Staircase Problem Recursion Quick Revision Notes (Hinglish)

Logic of the Given Code:

- Function `stair(int x)` ye calculate karta hai ki `x`th step tak kitne tareeke se pahunch sakte hain.
- **Base Condition**:
  - Agar `x == 1`, toh sirf ek tareeka hai pahunchne ka, isliye return `1`.
  - Agar `x == 0`, toh ek tareeka hai wahi rehne ka, isliye return `1`.
- **Recursive Case**:
  - Agar kisi step `x` tak pahunchna hai, toh hum ya toh `x-1` step se aa sakte hain (1 step upar chadh ke)
    ya `x-2` step se (2 step ek saath chadh ke).
  - Isliye, `stair(x) = stair(x-1) + stair(x-2)`, jo basically Fibonacci sequence hai.

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

// Function jo calculate karega ki x-th step tak kitne tareeke se pahunch sakte hain
int stair(int x){
    if(x == 1) return 1; // Base case: Step 1 tak sirf ek tareeka hai
    else if(x == 0) return 1; // Base case: Step 0 par rehne ka ek hi tareeka hai
    else return stair(x-1) + stair(x-2); // Recursive case
}

int main(){
    int p;
    p = stair(10); // Step 10 tak pahunchne ke tareeke count karne ke liye function call
    cout << p; // Result print karega
}
```

Dry Run of the Code (For stair(5))

Function Call	x	Return Value
stair(5)	5	stair(4) + stair(3)
stair(4)	4	stair(3) + stair(2)
stair(3)	3	stair(2) + stair(1)
stair(2)	2	stair(1) + stair(0)
stair(1)	1	1 (Base Case)
stair(0)	0	1 (Base Case)

Final Computation:

stair(2) = 1 + 1 = 2

stair(3) = 2 + 1 = 3

stair(4) = 3 + 2 = 5

stair(5) = 5 + 3 = 8

Output for stair(5): 8

stair(10) ka output ek bada number hoga, jo Fibonacci pattern follow karega.

Time Complexity:

- Is function ka time complexity **O(2^n)** hai kyunki isme bohot saari repeated calculations hoti hain.
- Agar hum **memoization (Dynamic Programming)** use karein toh isko **O(n)** tak optimize kiya ja sakta hai.

Key Takeaways:

- Yeh problem Fibonacci sequence par based hai.
- Given recursive solution large values ke liye inefficient hai kyunki redundant calculations hoti hain.
- Optimized version **Dynamic Programming ya Memoization** use karta hai jo **O(n) time** leta hai.