

# Recursion – 3

## Lecture-29

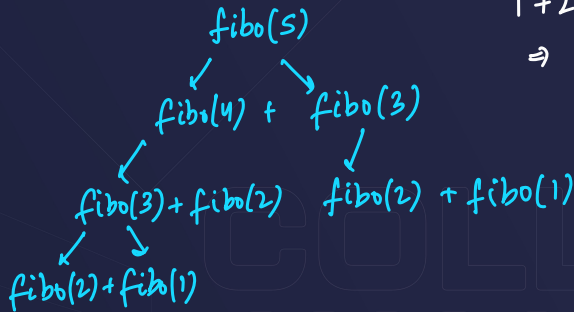
Raghav Garg

COLLEGE  
WALLAH

# Time Complexity of Fibonacci number

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

Stair



For 'n'

$$\underbrace{1 + 2 + 4 + 8 + \dots}_{n \text{ terms}}$$

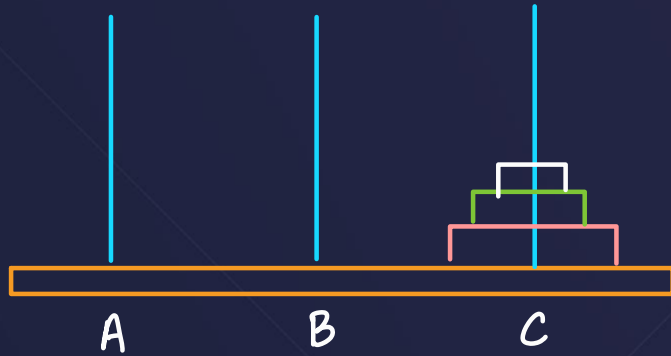
$$1 + 2^1 + 2^2 + 2^3 \dots 2^{n-1}$$

$$\Rightarrow \frac{2^n - 1}{2 - 1} = 2^n - 1$$

$$T.C. = O(2^n)$$

# Ques : Tower of HANOI

'Maza aa gaya'



Minimum	Moves	
$n$	$2^n - 1$	
1	$2^1 - 1$	= 1
2	$2^2 - 1$	= 3
3	$2^3 - 1$	= 7
4	$2^4 - 1$	= 15

A → C

A → B

C → B

A → C

B → A

B → C

A → C

A → C

A → B

C → B

A → C

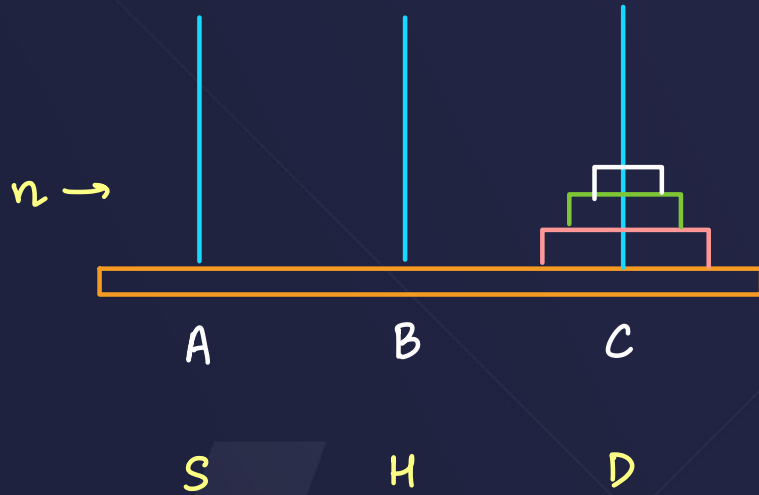
B → A

B → C

A → C

# Ques : Tower of HANOI

for  $n$  disks  $\rightarrow$   $S \rightarrow D$

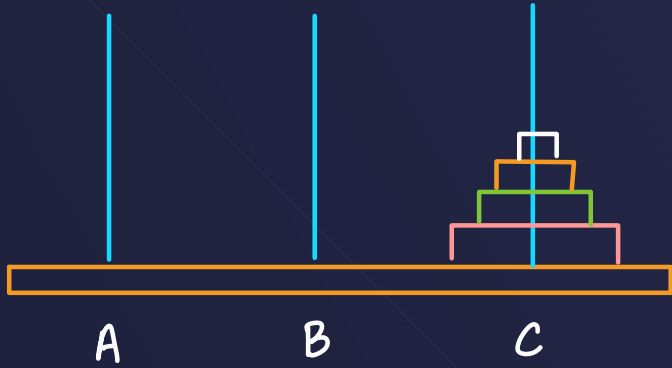


- 1)  $n-1$  disks :  $A \rightarrow B$  Recursion
- 2) Biggest disk :  $A \rightarrow C \rightarrow$  print
- 3)  $n-1$  disks :  $B \rightarrow C$  Recursion

hanoi( $n-1$ )

COLLEGE  
WALLAH

# Ques : Tower of HANOI



```
void hanoi(int n, char a, char b, char c){
    if(n==0) return;
    hanoi(n-1, a, c, b);
    cout<<a<<" -> "<<c<<endl;
    hanoi(n-1, b, a, c);
}
```

VERSION 37

- ✓ A -> B
- ✓ A -> C
- ✓ B -> C
- ✓ A -> B
- ✓ C -> A
- ✓ C -> B
- ✓ A -> B
- ✓ A -> C
- ✓ B -> C
- ✓ B -> A
- ✓ C -> A
- ✓ B -> C
- ✓ A -> B
- ✓ A -> C
- ✓ B -> C

# Traversing an array using recursion

Print all the elements of an array

```
for(int i=0; i<n; i++){  
    cout << arr[i] << " ";  
}
```

3

COLLEGE  
WALLAH

# Maximum Value of an array

Find out maximum element of an array using recursion

1) Print

2) Store

COLLEGE  
WALLAH

$\begin{matrix} 0 & 1 & 2 & 3 \\ \{1, & 5, & 2, & 3\} \end{matrix}$   
 $n = 4$

```

int maxInArray(int arr[], int1n, int0 idx){
    ✓if(idx==n) return INT_MIN;
    ✓return max(arr[idx], maxInArray(arr, n, idx+1));
}

```

Handwritten annotations: A yellow arrow points from the '0' above 'idx' to the '1' above 'n'. A yellow arrow points from the '5' above 'n' to the '1' above 'idx+1'. A yellow arrow points from the '5' above 'n' to the '5' above 'n' in the recursive call.

```

int maxInArray(int arr[], int4n, int1 idx){
    ✓if(idx==n) return INT_MIN;
    ✓return max(arr[5idx], maxInArray(arr, n, idx+1));
}

```

Handwritten annotations: The value '5' is circled in green next to 'arr[idx]'. The value '3' is circled in red next to 'maxInArray(arr, n, idx+1)'. A yellow arrow points from the '3' to the '3' above 'idx+1'.

```

int maxInArray(int arr[], int4n, int2 idx){
    ✓if(idx==n) return INT_MIN;
    ✓return max(arr[2idx], maxInArray(arr, n, idx+1));
}

```

Handwritten annotations: The value '2' is circled in red next to 'arr[idx]'. The value '3' is circled in green next to 'maxInArray(arr, n, idx+1)'. A yellow arrow points from the '3' to the '3' above 'idx+1'.

```

int maxInArray(int arr[], int4n, int3 idx){
    ✓if(idx==n) return INT_MIN;
    ✓return max(arr[3idx], maxInArray(arr, n, idx+1));
}

```

Handwritten annotations: The value '3' is circled in green next to 'arr[idx]'. The value 'INT\_MIN' is circled in red next to 'maxInArray(arr, n, idx+1)'. A yellow arrow points from the 'INT\_MIN' to the '4' above 'idx+1'.

```

int maxInArray(int arr[], int4n, int4 idx){
    ✓if(idx==n) return INT_MIN;
    return max(arr[idx], maxInArray(arr, n, idx+1));
}

```

Handwritten annotations: A yellow arrow points from the '4' above 'n' to the '4' above 'idx'. A yellow arrow points from the '4' above 'idx' to the '4' above 'idx+1'. A yellow arrow points from the '4' above 'idx' to the '4' above 'n'.



# Skip a character

`str = "Raghav Garg";`  
`s = "Rghv Grg";`

Remove all occurrences of 'a' from a string.

`String s = "xyz";`  
`str = "raghav";`  
`s += "abc";`  
`s = "xyzabc"`

`rchar("", "abc")`  
 ans ← ↓

`rchar("", "bac")`  
 ↓

`rchar("b", "ac")`  
 ↓

`rchar("b", "c")`  
 ↓

`rchar("bc", "")`; → return

`arr → {1, 2, 3, 1, 1, 4, 1, 7}`  
 ↓  
`{2, 3, 4, 7}`

# Subsets

Print all subsets of a string

Print subsets of a string with unique characters.

Follow Up : Do it for array as well

[Leetcode - 78]

string str = "abc";  $\rightarrow$  No. of subsets =  $2^3$

abc  $\rightarrow$  "a", "b", "c", "ab", "ac", "bc", "abc", ""

COLLEGE  
WALLAH

# Subsets

Print subsets of a string with unique characters.

Follow Up : Do it for array as well

[Leetcode - 78]

$\{1, 2, 3\}$

↓

$\{ \}$  ,  $\{1\}$  ,  $\{2\}$  ,  $\{3\}$  ,  $\{1, 2\}$  ,  $\{1, 3\}$  ,  $\{2, 3\}$  ,  $\{1, 2, 3\}$

$\{a, b, c\}$

$\{ \}$  ,  $\{a\}$  ,  $\{b\}$  ,  $\{c\}$  ,  $\{a, b\}$  ,  $\{a, c\}$  ,  $\{b, c\}$  ,  $\{a, b, c\}$

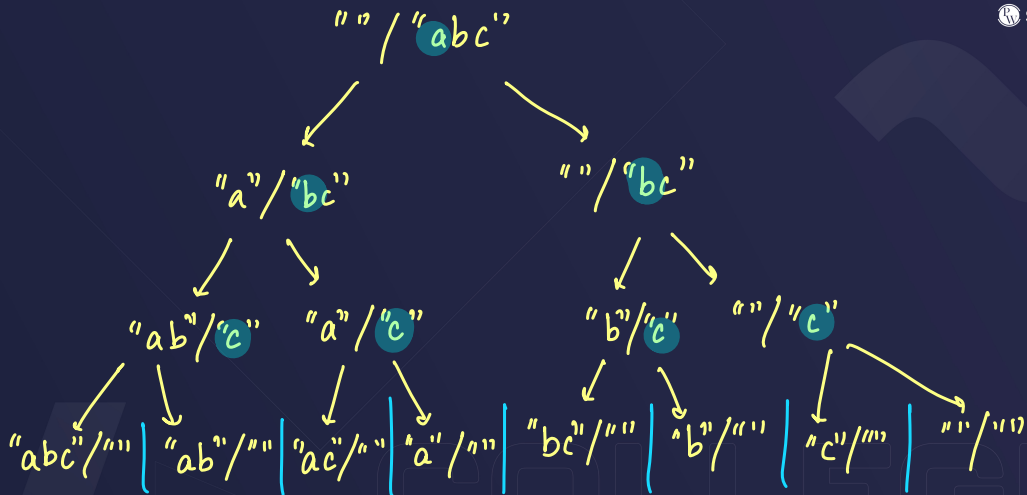
$[1, 2, 3]$

↓

$\{ \{ 3 \}, \{ 1 \}, \{ 2 \}, \{ 3 \}, \{ 1, 2 \}, \{ 1, 3 \}, \{ 2, 3 \}, \{ 1, 2, 3 \} \}$

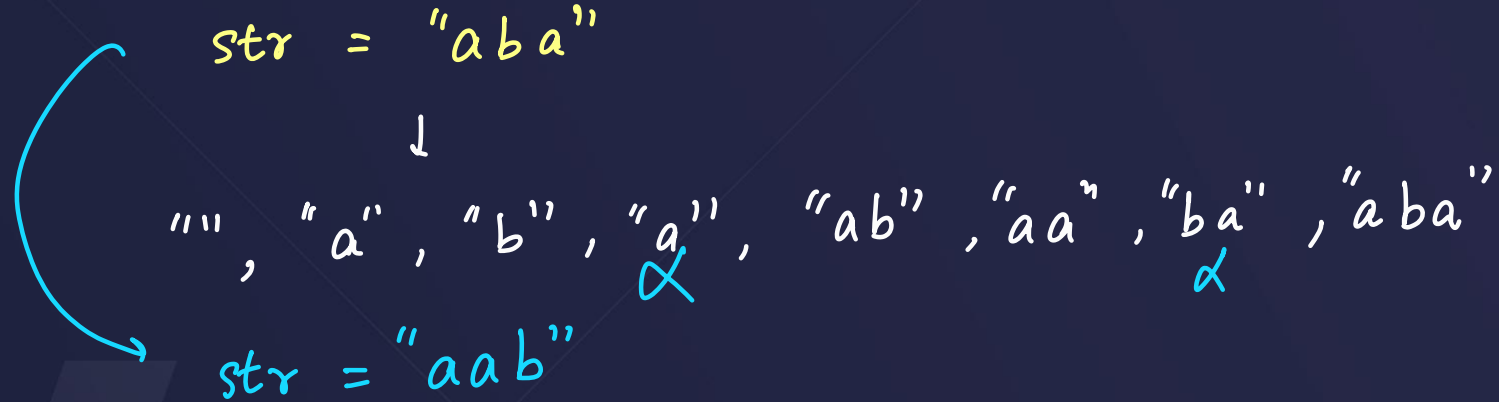
↓

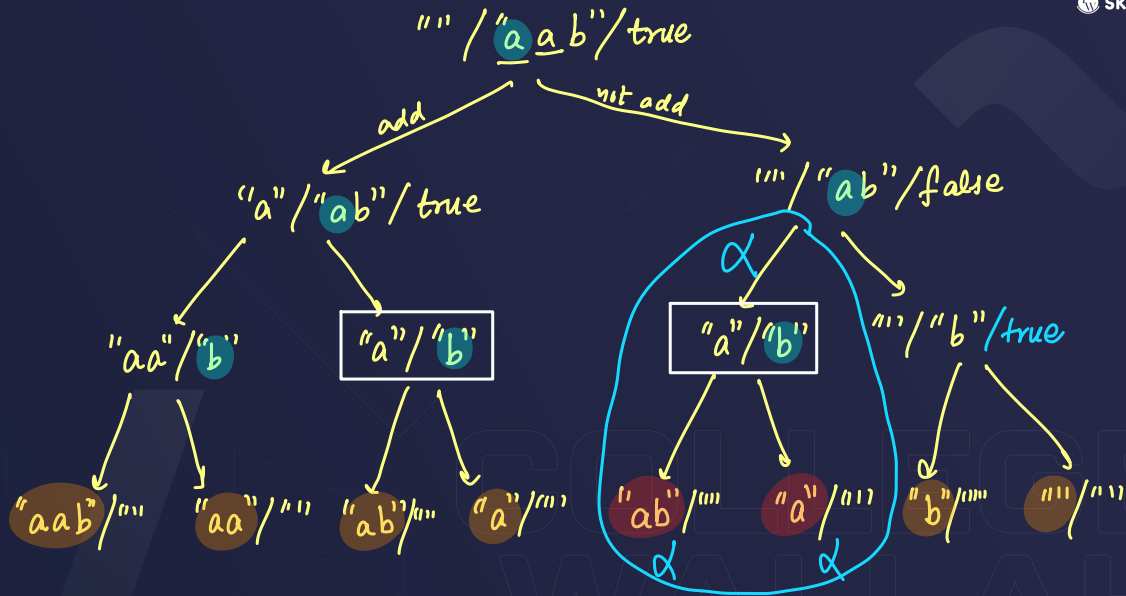
`vector < vector<int>> a;`



# Subsets II

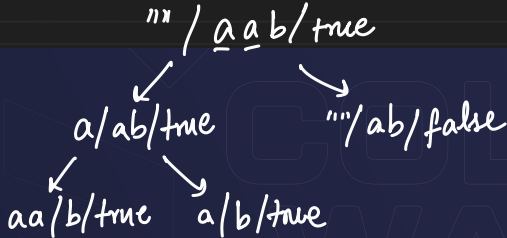
Print subsets of a string containing duplicate characters





```
void storeSubset(string ans, string original, vector<string>& v, bool flag){
    if(original==""){
        v.push_back(ans);
        return;
    }
    char ch = original[0];
    if(original.length()==1){
        if(flag==true) storeSubset(ans, original.substr(1), v, true);
        storeSubset(ans+ch, original.substr(1), v, true);
    }
    char dh = original[1];
    if(ch==dh){
        if(flag==true) storeSubset(ans, original.substr(1), v, true);
        storeSubset(ans+ch, original.substr(1), v, false);
    }
}
```

aab





"a a a b"

↓

a, a, a, b, aa, aa, aa, ab, ab, ab  
 $\alpha$   $\alpha$   $\alpha$   $\alpha$   $\alpha$   $\alpha$   $\alpha$

aaa, aab, aab, aab, aaab, ""  
 $\alpha$   $\alpha$

**Ques:** Print all increasing <sup>sub</sup>sequences of length  $k$  from first  $n$  natural numbers.

$n = 5$  ,  $k = 3$

$k \leq n$

1 2 3 4 5  $\rightarrow$  1 2 3, 2 3 4, 3 4 5  
1 3 5

Subsequence, subset, subarray

$\downarrow$   
Continuous part of array

1 2 3, 1 2 4, 1 2 5, 1 3 4, 1 3 5, 1 4 5  
2 3 4, 2 3 5, 2 4 5, 3 4 5

1 2 3 4

$k=3$

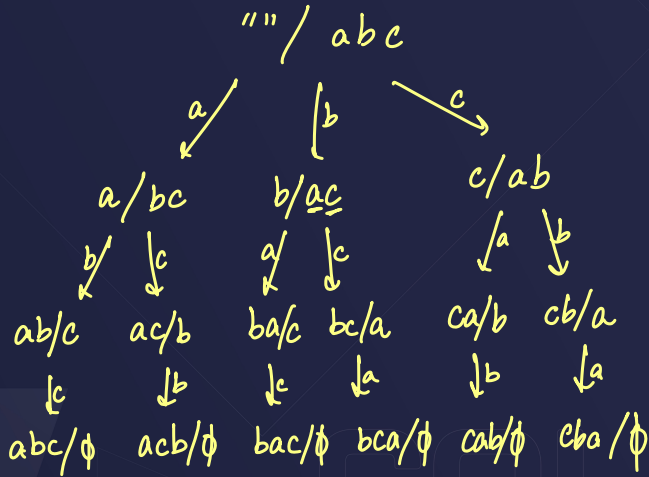
COLLEGE  
WALLAH

# Permutations

→ Bad in terms of T&S

Finding all permutations of a string given all elements of the string are unique.

$abc \rightarrow abc, acb, bac, bca, cab, cba$



""  $\rightarrow \phi$

# Hint: leftsubstr & rightsubstr

# Next Lecture

More **problems** on Recursion

$$LCM = \frac{a \times b}{HCF}$$

COLLEGE  
WALLAH