Recursion

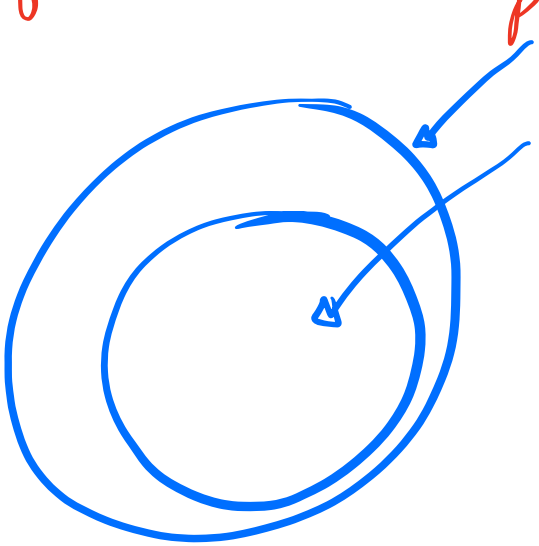
- Merge Sort, Quick Sort,
- Dfs → graph algo;
- DP → ✓
- Backtracking
- Trie

What is recursion?

→ f^n calling itself.

↳ Solve a problem with the help of smaller versions of the same problem!

```
f() {
  ==========
    f();
  ==========
}
```

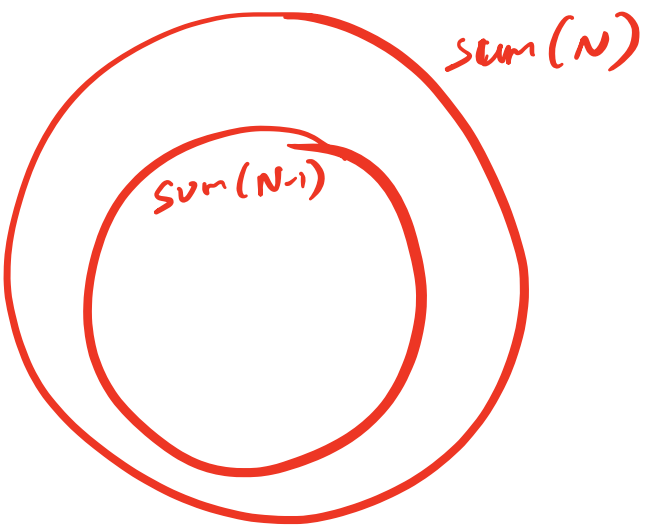


Q Given N. find the sum of first N natural no's.

$$Sum(N) = 1 + 2 + 3 + \cdots + N-1 + N$$

$$Sum(N-1) + N$$

$$Sum(N) = Sum(N-1) + N$$



How to write recursive code?

1) Assumption: Decide what your f^n should do & then ASSUME that it does it!

2) Main Logic: Calculate the ans of the problem using the ans of the subproblem(s).

3) Base Condition: find the stopping condition!

Q Given N . find $1 + 2 + \cdots + N$.

$1 <= N$

```
int sum(N) {
// Ass: ret the sum of 1st N natural no's.
   if(N==1) ret 1;
   ret sum(N-1) + N;
}
```

Q Given N ($N \geq 0$). find $N!$

```
int fact(N) {
// Ass: ret N!
    if (N==0) ret 1;
    ret fact(N-1) * N;
}
```

$N! = 1 \times 2 \times 3 \times \cdots \times N-1 \times N$

$N! = (N-1)! \times N$

$\boxed{fact(N) = fact(N-1) \times N}$

$0! = 1$

Q Given N. Calculate N^{th} fibonacci No!

$N \geq 1$

N →	1	2	3	4	5	6	7	8	9	10
f(N) →	0	1	1	2	3	5	8	13	21	34

$\boxed{f(N) = f(N-1) + f(N-2)}$

```
int f(N) {
// Ass: ret N^th fib. No!
    if(N<=2) ret N-1;
    ret f(N-1) + f(N-2);
}
```

BASE CASE

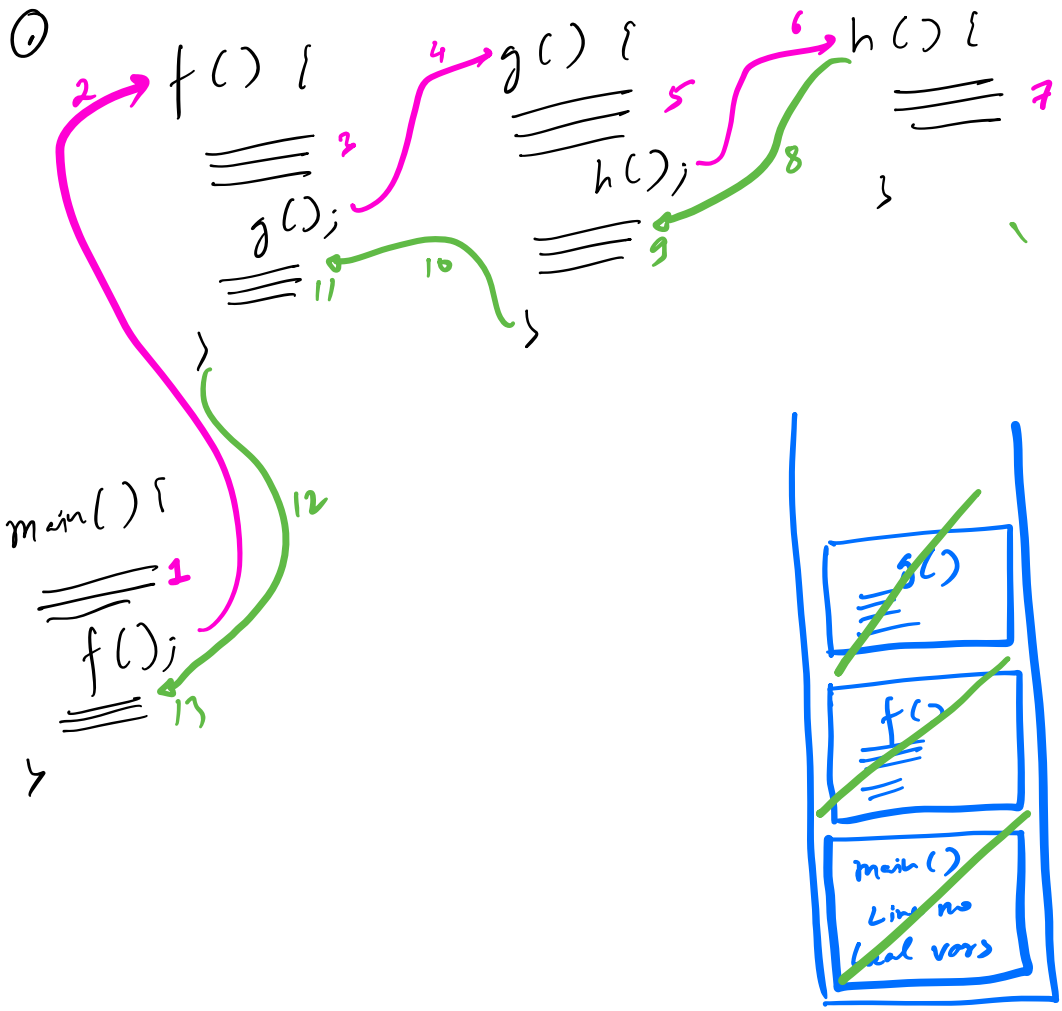
for the cases whose ans is undefined by the recurrence relation!

$N=2$

$f(2) = f(1) + f(0)$ — UNDEFINED

$N=1$

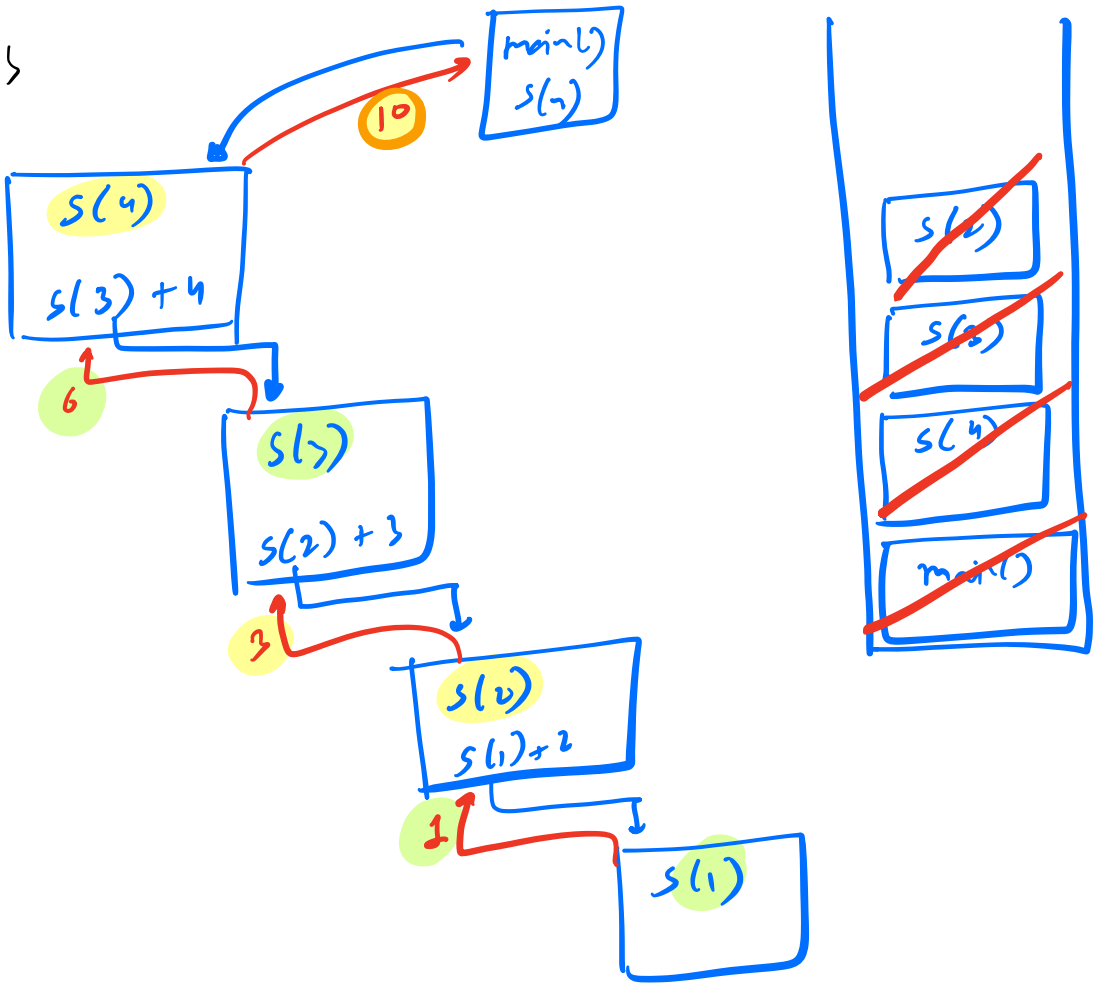
$f(1) = f(0) + f(-1)$


0
f() {
g();
}
g() {
h();
}
h() {
}
main() {
f();
}
1
2
3
4
5
6
7
8
9
10
11
12
13

g()
f()
main()
Live no
local vars

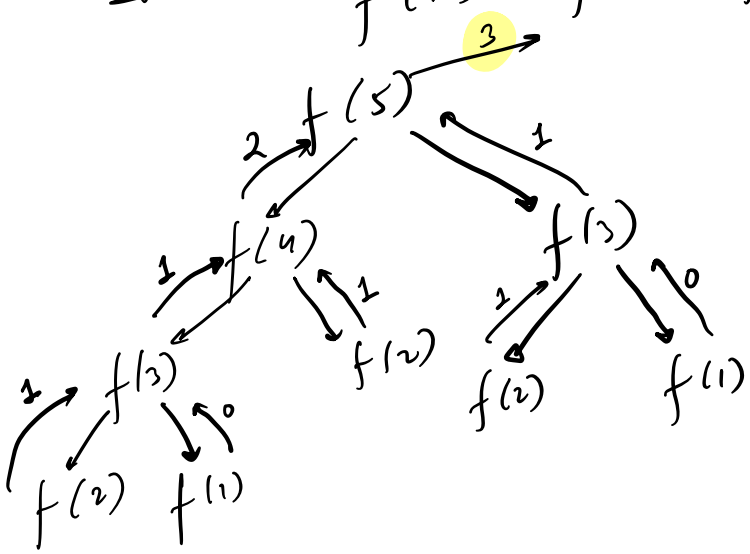
⊙ Sum of 1st N natural No's

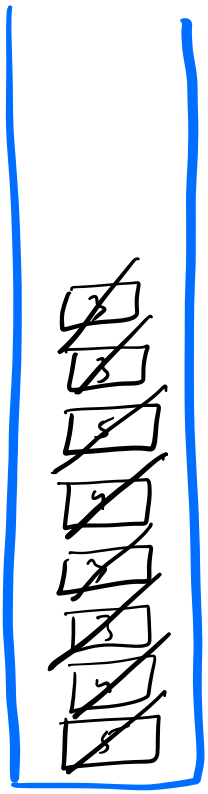
```
int sum(N) {
    if(N==1) ret 1;
    ret sum(N-1) + N;
}
```



④ fib

$f(N) = f(N-1) + f(N-2)$





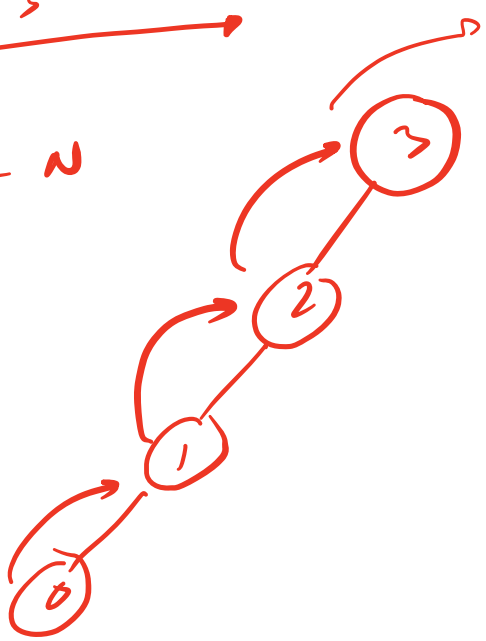
Q Given N. Print the 1st N natural no's!

N >= 1

N : 5

1 2 3 4 5

```
void inc (N) {
  //ASS : print 1, 2, 3- - - N
  if (N==0) { ret; }
  inc(N-1);
  print(N);
}
```



1 2 3

Q Given N. Print 1st N natural no's in reverse order!

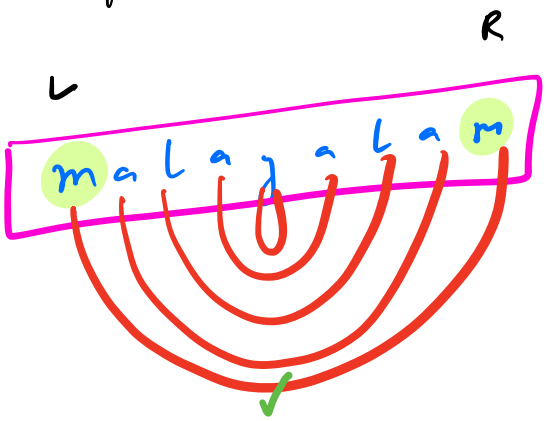
N = 5 ⇒ 5 4 3 2 1

```
void dec(N) {
  //Ass: print N ⟶ 1
  if(N==0){ ret; }
  print(N)
  dec(N-1);
}
```

N, N-1, ... 3, 2, 1

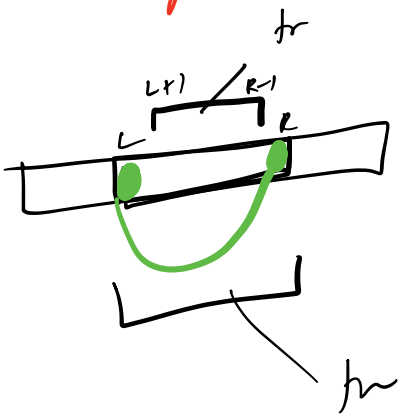
Q Given a string.
Check if it is a palindrome or not!

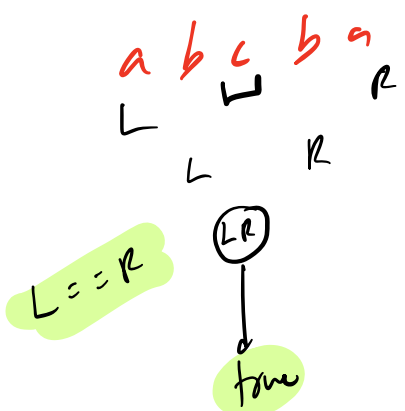
madam

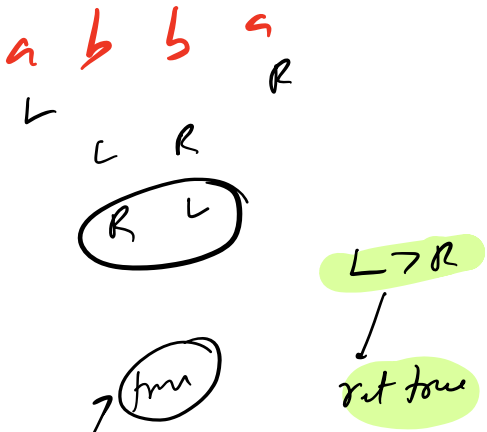


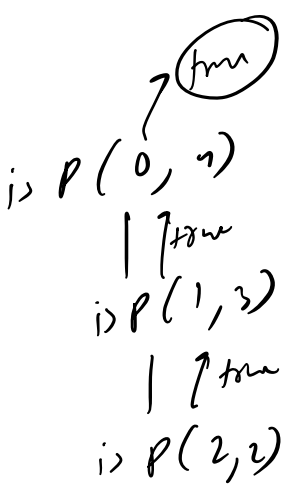
(, 0, N-1)

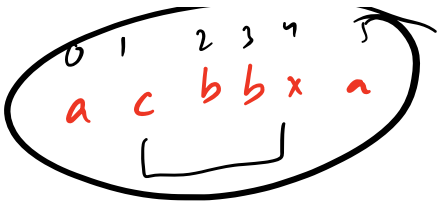
```
bool isP(S, L, R) {
  // Ass: ret true if S[L ... R] is a palindrome
  //      false otherwise
  if (L >= R) ret true;
  if (S[L] != S[R]) ret false;
  ret isP(S, L+1, R-1);
}
```











is P(0, 5) → false

| ↑ false

is P(1, 4)