Recursion and Searching

Recursion :- Function calling itself.

Base condition

Recurrance call should move towards the base condition

1) Factorial of a number

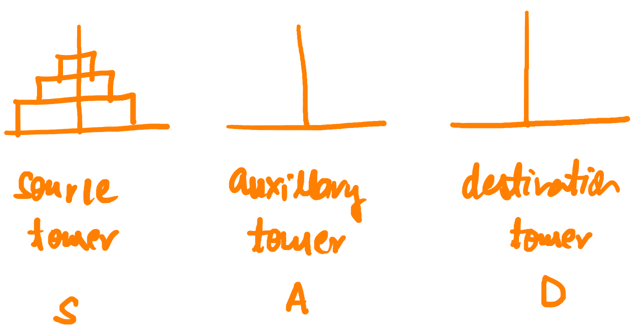
```
int fact (int n)
{
    if (n== 1 || n==0)
        return 1;

    return n * fact (n-1);
}
```

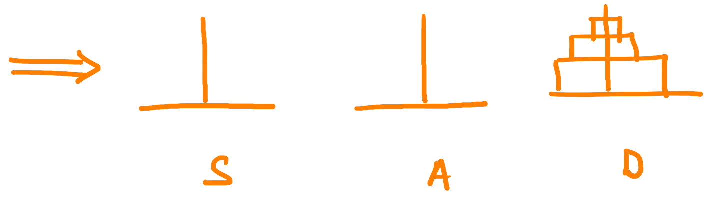
2) Fibonacci Number

0, 1, 1, 2, 3, 5, 8, . . .

```
int fib( int n)
{
    if (n== 1 || n==2)
        return n-1;
    return (fib (n-1) + fib(n-2));
}
```

3) **Tower of Hanoi :-**

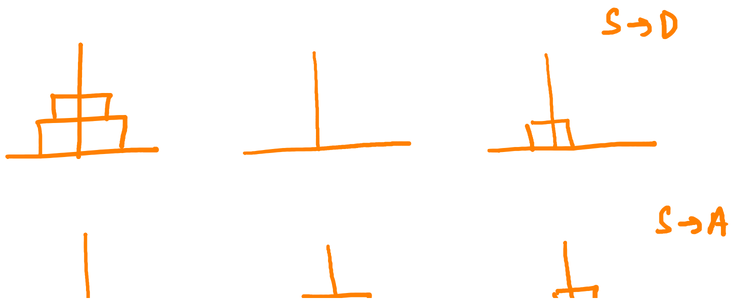




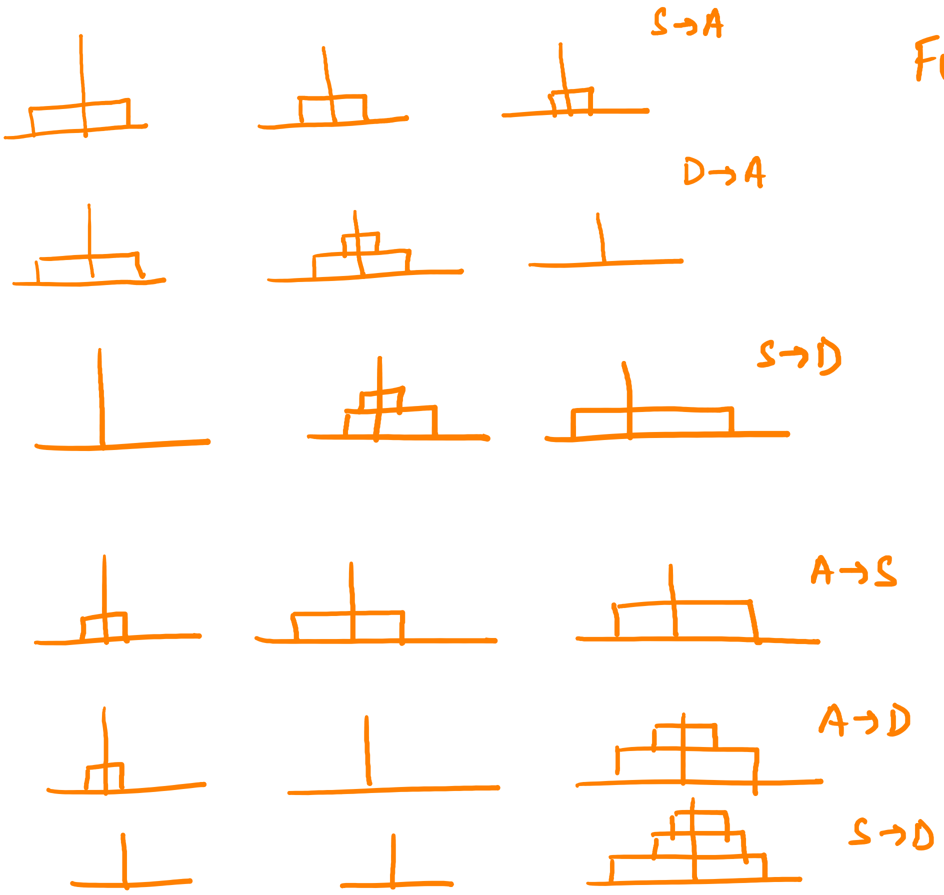
Two Constraints :-

✓ 1) Moving disks one at a time

2) Larger disk cannot be placed over a smallar disk.



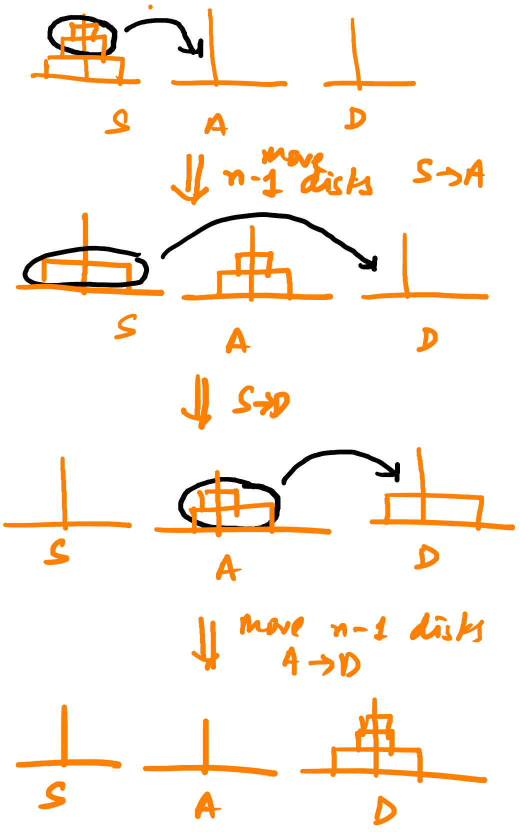
For n disks we require 2^n-1

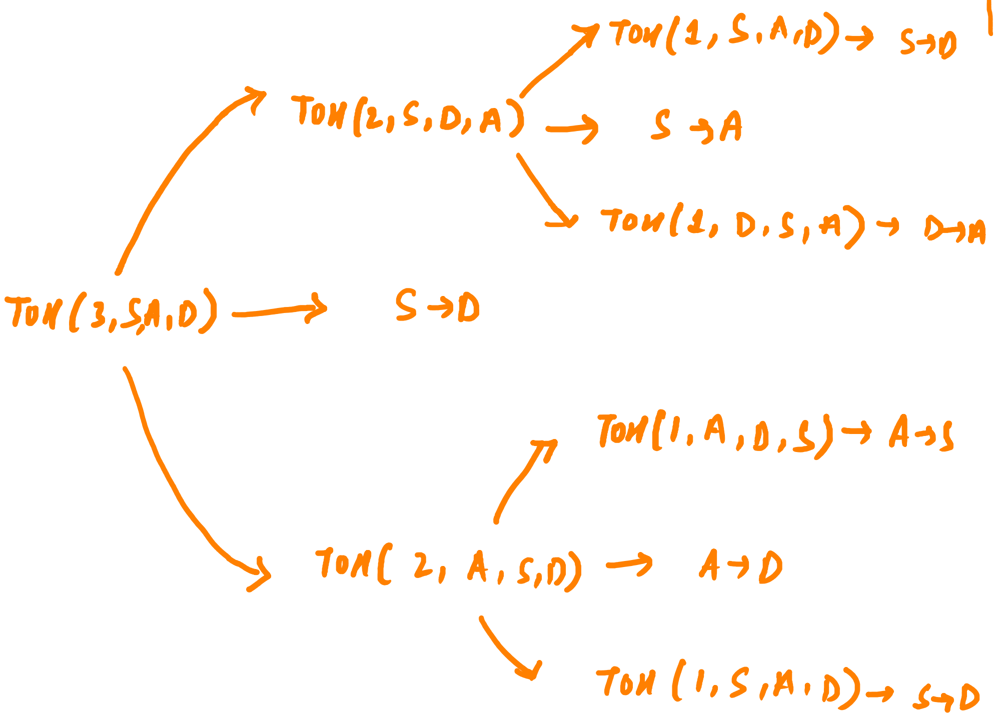


For n disks we require $2^n - 1$ moves

$O(2^n)$

```
TOH(n, S, A, D)
{
    if (n==1)
        print "S→D";
    else
    {
        TOH(n-1, S, D, A);
        print "S→D";
        TOH(n-1, A, S, D);
    }
}
```





Searching :- Given the data, return the position on which the element you wish to search is present.

Linear Search :-

Array ↓   #elements present   Item to search ↓

10	7	20	15	2

Linear Search:-

Array ↓   #elements present ↓   Item to search ↓

10	7	20	15	2
0	1	2	3	4

```
int linear search( int a[], int n , int x)
{
    int i;
    for(i=0; i<n ;i++)
        if( a[i]==x)
            return i;

    printf("Element not found");
    return -1;
}
```

O(n)

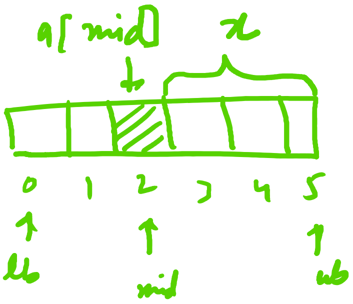
2) Binary Search:-

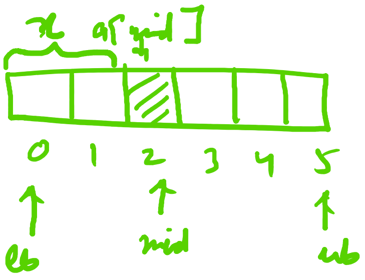
Data should be sorted

Mid point ⟷ Comparison with x

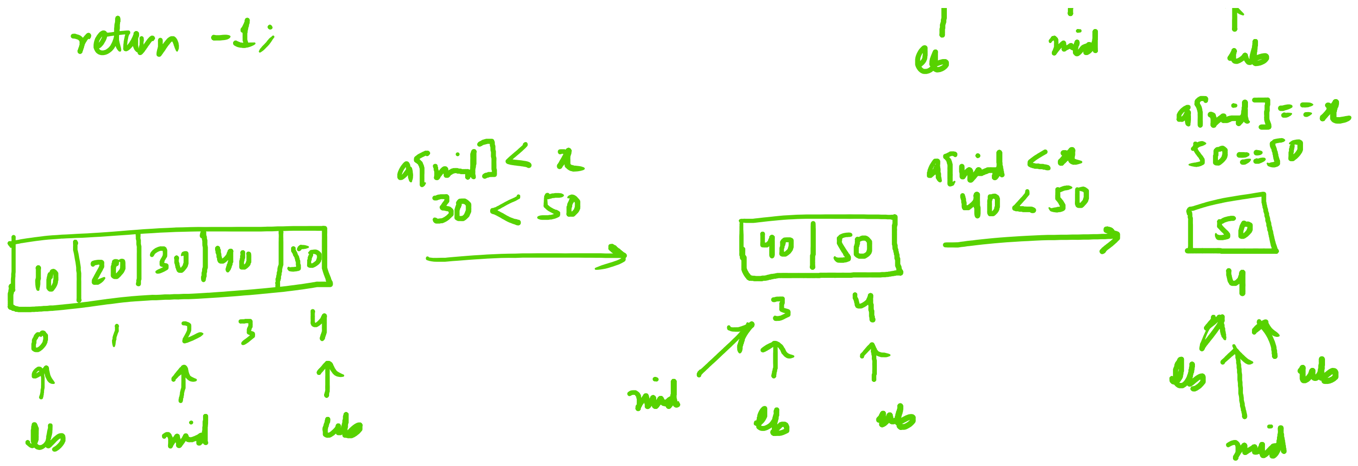
10	20	30	40	50
0	1	2	3	4

```
int binary search (int a[], int n, int x)
{
    int lb = 0, ub = n-1, mid ;

    while (lb <= ub)
    {
        mid = (lb+ub)/2;
        if (a[mid] == x)
            return mid;
        else if (a[mid] < x)
            lb = mid + 1;

        else
            ub = mid -1;
    }
    printf ("Element is not present");
    return -1;
}
```





```
}    return -1;
```



$$n \rightarrow \frac{n}{2} \rightarrow \frac{n}{2^2} \rightarrow \frac{n}{2^4} \; \ldots \rightarrow \frac{n}{2^i}$$

$$\frac{n}{2^i} = 1 \qquad n = 2^i \qquad \boxed{i = \log_2 n}$$

$$O(\log_2 n)$$