CS455 WK 5 HW1 Que9

Compare search algorithms: "Merge Sort + Binary Search" vs. "Linear Search"

Step 1 =>

Merge sort

1	5	9	4	3	8	7	6

[1 5 9 4]   [3 8 7 6]

[1 5]   [9 4]   [3 8]   [7 6]

[1] [5]   [9] [4]   [3] [8]   [7] [6]

[1 5] [4 9]   [3 8] [6 7]

[1 4 5 9]   [3 6 7 8]

[1 3 4 5 6 7 8 9]

Binary Search=> sorted array

1	3	4	5	6	7	8	9

L=0   M=5   H=8

1	3	4	5	6	7	8	9

L=5   M=7   H=8

1	3	4	5	6	7	8	9

L=5   H=6

1	3	4	5	6	7	8	9

L=H=5 , value=6

1	3	4	5	6	7	8	9

Time Complexity Analysis =>

Merge Sort Time Complexity Analysis=>

$T(n) = 2\, T(n/2) + O(n)$

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

Binary search Time Complexity Analysis=>

Given a sorted array A[0: n-1] and a search key

= if KEY = A[m], then return m

= if KEY < A[m], then search the left half of the array.

= if KEY > A[m], then search the right half of the array

$T(n) = 1\, T(n/2) + \theta(1)$

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

Description=> you tally with every element in half of the array to search value in sorted array. Time required to find value number becomes half of the length of array, every time in Binary search.

Step 2 =>

Linear Search => find value number v=6

1	5	9	4	3	8	7	6

i= 0 , v=6 ?

1	5	9	4	3	8	7	6

i= 1 , v= 6 ?

1	5	9	4	3	8	7	6

i= 2, v= 6 ?

1	5	9	4	3	8	7	6

i= 3, v= 6 ?

1	5	9	4	3	8	7	6

i= 4, v= 6 ?

1	5	9	4	3	8	7	6

i= 5, v=6 ?

1	5	9	4	3	8	7	6

i= 6, v=6 ?

1	5	9	4	3	8	7	6

i= 7 , v= 6 ? yes

1	5	9	4	3	8	7	6

Description=>

In unsorted array, while searching the element value , you go through every element in array to check if the value matches with the given number to find. If the element is at last index, then you walk through complete array of length n.

So the Time Complexity Analysis for Linear Search is

T(n)= n

Step 3=>

=>Master Theorem Approach =>

Recurrence for binary search=> T(n) = 1 T(n/2) + theta(1)

n ^ log(base b)a = n ^log(base 2)1 = n^0 = 1

⇨ T(n) = theta(log n)

=>Induction approach=>

Base: n=1: from recurrence, f(1) =1.

Claim: f(1) = [log 1] + 1 = 1. So the base case is correct.

Step: for some integer k

2^k <= n <= 2^ k+1

[log n]=k

2^k-1 <= [n/2] < 2^k

[log[n/2]] = k-1

f(m)= [log m] + 1,   m<n

for m =[n/2],

f([n/2]) = [log[n/2]] + 1 = (k-1)+1 =k=[log n]

then,

f(n)= f([n/2])+1 = k+1 = [log n] +1.

This completes the induction proof.