CSE 373 / L-9 / 05.03.2024 /

H.W. from Heap Sort

6.1-2	6.2-1	6.3-1	6.4-1	6.5-1 to 5
6.1-6 to 8	6.2-3 to 7	6.3-3	6.4-3 to 4	6.5-10

Chapter-7

Quicksort

⊛ Applied the divide-and-conquer method.

Divide: A[p : r]

A[p : q-1]   ~~A[p:~~ A[q : q]   A[q+1 : r]

↳ Pivot, already in the correct place

Conquer: recursively each of the subarray are sort.

Combine: Already sorted, no work needed here.

Quicksort - Algorithm

QUICKSORT (A, p, r)

if $p < r$

  q = PARTITION (A, p, r)

  QUICKSORT (A, p, q-1) → all numbers here are lower than A[q]

  QUICKSORT (A, q+1, r) → all numbers here are higher tha A[q]

PARTITION (A, p, r)

  x = A[r]

  i = p-1 $\boxed{\text{Slide-4}}$

  for j = p to r-1

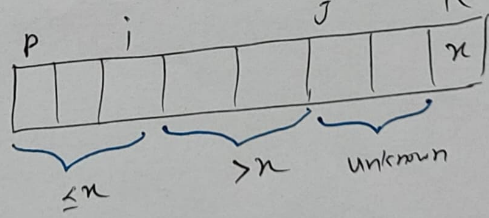
    if A[j] ≤ x

      i = i+1

      exchange A[i] with A[j]

  exchange A[i+1] with A[r]

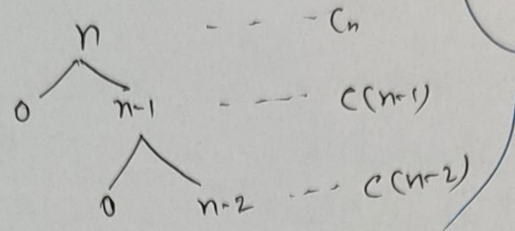
  return i+1



Quick-Sort Analysis:

- depends on which elements are used for partition.
- for balance partition, as fast as merge sort.
- for unbalance partition as slow as insertion sort.

Quick Sort - Worst Case



Sum = $\Theta(n^2)$

When already sorted

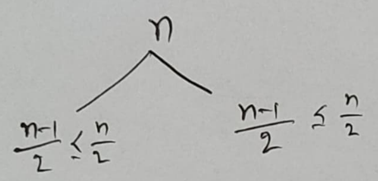
better than insertion sort

Running time,

$$T(n) = T(n-1) + T(0) + \Theta(n)$$

$$= T(n-1) + \Theta(n)$$

Quick-Sort - Best Case



Running time:

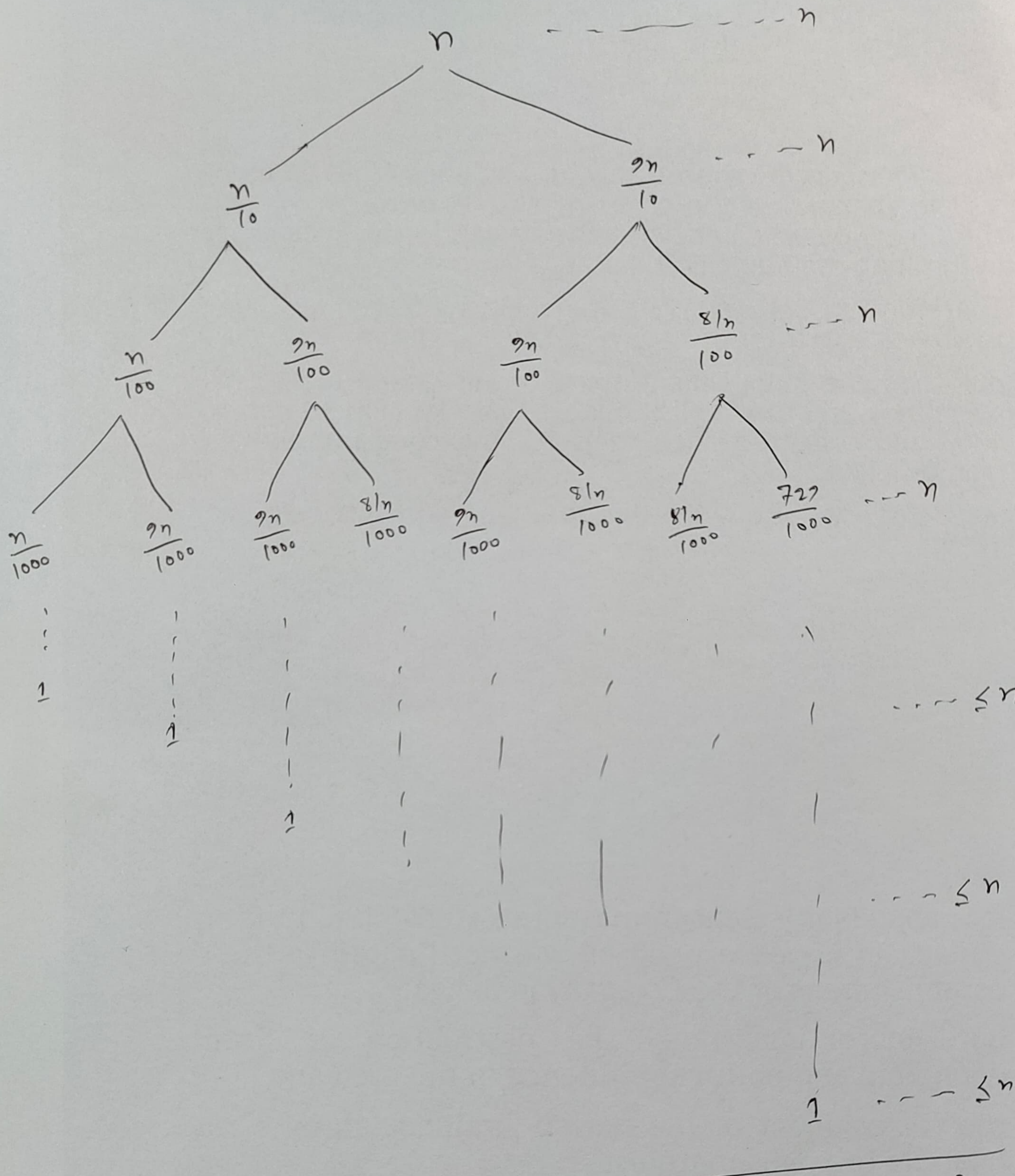
$$T(n) = 2T(n/2) + \Theta(n)$$

$$= \Theta(n \lg n)$$

Quick-Sort - Balanced Partition

Average case running time is much closer to the best case than to the worst case.

$$T(n) = T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + \Theta(n)$$



Total = $O(n \lg n)$

H.W. ⇒ 7.1-1 to 4

7.2-2 to 4

L-10/ 10.03.2024/

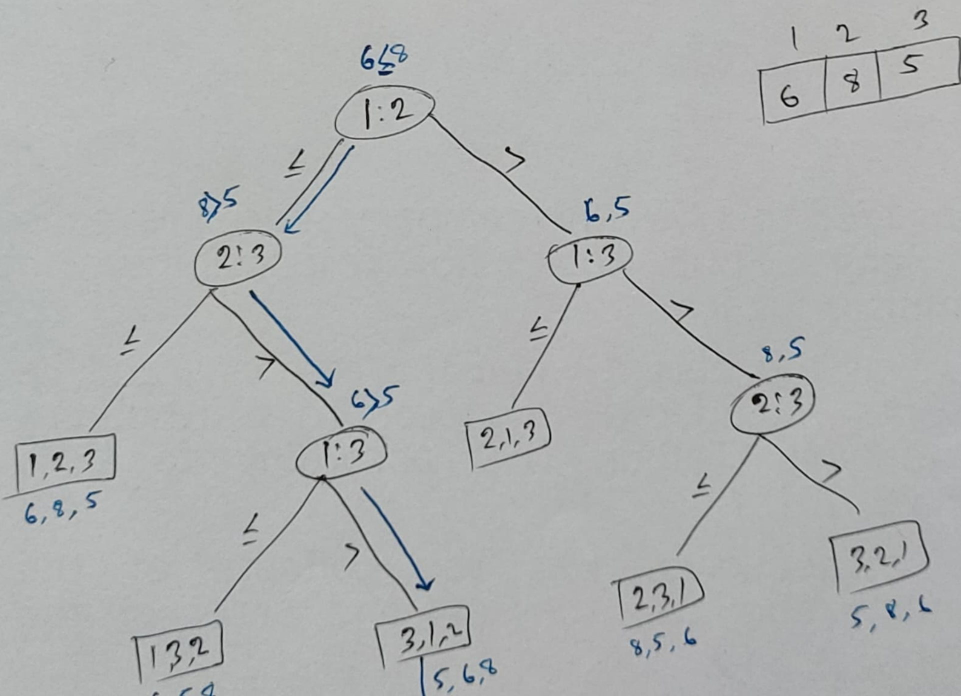
Chapter - 8

Sorting in Linear Time

Linear Time Sorting

Decision Tree

- Full binary tree
- each node is either a leaf or has both child.



leaf node are the decision made from the trea, in this case sorted.

Worst case running time

= height of tree

=

= $\Omega(n \lg n)$

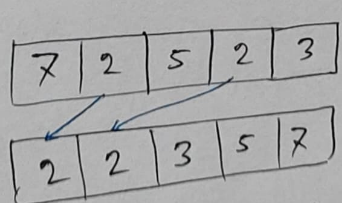
Theorem: Any decision tree that can sort n elements must have height $\Omega(n \lg n)$

Proof: The tree must contain $\geq n!$ leaves, since there are $n!$ possible permutations. A height-h binary tree has $\leq 2^h$ leaves. Thus, $n! \leq 2^h$.

$$\therefore h \geq \lg(n!)$$
$$\geq \lg\left(\left(\frac{n}{e}\right)^n\right) = n \lg n - n \lg e$$
$$= \Omega(n \lg n)$$

Counting Sort:

- No comparisons between elements.
- Need to know the range of integer.
- Stable



This order will maintain

$K = 7$

$A[1:5] \rightarrow$ Original Array

$B[1:5] \rightarrow$ Sorted Array

$C[0:7] \rightarrow$ Counting Array / Auxilary

✳ Counting Sort - Algorithm

COUNTING-SORT(A, n, k)

let B[1:n] and C[0:k] be new arrays

Initializing $\theta(k)$ { for i = 0 to k
    C[i] = 0

Counting $\theta(n)$ { for j = 1 to n
    C[A[j]] = C[A[j]] + 1

Cumulative count $\theta(k)$ { for i = 1 to k
    C[i] = C[i] + C[i-1]

Sorting $\theta(n)$ { for j = n downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1

return B

Total Time = $\theta(k+n)$
if $k = O(n)$
then, time = $\theta(n)$

k=5

⇒

A

1	2	3	4	5	6	7	8
2	5	3	0	2	3	0	3

C

0	1	2	3	4	5	
0	0	0	0	0	0	→ after 1st Loop
2	0	2	3	0	1	→ 2nd Loop
2	2	4	7	7	8	→ 3rd Loop

B

1	2	3	4	5	6	7	8	
0	0	2	2	3	3	3	5	→ 4th Loop

L-11 / 12.03.2024

Quiz-1