LECTURE - 27 (Recursion)

In recursion we try to solve a bigger problem by finding out solutions to smaller sub problems. We represent there ~~functions~~ problems in the form of functions and these functions calls itself to solve smaller sub problems.

PMI → Principal of Mathematical induction.

Q-1 Prove sum of first N natural no. is $\frac{N*(N+1)}{2}$

Base case → for N==1, ans. will be 1.
Assumption → Let's say formula valid for N=k ↑ $\frac{k*(k+1)}{2}$
self work → 1+2+3+----------+k + k+1

$\frac{k(k+1)}{2} + (k+1) \rightarrow (k+1)\left(\frac{k}{2}+1\right) \rightarrow \frac{(k+1)(k+2)}{2}$

Q-1 Write funct. for n!

Sol:
```
if(n==1)
{f(n)=1}
f(n) = n * f(n-1)
```

→ Base case: n==1 ? f(n)=1
→ assumption: f(n-1)
→ selfwork: Multiply n & f(n-1)

```
int f(int n){
    if(n==1)
        return 1;
    int ans = n * f(n-1);
    return ans;
}
int main(){
    int result = f(5)
    cout << result;
    return 0;
}
```

O(n) [Time and Space complexity]

{Space gets introduced} { due to call stack }

[50:00] Debugging

⇒ Working of Recursive function

- Syntax

```
methodName (N Parameters){
        if (Halting condition){ } Base case
            return result
        }
        return methodName (N parameters)
}
```

smaller sub Problem

Assumption + self work

Q1 Sol

nth fibonacci no.

```
int fib(int n){
    if (n==0 or n==1) return n;
    return fib(n-1) + fib(n-2);
}
int main(){
    int result = fib(5);
    cout << result;
    return 0;
}
```

space → O(n)

Time → $O(2^n)$

FREEMIND

Date

Page

LECTURE-28

(Problems-1 on Recursion)

Q) Given an integer, find out the sum of its digits using recursion.

Ans

⇒ Base case → if(n≥0 and n≤9) return n;

⇒ Self Work's Recursively go and calc. sum of d-1 digits & add the last digit to it.

⇒ Assumption:- $f(n) = f(n/10) + n\%10$

↳ $f(n/10)$ works correctly.

Ans

```
int f(int n){
    if(n≥0 and n≤9) return n;
    return f(n/10) + (n%10);
}
int main(){
    int result = f(1234)
    cout<<result;
} return 0;
}
```

Q) Given two nos. p and q, find the value p^q using a recursive function.

Ans ⇒ # Base case → if(q==0) return 1;

Assumption → $f(p,q) = p * f(p,q-1)$ → return

```
int main(){
    int result = f(3,4)
    cout<<result;
```

O(q)

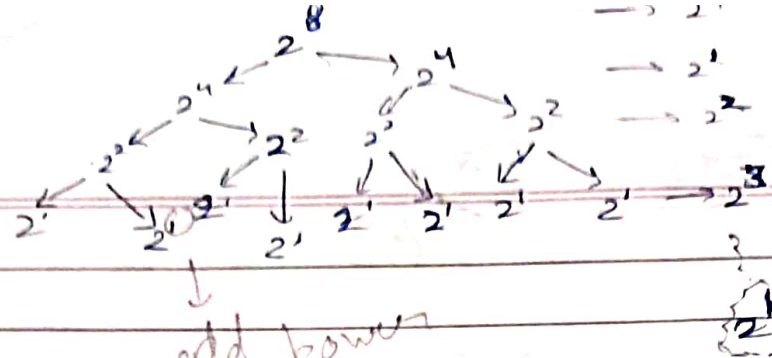
{Time complexity & Space}

FREEMIND

Date

Page

Method 2



$\frac{q}{2^0} \to \frac{q}{2} \to \frac{q}{2^2} \to \frac{q}{2^3} \cdots \cdots \cdots \frac{q}{2^k} \to 1$

$2^{16} \to (2^8)^2 \to (2^4)^2 \to (2^2)^2 \to (2)^2$

$q = 2^k$

$k = \log_2 q$

$f(p,q) \to \left(f\left(p, \frac{q}{2}\right)\right)^2$ $\quad q \to$ even

$f(p,q) \to p * \left(f\left(p, \frac{q}{2}\right)\right)^2$ $\quad q \to$ odd

$O(\log q)$ Time complexity

Code ⇒

```
#include <iostream>
using namespace std;
int f(int p, int q){
    if (q==0) return 1;
    if (q%2==0){
        int result = f(p, q/2);
        return result * result;
    }else{
        int result = f(p, q/2);
        return p * result * result;
    }
int main(){
    int res = f(5, 6);
    cout << res;
}
```

Aakhri pe add? to ban hi jayega

FREEMIND

Date

Page

Lecture 29

(Problem on Recursion-2)

→ Recursion on arrays
→ Recursion on strings

f(arr, idx, n) ---→ length of array
↳ current idx we are pointing to in the array
given array

Q) Given an array, print all the elements of the array recursively.

A **Base case** → if array is empty print nothing

Assumption → f(arr, idx+1, n)

Self Work → print the element at the current index

Before you go to print remaining array, I can print myself

f(arr, idx, n) → print(arr[idx]) →
↓
f(arr, idx+1, n) → assume that we can correctly print the remaining array using f

#Code

```
void f(int *arr, int idx, int n){
    if(idx == n) return; // base case
    cout << arr[idx] << "\n"; // self Work
    f(arr, idx+1, n); // assume it work properly
}
int main(){
    int n=5;
    int arr[] = {6,1,9,3,4}
    f(arr, 0, n);
    return 0;
}
```

O(n) → Time
→ Space

FREEMIND

Date

Page

Q> Print the max value of the array [3, 10, 3, 2, 5].

Base case:- If array has only one element that is the max.
→ (idx == n-1) → return arr[idx]

Assumption :- we assume → f works correctly for f(arr, idx+1, n)
i.e. it successfully finds out max in the remaining array.

Self Work :- Compare the max. of remaining array with current array.

f(arr, idx, n) = max(arr[idx], f(arr, idx+1, n))

Code

```
int f(int *arr, int idx, int n)
{
    if (idx == n-1)
        return arr[idx];          // Base Case
    return max(arr[idx], f(arr, idx+1, n));   // Self Work + Assumption
}
int main() {
    int arr[5] = {3, 10, 3, 2, 5};
    int n = 5;
    cout << f(arr, 0, n);
    return 0;
}
```

Q> Find the sum of the values of the array [2, 3, 5, 20]

Base case, Assumption, self work same as prev. ques. (max → sum)

```
int f(int * arr, int idx, int n)
    if (idx == n-1) {
        return arr[idx];
    } return [arr idx + f(arr, idx+1, n)];
}
int main() {
    int arr[] = {2,3,5,20,1};
    int n = 5;
    cout << f(arr, 0, n);
    return 0;
}
```

LECTURE -30

(Recursion Problems - 3)

Q: Remove all occurrences of 'a' from strings s = "abcax".

Iteratively.

```
result = " "
for (i=0, i<n; i++) {
    if (S[i] != a) {
        result += S[i]
    }
}
```

Recursively

Base case → empty string (no char. remains)
↳ return empty string

assumption → f(arr, idx+1, n) → gives a string without 'a'.

Self work → if my current char is not 'a' append it with f(arr, idx+1, n)
else don't.

Note
S: Substring :-
Phle element ko remove karke uske sub part p work krta hai or aise hi last element tak work krta hai

FREEMIND

Date

Page

$f(s, idx, n) = ((s[idx] == 'a')\ ?\ "" : s[idx]) + f(s, idx+1, n)$

Code

```
string f(string &s, int idx, int n) {
    if (idx == n) return "";
    string curr = "";
    curr += s[idx];
    return ((s[idx] == 'a' ? "" : curr) + f(s, idx+1, n));
}
int main() {
    string s = "abcax";
    int n = 5;
    cout << f(s, 0, n);
    return 0;
}
```

Q) Write a Program to check whether a given no. is palindrome or not.

$f(num, *temp) = f(num/10, temp)$ and $(num\%10 == temp\%10)$

$*temp = *temp/10$

? → true/false result

```
bool f(int num, int *temp) {
    if (num >= 0 and num <= 9) { lastdigitoftemp = (*temp)%10;
        (*temp) /= 10;
        return (num == lastdigitoftemp);
    }bool result = (f(num/10, temp) and (num%10) == ((*temp)%10));
    (*temp)/=10;
    return result;
}
int main() {
```

1261 → 1261
126 → 1261
12 → 1261
(1) → 1261

Date

Page

```
int main(){
    int num = 12621;
    int another num = num;
    int *temp = &anothernum;
    cout<<f(num, temp);
    return 0;
}
```

LECTURE-31

(Recursion Problem-4)

Q→ Given a no. n. Find the increasing sequence from 1 to n without using any loop.

Constraint: $0 < n < 1e6$

input: 4

output: 1234

$f(n) = f(n-1) \longrightarrow print(n)$

Base case → if(n<1) { return; }

Assumption → f(n-1) gives no. from 1 to n-1

Self Work → print(n) in f(n)

```
Void f(int n){
    if(n<1) return; // base case
    f(n-1)  // go and print first n-1 natural no. → assumption
    cout<<n<<" ";
}
int main(){
    f(20)
    return 0;
}
```

Date
Page

Q1 Given a number num and a value K. Print K multiples of num.
constraints: K>0
input: num = 12, K = 5
output: 12, 24, 36, 48, 60

Ans) num = 3, K = 5 → 3, 6 9 12 15

f(num, K) = f(num, K-1) → print(num*K)

Print first K multiples of n — assumption — selfwork

Base case (K==0) → we do nothing & just return

Code

```
Void f(int num, int k)
    if (K==0) return; // Base case
    f(num, k-1); //assumption → correctly prints K-1 terms
    Cout << num*k << " "; //Self Work
}
int main(){
    f(3,5)
}
```

Q2 Given a no. n. Find the sum of natural no.s till n but with alternate signs.
Ex:- if n=5 → return 1-2+3-4+5 = 3
constraints: 0 <= n <= 10^6
input: n=16
output -5

logic →

f(n) = f(n-1) + ((n%2==0)?(-n):n)

Base case (n→0) → return 0

FREEMIND

Date

Page

Ans> int f(int n){
if(n==0) return 0; //base case
return f(n-1) + ((n%2==0)?(-n):n); //assumption
//Calculate sum of 1st (n-1) natural no. with alternate signs, and we add n acc. to sign.
}

int main(){
cout<<f(10)<<"\n";
return 0;
}

LECTURE-32

(Recursion Problem - 5)

Q> Given two no.s x and y. Find the greatest common divisor of x and y using recursion. gcd/hcf

constraints: 0<=x,y<=1e6

input: x=4, y=9
output: 1

input: x=12, y=20
output: 4

12 → 2×2×3
20 → 2×2×5

Euclid's Algo.

If we sub. a smaller no. from a larger one. (we can reduce the large no.) but the gcd will not change.

Eg. gcd → 72, 54 → G
54, 72-54 → G
↳ 18

54 = 2×3×3×3
72 = 2×2×2×3×3
GCD ↳ 18

OR

54, 072%54
↳ 18

54 = 2×3×3×3
18 = 2×3×3
GCD ↳ 18

FREEMIND

Date

Page

If b > a

$gcd(a, b) = gcd(a, b-a) = gcd(a, b\%a)$

(A) gcd(a,b) is Gr

$a = bq + r$ (If (a>b) (a+b))

↳ $a - bq = r$

b | a | q

1

r

⇒ We can say that Gr completely divides a and b. / bq

(a - bq) is completely divisible by Gr.

```
int gcd(int a, int b){
    if(b>a) return gcd(b,a);
    if(b==0) return a;          // base case
    return gcd(b, a%b);
}
int main() {
    int x = gcd(40,8)
    cout << x << "\n";
    return 0;
}
```

40, 8

8, 32

(32, 8)

8, 24

8, 16

8, 8

(8, 0)

GCD = 8

Q) Given a no. n. Print if it is an armstrong no. or not.

Logic $f(n,d) = pow(n\%10, d) + f(n/10, d)$

f → Sum of digits of n, raised to power d.

d → no. of digits in the original No.

pow(n%10, d) → Self Work

f(n/10, d) → Assumption

1) Pow funct.

2) Armstrong Functi.

3) int main

- n → input
- temp = n
- no. of digits
- Funct. call
- result = n

FREEMIND

Date

Page

```
// Power Function.
int pow_recursive(int p, int q) {
    if (q == 0) return 1;
    if (q % 2 == 0) {
        int result = pow_recursive(p, q/2);
        return result * result;
    } else {
        int result = pow_recursive(p, (q-1)/2);
        return p * result * result;
    }
}
int f(int n, int d) {
    if (n == 0) return 0;
    return pow_recursive(n % 10, d) + f(n/10, d);
}
int main() {
    int n;
    cin >> n
    int noofdigits = 0;
    int temp = n;
    while (temp > 0) {
        temp = temp / 10;
        noofdigits++;
    }
    int result = f(n, noofdigits);
    if (result == n) {
        cout << "Yess";
    } else {
        cout << "No";
    }
    return 0;
}
```

FREEMIND

Date

Page

Q1 There are N stones numbered 1, 2, ..., N. For each i ($1 \le i \le N$), the height of stone i is h_i. There is a frog who is initially on stone 1. He will repeat the following action some no. of times to reach stone N:

If the frog is currently on stone i, jump to stone i+1 or stone i+2. Here a cost of $|h_i - h_j|$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches stone N.

input n = 4 (starting, Jump, landing)

arr[] = 10 30 40 20

output = 30

logic

$$f(h, n, i) = \min \begin{cases} f(h, n, i+1) + |h_i - h_{i+1}| \\ f(h, n, i+2) + |h_i - h_{i+2}| \end{cases}$$

$f(h, n, i)$: min. cost to reach n^{th} stone from i^{th} stone

#base case ⟶ i == last_stone
i == second_last_stone

#Code

```
int f(int *h, int n, int i){
    if (i = n-1) return 0;
    if (i = n-2) return f(h, n, i+1) + abs(h[i] - h[i+1]);

    return min( f(h, n, i+1) + abs(h[i] - h[i+1]), f(h, n, i+2)
                + abs(h[i] - h[i+2]) );
}
```

FREEMIND

Date

Page

```
int main(){

    int arr[] = {10,30,40,20}
    int n = 4;
    cout << f(arr, n, 0);

    return 0;
}
```

Lecture - 33

(Recursion Problem - 6)

Q⇒ Given an array of n integers and a target value x. Print whether x exists in the array or not.

constraints:- $0 <= n <= 1e6$, $-1e8 <= x <= 1e8$, $-1e8 <= a[i] <= 1e8$

input 1 ⇒ n = 8, x = 14, array = [4,12,54,14,3,8,6,1]

output 1 ⇒ Yes

input 2 ⇒ n = 1, x = 9, array = [2]

output 2 ⇒ No

Sol ⇒

$f(arr, n, i, x) = (arr[i] == x) \;||\; f(arr, n, i+1, x)$ (or)

this function returns whether element x, is present in the array (arr) from index i to n-1 or not

this check detects whether the element is present on i^{th} index or not.

step of assumption

// base case ⇒ if (i == n) return false

FREEMIND

Date

Page

Code

```
#include<iostream>
using namespace std;
// f represents whether x is present in the
   range [i, n-1] or not?
bool f(arr, n, i, x){
    //base case
    if i(i==n){
        // arrray is exhausted
        return false
    }
    return arr[i]==x || f(arr, n, i+1, x)
}

int main(){
    int arr[] = {5, 6, 7, 9, 15, -2, 10, 11, 4}
    int n = 9
    int x = 15
    bool result = f(arr, n, 0, x)
    if (result){
            cout<<"Yess";
    }
    else{
        cout<<"No";
    }
    return 0;
}
```

Q5 Given an array of integers, print sums of all subsets in it. Output sums can be printed in any order.

FREEMIND

Date

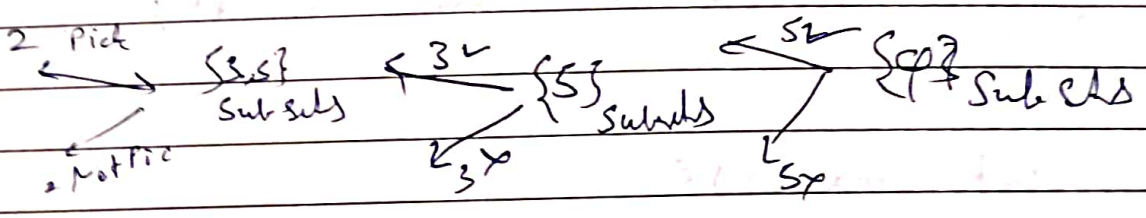
Page

input :- arr[] = {2,3}
output :- 0, 2, 3 5

→ How many total subsets exist for a set of length n.
is 2^n

→ Every element has two choices → Pick / Not Pick

→ let arr = {2, 3, 5}



- assumption ⇒ Aage wale elements k subsets mil jayenge
- Self work ⇒ Current Element ko 2 choices k acc. add krna hai or subsets banane hai
- base case ⇒ Last me null element or last nth elem

f(arr, n, idx, sum, result) = f(arr, n, idx+1, sum+arr[idx], result)
f(arr, n, idx+1, sum, result)

{
print sum of all subsets
[idx, n-1]

//base case if (idx == n) push back(sum);
result, print(sum)
return;

```
#include <iostream>
#include <vector>
using namespace std;
```

FREEMIND

Date

Page

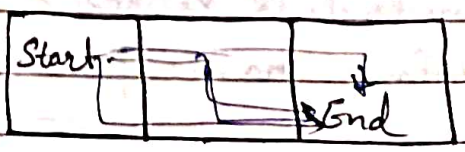
```
Void f(int *arr, int n, int i, int sum, Vector<int> & result)

    if (i == n)
        result.push_back(sum);
        return;
    }
    f(int *arr, int n, int
    f(arr, n, i+1, sum + arr[i], result); //Pick the ith elmt
    f(arr, n, i+1, sum, result); //Don't Pick the ith elmt

}
int main(){
        int arr[] = {2, 4, 5};
        int n = 3;
        Vector<int> result;
        f(arr, n, 0, 0, result);
        for(i=0; i< result.size; i++){
            cout<< result[i] <<" ";
        }
        return 0;
    }
```

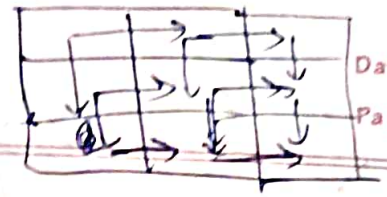
Q:- The problem is to count all the possible paths on an m x n grid from top left (grid[0][0]) to bottom right (grid[m-1][n-1]).

Having constraints that from each cell you can either move only to right or down.

→ input :- m = 2, n = 3

→ output :- 3





FREEMIND

Date

Page

$f(i,j,m,n) = f(i,j+1,m,n) + f(i+1,j,m,n)$

This returns no. of ways to reach m-1, n-1 from i, j if we can only move right & down

we assume func. f works correctly for i, j+1. i.e it correctly gives no. of ways to reach bottom right from right cell

No. of Ways to reach B-R from down cell

// base case

```
if (i == m-1 && j == n-1)
    return 1
if (i >= m || j >= m)
    return 0;
```

#Code

```
#include <iostream>
using namespace std;

int f(int i, int j, int m, int n) {

    if (i == m-1 && j == n-1)
        return 1;
    if (i >= m || j >= n)
        return 0;
    return f(i, j+1, m, n) + f(i+1, j, m, n);
}
int main() {
    cout << f(0, 0, 2, 3);
    return 0;
}
```

FREEMIND

Date

Page

Lecture - 34

(Recursion Problem - 7)

Q: Given a string we have to find out all its subsequences of it. A string is a subsequence of a given string, that is generated by deleting some character of given string without changing its order.

input :- abc
output :- a, b, c, ab, bc, ac, abc

Sol: n → length 2^n → subsequences

⇒ Same as previous Subsets Question there are two choices Pick / Not Pick.

$f(str, i, result, li) = f(str, i+1, result + str[i], li)$
(result : string)

$f(str, i+1, result, li)$

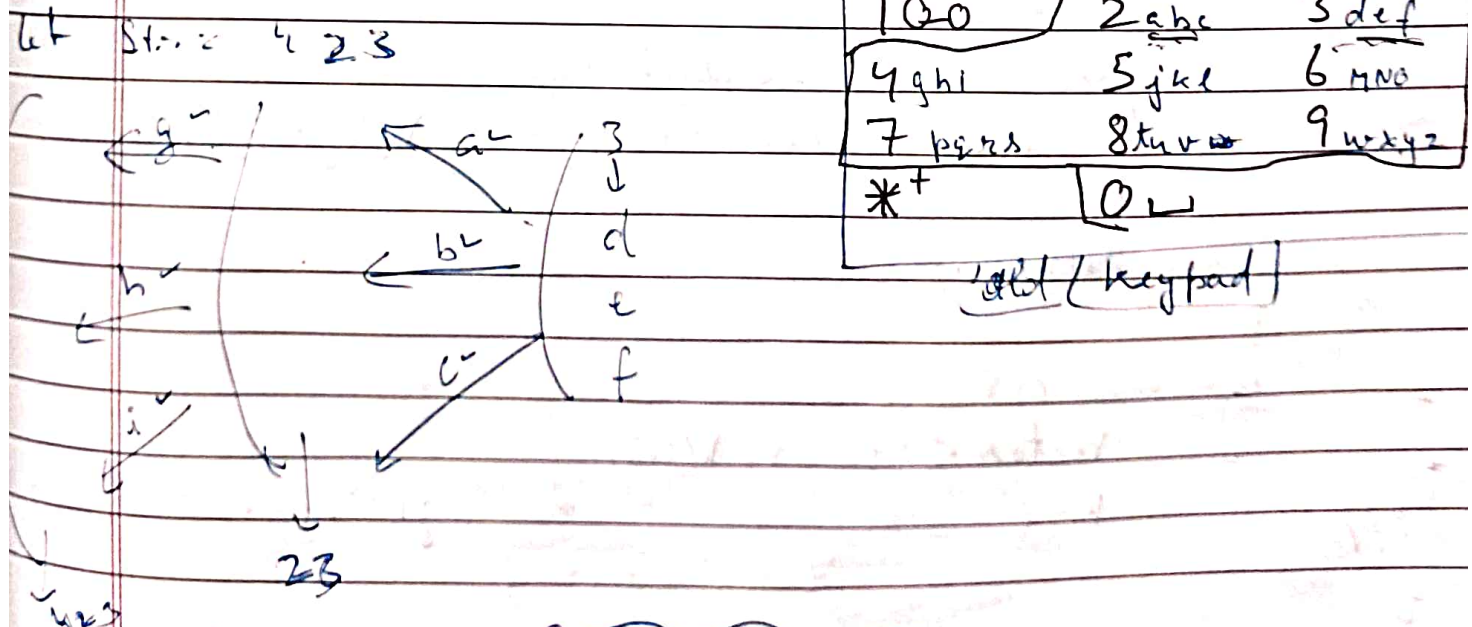
#Code

```
#include <iostream>
include < vector>
using namespace std;

Void f (string &str, int i, string result, Vector <string &li);

    if (i == str.length()) {
        li.push_back(result);
        return;
    }
```

```
    f(str, i+1, result + str[i], li);
    f(str, i+1, result, li);
}

int main() {

    Vector<string> res;
    string str = "abc";
    f(str, 0, "", res);
    for (i=0; i<res.size(); i++) {
        cout << res[i] << " ";
    }
    return 0
}
```

Q→ Given a string containing digits from 2-9 inclusive, retur all possible letter combinations that the number could represent. Return the answer in any order.

input :- digits = '23'
output :- ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"



FREEMIND

Date

Page

f(str, i, result, li) = f(str, i+1, result + comb[str[i]], li)

↓

Creates all string combinations (i, n-1)

Code

```
#include <iostream>
#include <vector>
using namespace std;

void f(string &str, int i, string result, vector<string> &li,
                                    vector<string> &v) {
    if(i == str.size()) {
        li.push_back(result);
        return;
    }
    int digit = str[i] - '0';
    if (digit <= 1) {
        f(str, i+1, result, li, v);
        return;
    }
    for( j=0; j < digit.size(); j++) {
        f(str, i+1, result + v[digit][j], li, v);
    }
    return;
}
int main() {
    Vector<string> v(10);
    v={" ", " ", "abc", "def", "ghi", jkl, mno, pqrs
        "tuv", "wxyz"}
```

FREEMIND

Date

Page

```
string str = '23';
Vector<string> li;
f(str, 0, " ", li, v);
for(i=0; i< li.size(), i++) {
    cout<< li[i]<<" ";
}
return c;
}
```