Page No.:
Date: / /

Tutorial-2 DAA

U20CS110
Krishna Pandey

Given the following algorithms, answer the questions
- Linear search
- Binary search
- selection sort

1. Analyse the time complexity of above algorithms and write recurrance rel^n for the same.

⇒ Linear search : The idea behind linear search is to search given element x linearly in the given array

→ A recursive approach to linear search. first search the element in the 1st locn if not found it recursively calls the linear search with the modified array without the 1st element

let $T(n)$ be no. of comparsion (i.e. time) required for linear search on an array of size n.

when. $n=1$ $T(1) = 1$

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

$$= 1 + (1+1+1 \ldots T(1))$$

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

Page No.:
Date: / /

$T(n) = O(n)$

→ Binary search

→ The approach is to check weather $A[\frac{n}{2}] = x$. If $x < A[\frac{n}{2}]$ then consider lower half of the array or else upper half of the array

→ After every iteration problem size reduces by half

Recurance relⁿ is $T(n) = T(\frac{n}{2}) + 1$

$T(n)$ → time required for binary search in an array of size n

$$T(n) = T(\frac{n}{2}) + 1$$

$$T(n) = T\left(\frac{n}{2^k}\right) + 1 + 1 + \ldots 1$$

Since $T(1)$ $T(1) = 1$

when $n = 2^k$ $T(n) = T(1) + k$

$k = \log_2 n$

$$T(n) = 1 + \log_2 n$$

$$\log_2 n \leq 1 + \log_2 n \leq 2 \log_2 n \quad \forall n \geq 2$$

$$T(n) = O(\log_2 n)$$

Page No.:
Date: / /

→ Selection sort : We need to sort an array using selection sort, for that we have to find index of minimum element. Each individual iteration takes a const. time.

→ The no. of iteration of thin loop is n in first call then n-1 & so on.

→ The recurrance rel^n is

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

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

$$T(n) = T(0) + T(1) + T(2) + \dots n-1 + n$$

$$T(n) = \frac{n(n+1)}{2}$$

$$T(n) = \frac{n^2}{2} + \frac{n}{2}$$

We know that, there are n calls to swap and each call takes const. amound of time.
using asymptotic notation, $O(n^2)$ term is most significant, so running time of sel. sort is $O(n^2)$

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

Page No.:
Date: / /

2- Assume that you don't know the time complexity of above algorithm.

① Can you predict the same based on your implementation of above algorithm.

Sol- ① Linear search In worst case we have to compare n elements of array, so it takes $O(n)$ operations by implementation

② Binary search By implementation in each recursive call the search get reduced by half of the array so, for n elements, there are $\log_2 n$ recursive calls.

③ Selection sort $O(n^2)$ becz there are 2 nested loop.

from above we can say that we can predicts via proving recursive relnship

(ii) Do they match with theoretical time complexity Yes/ No

Ans Yes

(iii) If yes then write the time complexity of all if NO, then write the difference

Lineary search = $O(n)$
Binary search = $O(\log_2 n)$
Selection sort = $O(n^2)$