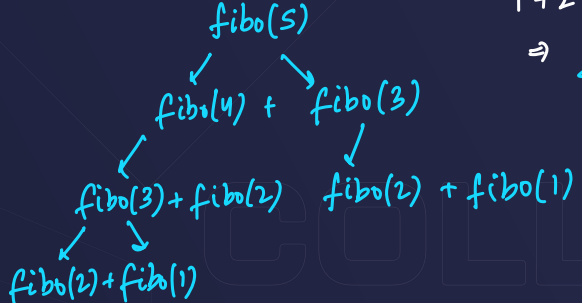
Recursion - 3

Lecture-29

Raghav Garg

Time Complexity of Fibonacci number


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



For 'n'

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

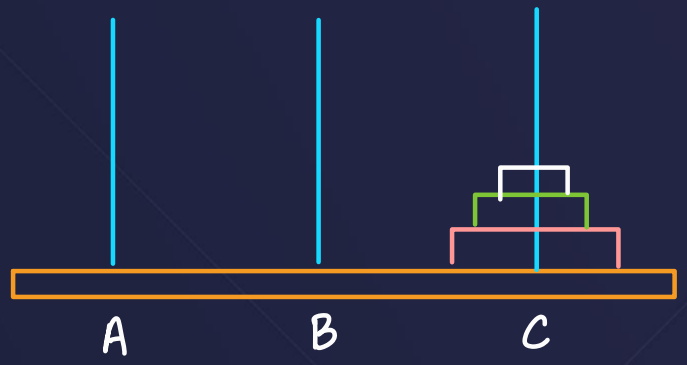
$$1 + 2^1 + 2^2 + 2^3 \cdots 2^{n-1}$$

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

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

Ques : Tower of HANOI

'Maza aa gaya'



Minimum Moves

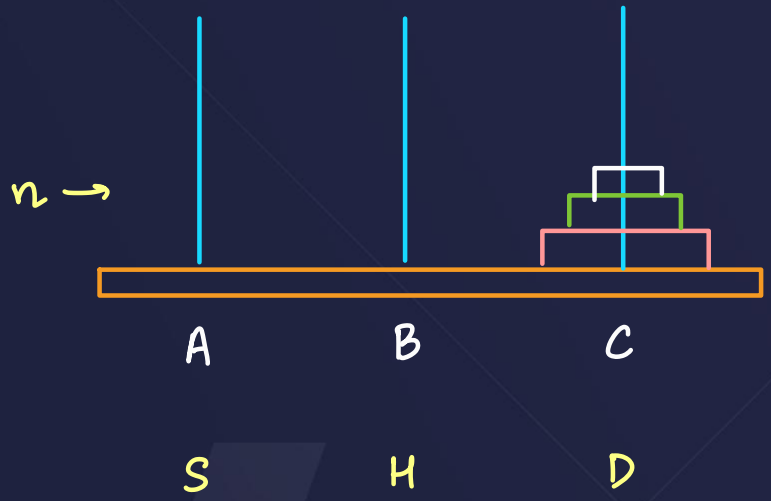
n	$2^n - 1$	
1	$2^1 - 1$	= 1
2	$2^2 - 1$	= 3
3	$2^3 - 1$	= 7
4	$2^4 - 1$	= 15

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 → S → D



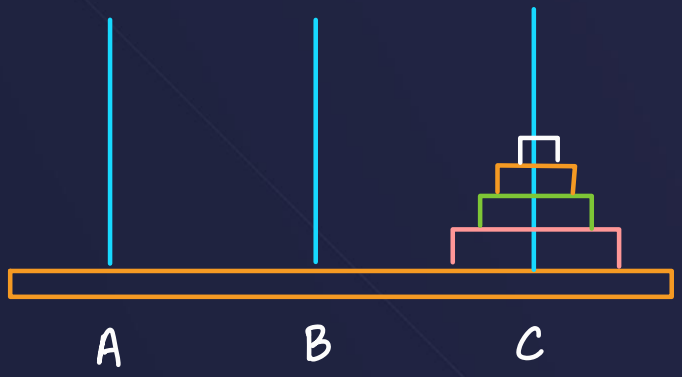
1) n-1 disks : A → B Recursion

2) Biggest disk : A → C → print

3) n-1 disks : B → C Recursion

hanoi(n-1

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);
}
```

```
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] << " ";
}
```

Maximum Value of an array

Find out maximum element of an array using recursion

1) Print

2) Store

{1, 5, 2, 3}

n = 4

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

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

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

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

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

Skip a character

Str = "Raghav Garg";

S = "Rghv Grg";

Remove all occurrences of 'a' from a string.

"xyz"

9

String S = "xyz";

str = "raghav";

S += "abc";

S = "xyz abc"

rchar("", "abac") → original

arrs

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"; → No. of subsets = 2^3

abc → "a","b","c","ab","ac","bc","abc",""

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}

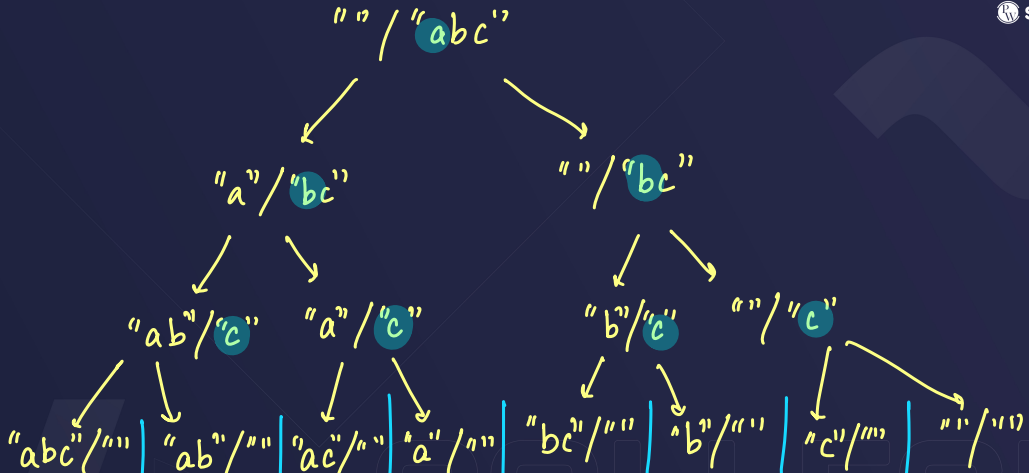
[1, 2, 3]

↓

{ { }, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3} }

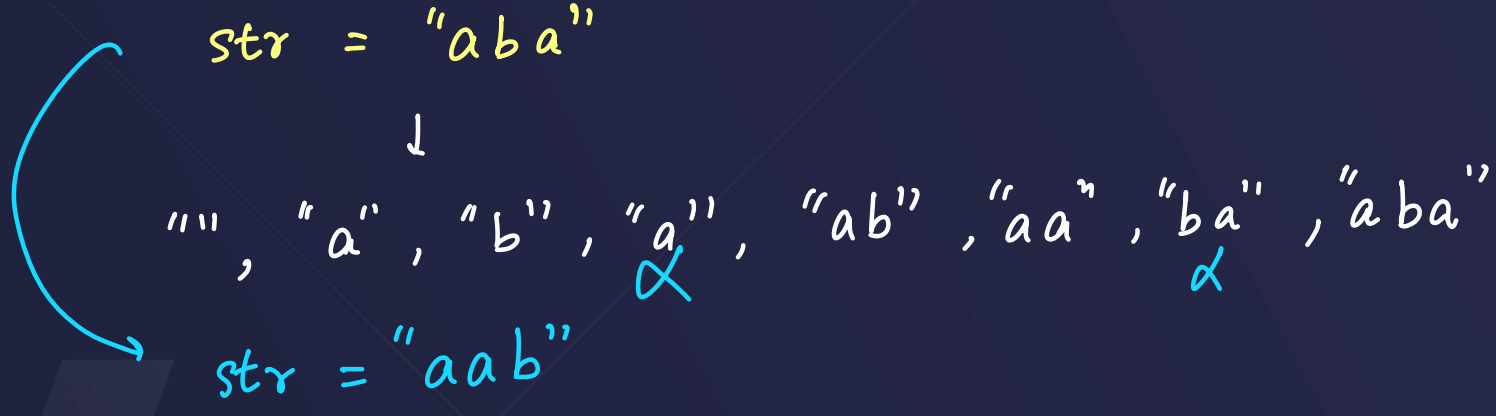
↓

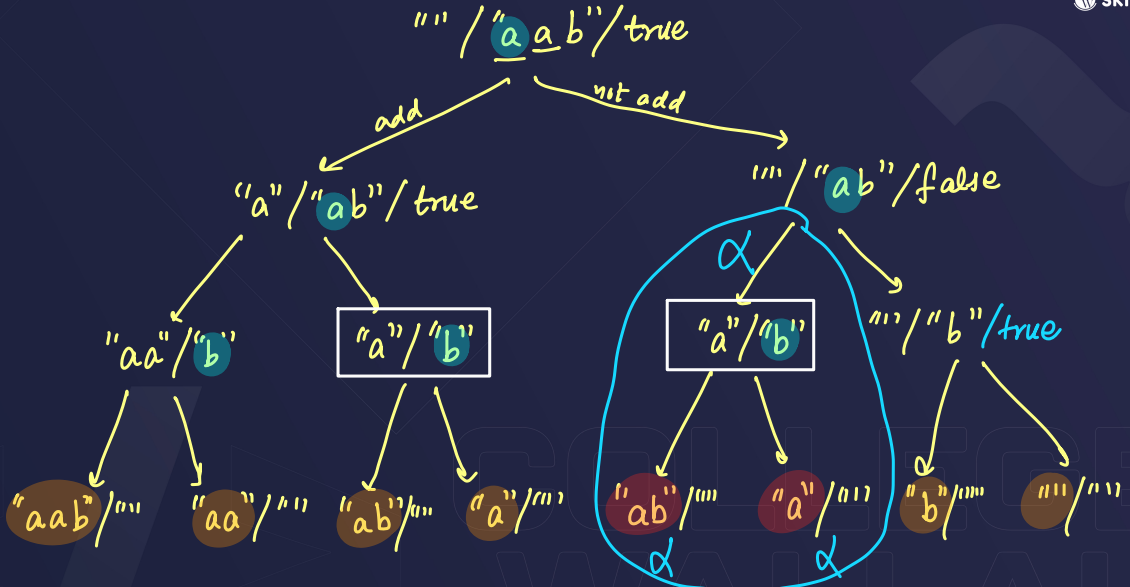
```
vector < vector<int> > a;
```


"" / "abc"
"a" / "bc"
"" / "bc"
"ab" / "c"
"a" / "c"
"b" / "c"
"" / "c"
"abc" / ""
"ab" / ""
"ac" / ""
"a" / ""
"bc" / ""
"b" / ""
"c" / ""
"" / ""

Subsets II

Print subsets of a string containing duplicate characters

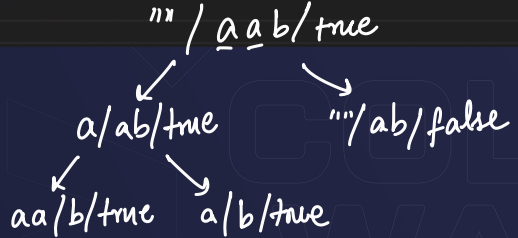



"" / "a a b" / true
add
not add
"a" / "ab" / true
"" / "ab" / false
"aa" / "b"
"a" / "b"
"a" / "b"
"" / "b" / true
"aab" / ""
"aa" / ""
"ab" / ""
"a" / ""
"ab" / ""
"a" / ""
"b" / ""
"" / ""

```
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+ch,original.substr(1),v,true);
        storeSubset(ans+"",original.substr(1),v,true);
    }
    char dh = original[1];
    if(ch==dh){
        if(flag==true) storeSubset(ans+ch,original.substr(1),v,true);
        storeSubset(ans+"",original.substr(1),v,false);
    }
}
```

a ab true

aab



"aaab"

a, a, a, b, aa, aa, aa, ab, ab, ab

aaa, aab, aab, aab, aaab, ""

Ques: Print all increasing sub sequences of length k from first n natural numbers.

$k <= n$

$n = 5$, $K = 3$

1 2 3 4 5 → 123, 234, 345

1 3 5

Subsequence , subset , subarray

↓

continuous part of array

123, 124, 125, 134, 135, 145

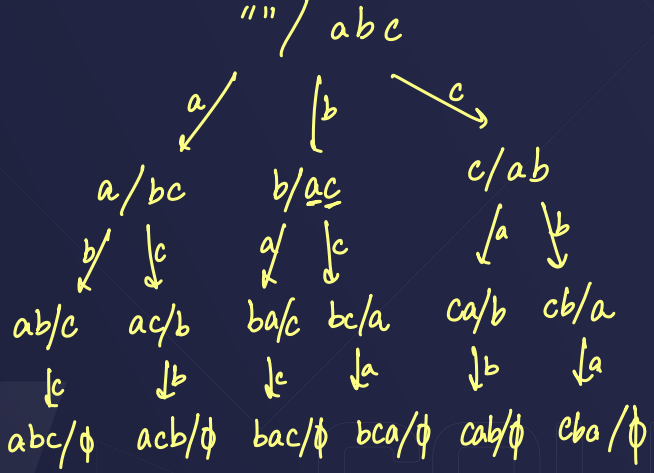
234, 235, 245, 345

1 2 3 4 K = 3

Permutations → Bad in terms of T&S

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

abc → abc, acb, bac, bca, cab, cba



"" → ϕ

Hint: leftsubstr & Rightsubstr

Next Lecture

More problems on Recursion

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