Design Analysis of Algorithm

DAA

7/1/25

Algorithm : Set of rules by which we can perform certain logical tasks

Program	Algorithm
(i) Implemention	(i) Design
(ii) C, c++, Py	(ii) Any language
(iii) Dependent of hardware	(iii) Independent of hardware.

Characterictics:

(i) Finiteness → Finite no of steps
(ii) Definiteness → performs logically
(iii) Input 0 or more. gives 1 output or more

Analysis: Priority Analysis:

(i) Time
(ii) space

Q. Swap (a, b)

```
begin
    temp := a      ——— 1
    a := b         ——— 1
    b := temp      ——— 1
    return a and b ——— 1
```

Time / 4 unit — $f(n)=4$

Space / 3 unit — $s(n)=3$

Time comp $O(1)$
Space comp $O(1)$

Q. Begin

```
sum = 0                                   ——— (1)
for (i = 0; i < n; i++)                   ——— (n+1)
                                          ——— (n)
    S = S + A[i]
return sum                                ——— (1)
```

$2n+2$
$2(n+1)$
$\Rightarrow n+1$

$f(n) = (n+1) + n + 1$

$= 2n + 3$

$O(n)$

Space comp → sum → 1
A[] → n
i = → 1

$S(n) = n + 2$

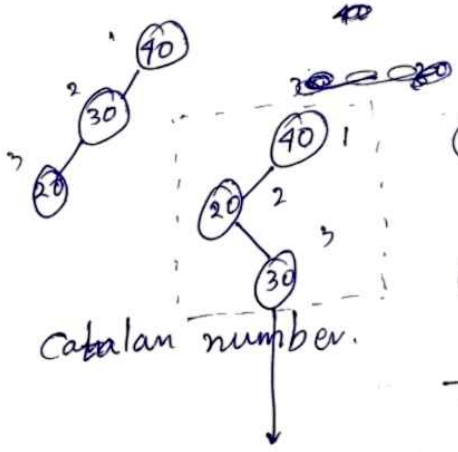
$\Rightarrow O(n)$

8/1/25 SKM

Optimal Binary Search Tree

key	20	30	40
frequency	2	1	6

construct BST



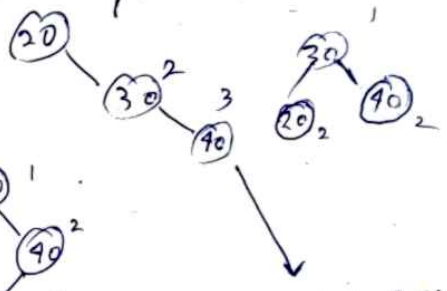
Catalan number.

$\frac{2n!}{n!(n+1)!}$

cost $1\times6 + 2\times2 + 3\times1$

$= 17$

optimal BST



$1\times2 + 2\times1 + 3\times6$

$= 22$

9/1/25 SKM

Design Strategy:

1. Divide & conquer Approach
2. Greedy Method
3. Dynamic Programming
4. Back tracking
5. Branch & Bound.

Divide & Conquer:

Problem divided into solvable subproblem using recursive function.

Algo:

```
D&C (Problem P)
  if subproblem(P) is small then
      solution (subproblem (P))
  else
      Divide(P) into P1, P2, ... Pn
      D&C (P1) - D&C (P2)
  combine (solution1, solution2 ..)
  end if
```

Q. Problem: Binary Search (A, lo, hi, x)

```
if (lo > hi)
    return False
mid := (lo + hi) / 2
if (A[mid] == x)
    return mid
else if (x < A[mid])
    Binary Search (A, lo, mid-1, x)
else
    Binary Search (A, mid+1, hi, x)
```

solution / conquer

Recurrence relation:

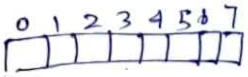
$$T(n) = \begin{cases} T(n/2) + 1 & ; \text{if } n > 1 \\ 1 & ; \text{if } n = 1 \end{cases}$$

$T(n/2)$ ↑ for divide; 1 ↑ conquer

LAB

n = Array size

time complexity = $O(\log n)$



to find 6th/7th element ; $n = 8$

time comp $O(\log_2 8) = 3 + 1 = 4$

```
int count = 0;
int binarySearch (int A[], int lo, int hi, int x){
    if (lo > hi){
        count++;
        return -1;
    }
    int mid = (lo + hi)/2;
    if (A [mid] == x){
        count ++;
        return mid;
    }
    else if (x < A[mid]){
        count++;
        binarySearch (A, lo, mid -1, x);
    }
    else {
        count++;
        binary search (A, mid+1, hi, x);
    }
}
```