Session-5 [Oct 07 2023]

DOMS | Page No. | Date / /

Recursion:- It's a programming paradigm which will solve a bigger problem by solving smaller instances of the exact same problem.

Step's to solve recursive problems:-

1) Assumption
2) Main logic
3) Base condition

```
int multiply (int N)        Assumption
    if (N==1) return 1;     base condition
    return N * multiply(N-1) Main logic
```

Q) Print n^{th} fibanoci series:-

```
int fib(int n){                    Assumption
    if(n==1 || n==2) return 1;     base condition
    return fib(n-1) + fib(n-2);    main logic
}
```

Q) $a, a+d, a+2d, a+3d, \ldots\ldots a+(n-1)d$

```
int sumAP(int a, int d, int n)
    if(n==1) return a;

    return a+(n-1)d + sumAp(a, d, n-1)
```

For iterative codes we count # of iterations for recursive codes we calculate using recurrance relation.
So to calculate TC of ~~fib~~ multiply series we need to calculate

Time $(N) = T(N) \Rightarrow T(N) = 1 + T(N-1)$
(1 → base, main logic)

$T(1) = 1$ base logic

$T(N) = T(N-1)+1 = T(N-2)+2 = T(N-3)+3 \ldots\ldots T(N-K)+K$

$\Rightarrow T(N) = T(N-(N-1)) + N-1 = T(1) + N - 1 = N$

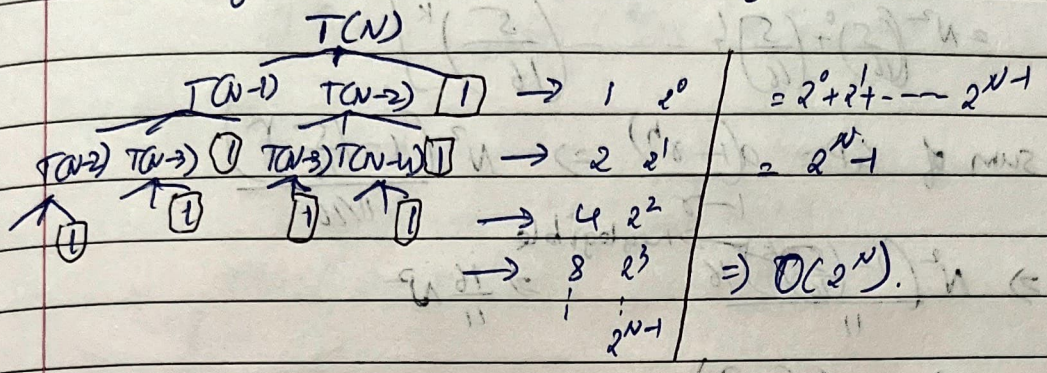
$\therefore T(N) = O(N)$.

For fib series $T(N) = 1 + T(N-1) + T(N-2)$

$= 1 + (1 + T(N-2) + T(N-3)) + (1 + T(N-3) + T(N-4))$

$= 5 + T(N-2) + 2T(N-3) + T(N-4)$

We cant solve this using reccurance relation, as the number of terms are increasing.



This is called Induction methode. If number of term's are increasing we use Induction method.

Solve these :- for all these $T(0) = 1$

1. $T(N) = 2T(N/2) + N = 2T(N/4) + N + N/2 + 4T(N/4) = 2N + 4T(N/4) = 3N + 8T(N/8) = KN + 2^K T(N/2^K)$

   Let $2^K = N \Rightarrow K = \log N$.

   $= N\log N + 1$

   $\Rightarrow T(N) = O(N \log N)$.

2. $T(N) = 2T(N/2) + 1 = O(N)$
3. $T(N) = T(N/2) + N = O(N)$
4. $T(N) = T(N/2) + 1 = O(\log N)$
5. $T(N) = T(N/2) + T(N/4) + N^2 = O(N^2)$
6. $T(N) = 2T(N/2) + N = O(N \log N)$

$T(N) = T(N/2) + N = N + N/2 + T(N/4) = \frac{3N}{2} + T(N/4) = \frac{8N}{2} + \frac{N}{4} + T(N/8)$

$= \frac{7N}{4} + T(N/8) = \frac{(2^K - 1)N}{2^{K-1}} + T(N/2^K)$

$2^K = N \Rightarrow K = \log N$.

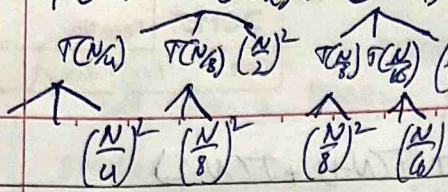
$= \frac{2^{\log N} - 1}{2^{\log N - 1}} \times N = 2 \times N - 1 \times N = O(N)$

4. $T(N) = T(N/2) + 1 = T(N/4) + 2 = T(N/8) + 3 = T(N/2^K) + K$

   $2^K = N \Rightarrow \log N = K$

   $\Rightarrow T(N) = \log N$.

⑤ $T(N) = T(N/2) + T(N/4) + N^2$



$N^2 + \frac{N^2}{4} + \frac{N^2}{16} = \frac{5N^2}{16}$

$\frac{N^2}{16} + \frac{N^2}{32} + \frac{N^2}{16^2} = \left(\frac{N}{4} + \frac{N}{16}\right)^2 = \frac{25N^2}{256}$

$\Rightarrow T(N) = N^2 + \left(\frac{N}{2}\right)^2 + \left(\frac{N}{4}\right)^2 + \left(\frac{N}{4}\right)^2 + 2\cdot\left(\frac{N}{8}\right)^2 + \left(\frac{N}{16}\right)^2 + \cdots$

$= N^2 + \frac{5}{16}N^2 + \left(\frac{5}{16}\right)^2 N^2 + \cdots + \left(\frac{5}{16}\right)^K N^2$

$= N^2\left[\left(\frac{5}{16}\right)^0 + \left(\frac{5}{16}\right)^1 + \cdots + \left(\frac{5}{16}\right)^K\right]$

sum of GP $= \frac{a(1-r^n)}{1-r} \Rightarrow \frac{N^2\left(1-\left(\frac{5}{16}\right)^K\right)}{11/16}$

$\Rightarrow \frac{N^2\left(1-\left(\frac{5}{16}\right)^K\right)16}{11}$ ($\left(\frac{5}{16}\right)^K$ negligible) $\Rightarrow \frac{16}{11}N^2$

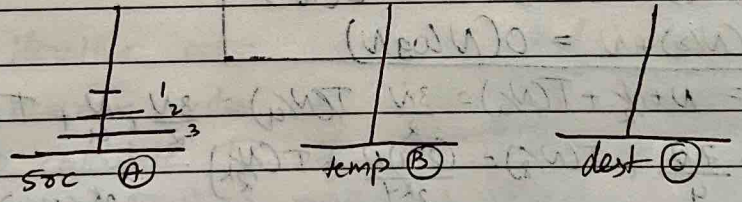
$\therefore T(N) = O(N^2)$

⑥ Towers of Hanoi :-

Move all the disks in source tower to destination tower with help of temporary tower following the constraints.

Constraints :- ① Move one disk at a time

② Never place bigger disk on a smaller disk.



Steps :- 1: A → C
2: A → B
1: C → B
} Src → temp via dest

3: A → C Src → dest

1: B → A
2: B → C
1: A → C
} temp → dest via source

Lets consider there are N disks on src

DOMS | Page No.
Date / /

Towers of Hanoi

Step1) in this step we need to move N-1 disks to temp & N^th to dest. Then move temp to N-1 disk to dest.

My code

```
void TOH (int N, char Src, char temp, char dest) {
    if (N==1) print ("Src" + "to" + dest).
    else {
    TOH (int N-1, Src, dest, temp)
    print (Src + "to" + ~~temp~~ dest)
    TOH (int N-1, temp, src, dest) }
}
```

```
void towers_of_Hanoi (int N, char src, char dest, char temp) {
    ① if (N==0) return;                          T(0)=1
    ② towers_of_Hanoi(N-1, src, temp, dest)      T(N-1)
    ③ print (N: src → temp)                      1
    ④ towers_of_Hanoi(N-1, temp, dest, src)      T(N-1)
}
```

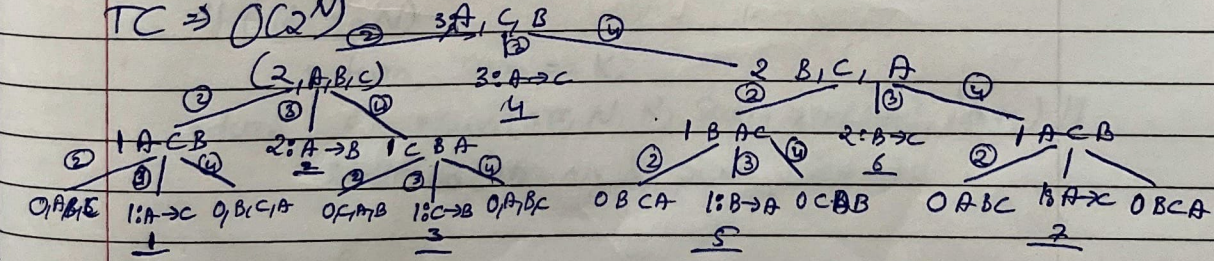
$T(N) = 2T(N-1)+1 = 2[2T(N-2)+1]+1 = 4T(N-2)+3$

$= 8T(N-3)+7 = 2^K T(N-K) + (2^K-1)$

Let $N-K=0 \Rightarrow N=K$ $T(N-K) = T(0) = 1$

$= 2^K + 2^K - 1 = 2^{K+1} - 1 = 2^{N+1} - 1 \Rightarrow T(N) = 2^{N+1} - 1$

$TC \Rightarrow O(2^N)$



DOMS | Page No.
Date / /

a power b

Q) Solve a^N using recursion.

```
int pow(int a, int N){
    if (N==0) return 1;
    if (N%2==0)
        return (pow(a, N/2) * pow(a, N/2)) % mod
    else
        return ((pow(a, N/2) * pow(a, N/2)) % mod * a) % mod
}
```

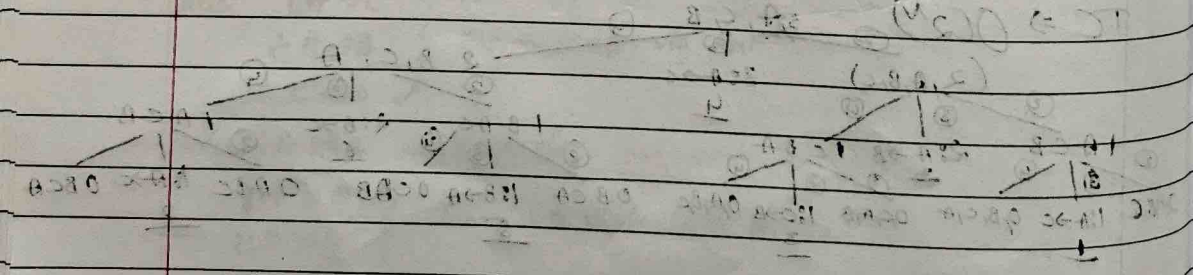
let N=10 ⇒ N/2 = 5 $a^{10} = a^5 * a^5$ if N is even

N=11 ⇒ N/2 = 5 $a^{11} = a^5 * a^5 * a$ if N is odd.

TC for above code is O(n) ∵ $T(N) = 2T(N/2) + 1$

```
int pow(int a, int N)
    if (N==0) return 1;
    x = pow(a, N/2)
    if (N%2==0)
        return (x * x) % mod
    return ((x * x) % mod * a) % mod.
```

TC for above code is O(logn) ∵ $T(N) = T(N/2) + 1$



Session 6 (8 Oct 2023)

DOMS | Page No. | Date / /

Backtracking. Sub Set Sum

Q) Given an array find if there is a subset whose sum is K. (Already done using bit manipulation) Generated all possible subsets & checked sum.

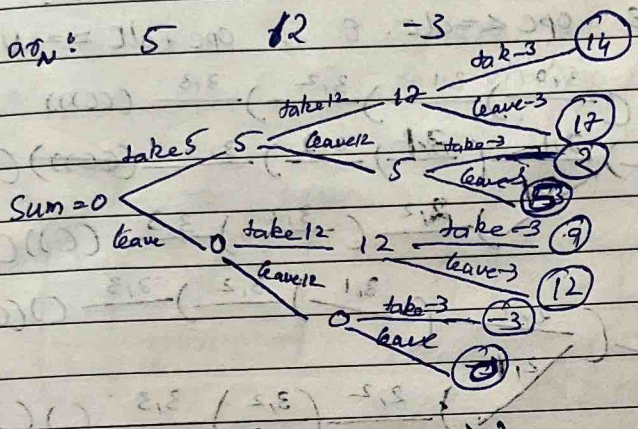
Sample input:- 5 12 -8 8 4 -2 1 K=16.
{12, 4}, {5, 6, 4, 1}

Solution. In a array of size N we will have 2^N subsets

Ex [0, 2] → [0], [2], [0, 2], ∅ so $4 = 2^2 \Rightarrow 2^N$

take or leave → take or leave
2 chances 2 chances ⇒ 2*2 = 4.

process arr: 5 12 -3



① figure out all the required parameters
② Recursion.

```
bool sumK(sum, i, arr, K)
    if K==0 return true
    return ((sum+arr[i], i++, arr, K) || (sum, i++, arr, K))
    sumK(0, 0, arr, K)
```

Code written while explaining logic, not checking if Sum==K so not correct.

if(K==sum) return true
else if(i<N-1) false
else false

```
bool subsetSum(arr, N, K, sum, idx){
    if (idx == N)
        return sum == K;
    return subsetSum(arr, N, K, sum + arr[idx], idx++) ||
           subsetSum(arr, N, K, sum, idx++)
}
int main(){
    Read()
    cout<<subsetSum(arr, N, K, 0, 0)<<endl;
return 0.
```

Paranthesis checking & printing

DOMS | Page No. | Date / /

Q) Given N, print valid paranthesis of length N in lexicographic order.

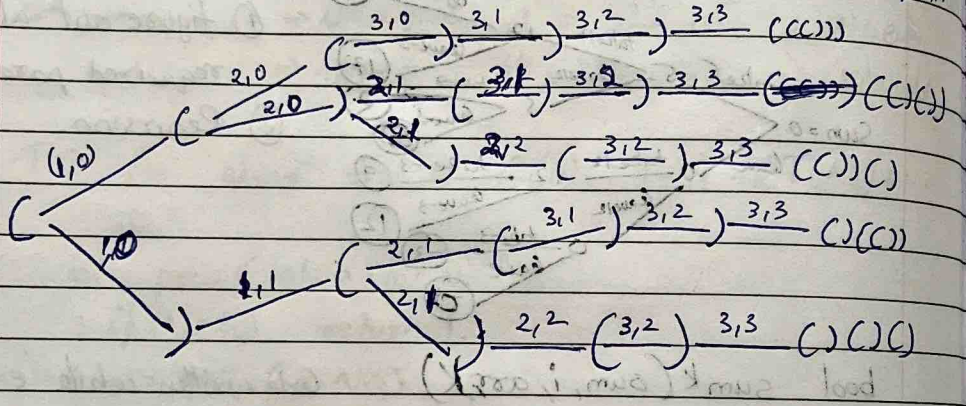
process) Let N=4 ((())) ; (()()

N=6 ; ((())), (()()), (())(), ()(()), ()()()

We have two way's < generate all & check for validity / generate only valid

Example for generating for N=6. for every branch write open count opc, closed count clc as (opc, clc).

opc & clc <= n/2 & opc <= clc. & if opc+clc == N print



idx

void validparanthesis (N, op, cl, char ar[]) {

if (idx == N) { print(arr); return; }

if (op < N/2) {

ar[idx] = '(';

paranthesis (N, op+1, cl, ar, idx+1); }

if (cl < op) {

ar[idx] = ')';

paranthesis (N, op, cl+1, ar, idx+1); }

}

TC = Catalan number $\frac{2n C_n}{n+1}$. (TODO) where N=2n.

Matrix to magic square with minimum cost

constraints :- Matrix size = 3×3
$1 \le Mat[i][j] \le 9$

DOMS | Page No. | Date / /

Magic square ⇒ row sum = col sum = diag sum

We need to use all the digits.

Example of Magic square :-

4	9	2
3	5	7
8	1	6

Cost calculation :- $\sum_{i=1}^{3}\sum_{j=1}^{3} |mat[i][j] - magicsq[i][j]|$

process :- → Generate all magic squares
→ calculate cost → answer = min (cost's).

To generate magic squares we generate all possible arrays & check if the matrix is magic square.
To generate all the matrices we will convert 2D array to 1D array.

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

→

0	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9

rows :- 012, 345, 678
columns :- 036, 147, 258
diagonals :- 048, 246.

Array generations are done using visited array

visited array

0	1	2	3	4	5	6	7	8	9	
F	F	F	F	F	F	F	F	F	F	initially

0 1 2 3 4 5 6 7 8
1 2 8 4 5 6 7 8 9
9 8

8 7 9
8 9 7
9 8 7
7 6 8 9
7 8 6 9
2 8 9 6
7 9 6 8
7 9 8 6
8
9

Global min_cost = INT_MAX;

```
main() {
  int ar[9];
  bool visited[10];
  int magic_arr[9];
  permutations(ar, visited, magic_arr, 0)
  print(min_cost)
}
```

↓ function in next page

There is a inbuilt function to generate permutations check it on the web.

DOMS | Page No.
Date / /

```
Void permutations (arr, visited, magic_arr, idx){
    if (idx == a){
        if(magic_sqauare(magic_arr)){
            min_cost = min(min_cost, cost(magic_arr, arr))
        }
        return;
    }
    for(int i=1; i<=a; i++){
        if(!visited[i]){
            magic[idx] = i;
            visited[i] = true;
            permutations(arr, visited, magic_arr, idx+1);
            visited[i] = false;
        }
    }
```

This step is helping in backtracking

Given a string A & list of string's check it is possible to partition string A in such a way that every partition is present in list.

Ex:- L = {smart, mart, s, inter, view, interviews, views}
str A: smartinterviews

code to check if string partition is possible for all substrings in L are present.

```
bool partition (str, list, idx) {
    int n = str.size();
    if (idx == n)
        return true;
    for (int i=idx; i<n; i++) {
        if (str[idx:i] ∈ L) {
            if (partition (str, list, i+1)) {
                return true
            }
        }
    }
    return false;
```

code to count number of ways to do partition

```
int ways = 0
void partition (str, list, idx = 0) {
    int n = str.size();
    if (idx == n)  ways += 1;
    for (int i=idx; i<N; i++) {
        if (str[idx:i] ∈ L) {
            partition (str, list, i+1)
        }
    }
}
```

minimum number of partitions #TODO