PAGE 1

Recurrence Relation

fun" calls itself

Step1:- How to write r.r.

Step2:- How to solve r.r.

Let's write r.r. for Binary Search

wt is
↑Binary search ?
Searching an element in the sorted array

Let's consider the array we have is

0	1	2	3	4	5	6
10	20	30	40	50	60	70
i↑						j↑

Algorithm:-

```
BS(a, i, j, x)

mid = (i+j)/2
if (a[mid] == x)
    return(mid)
else
    if (a[mid] > x)
        BS(a, i, mid-1, x)
    else
        BS(a, mid+1, j, x)
```

PAGE 2

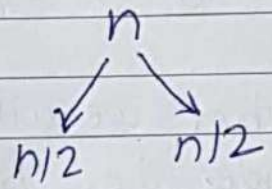
Let's consider x = 30
Now check value of i & j
i = 0, j = ~~0~~ 6

First find mid = (i+j)/2
= (0+6)/2
= 3

a[3] = 40
mid = 40

			mid			
10	20	30	(40)	50	60	70

So the problem i have with size n, is divided into 2 parts n/2 & n/2



Now check
if (a[mid] == x)
40 ≠ 30

If consider x = 40 the element is found & our search operation is complete

So time required for execution is O(C), constant time for finding mid & comparaision.

PAGE 3

Now let's continue with $x = 30$

$a[mid] = x$

$40 \neq 30$

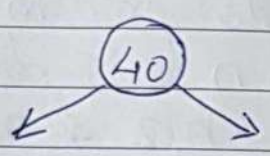
So you need to go to else part of your algorithm

check if $a[mid] > x$

$40 > 30$

condition is true

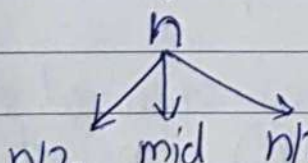
your problem is now divided into 2 parts



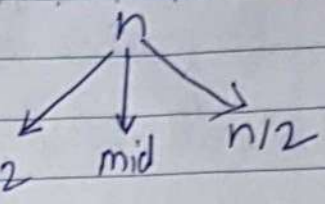
So either we will go to left side i.e. first sub problem or we will go second subproblem

As $40 > 30$ let's go to left sub problem

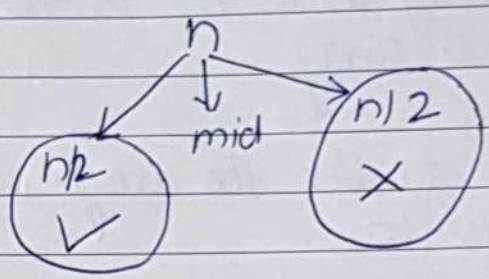
Let's say for problem of size n,



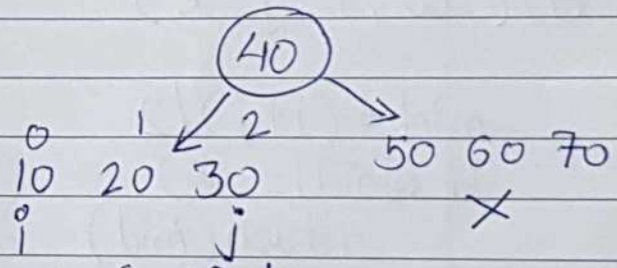
PAGE 4



if element we want to search is smaller than mid then we are interested in first part of problem. And we don't consider second part of problem.



Moving further left subproblem n/2 is divided into subparts.



$mid = (i+j)/2$

$= (0+2)/2$

$= 1$

$a[mid] = 20$

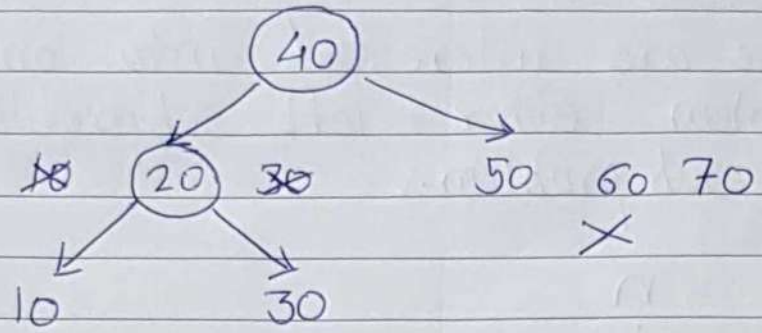
Let's compare mid with x

$a[mid] > x$

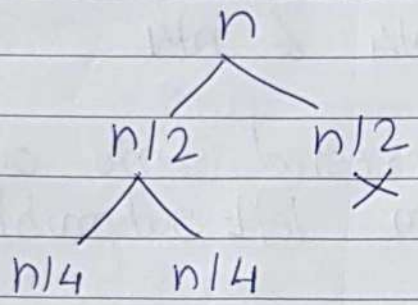
$20 > 30$ (False)

In that case we will go for the second condition & check a[mid] < x

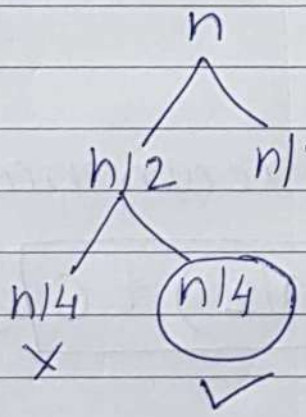
20 < 30 True



So we say for problem with size n,



As a[mid] < x i.e. 20<30 is True we are further going with second subpart of problem



Now let's see that we have a problem with size n, we divide it into 2 parts n/2 & n/2.

But we are interested into only one subproblem either left subproblem or ~~either~~ right sub problem.

n
↓
n/2

Further this n/2 we divide into two more subproblems n/4 & n/4

But we are interested into only one subproblem from left subproblem & right subproblem

n
↙
n/2
↓
n/4
↓
n/8

So what is the recurrence relation

$$T(n) = T(n/2) + C$$

constant time for finding mid or comparisions of mid & x

So this the recurrence relation for Binary search.

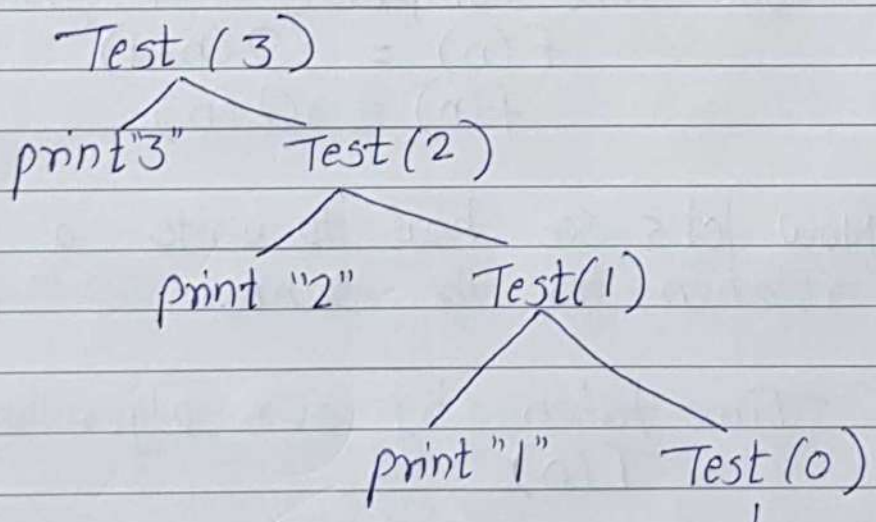
Now next step is how to solve recurrence relation..

Before we see how to solve a recurrence relation, let's take one more example of how we write recurrence relation.

Let's we have an algorithm as given below.

```
Void Test (int n)
{
   if (n>0)
   {
      print("%d", n)
      Test (n-1)
   }
}
```

Let's take n=3



Tracing Tree for recursive fun^n

As 0 >0 False
X

In every call if $n > 0$ it is printing a value & calling fun^n again

~~For~~ There are 2 tasks
① print a value
② call the fun^n again

For $n = 3$,
① 3 times print statement is executed

② 4 times the recursive call is made to the fun^n

For n, $n+1 \rightarrow$ calls
$n \rightarrow$ printing a value

Let's ignore time for printing a value as it constant. we have only $(n+1)$

So time complexity of given algorithm is
$f(n) = O(n+1)$
$f(n) = O(n)$

Now let's see how to write a recurrence relation for the same.

Time taken by our algorithm is let's say $T(n)$

So, $T(n) = C + T(n-1)$
↑ constant time for comparision & printing a value

Page :
Date :

Therefore,

$$T(n) = \begin{cases} T(1) & \text{for } n=0 \\ T(n-1) + c & \text{for } n>0 \end{cases}$$

This is the recurrence relation for the given algorithm.

Solving Recurrence Relation :-

1. The Substitution Method
2. Recursive Tree Method
3. Master's Method