned with CamScanner

For level 2,

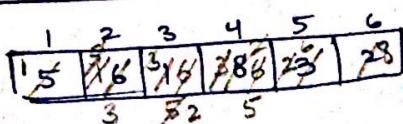
BUBBLE-SORT (arr).

[IN PLACE ALGORITHM]

```
1 for(i ← 1 to n):  
2   for(j ← 1 to n-i):  
3     if arr[j] > arr[j+1]:  
4       swap(arr[j], arr[j+1]).
```

~~swap take O(1) / constant time~~

* Time complexity of Bubble sort

eg: 

i	1	2	3	4	5	6
j	1, 2, 3, 4, 5	1, 2, 3, 4	1, 2, 3	1, 2	1	-
comparisons	1, 2, 3, 4, 5 ⑤	1, 2, 3, 4 ④	1, 2, 3 ③	1, 2 ②	1 ①	-
swaps	0, 1, 2, 3, 4 ④	1, 1, 2, 3 ③	0, 1, 2 ②	0, 0 ①	0 ①	-

Let the cost of computation of lines 3 & 4 be c_1 and c_2 altogether, then.

$$T(6) = c_1(5+4+3+2+1) + c_2(4+3+2)$$

*

⇒ Generalizing,

$$T(n) = \underbrace{c_1(n^2 - 1)}_{\text{number of comparisons}} + O(n^2)$$

$$\boxed{T(n) \approx O(n^2)}$$

* NOTE : Since, the comparisons will be made irrespective of the arrangement of elements, ∴ best case time complexity of bubble sort = $O(n^2)$.



Scanned with CamScanner

* MERGE SORT Vs QUICK SORT Vs INSERTION SORT
Vs BUBBLE SORT.

	MERGE SORT	QUICK SORT	: INSERTION SORT	BUBBLE SORT.
TIME COMPLEXITY	$O(n \lg n)$	$\Theta(n^2)$ * $O(n \lg n)$	$O(n^2)$	$O(n^2)$.
CATEGORY	OUT-OF-PLACE [$O(n)$]	INPLACE	INPLACE	INPLACE
SPACE COMPLEXITY (EXTRA)	$O(n)$	—	—	—
EFFECT OF ALREADY SORTED I/P	—	—	ACHIEVES BEST CASE [$O(n)$]	—

* → Randomized Quick sort can be used to achieve an expected running time of $O(n \lg n)$ moreover an avg case time complexity argument of $O(n \lg n)$ can also be made for quick sort.