Recursion

STEPS:

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

2. Main Logic: Solve the problem using the solutions of the subproblem!

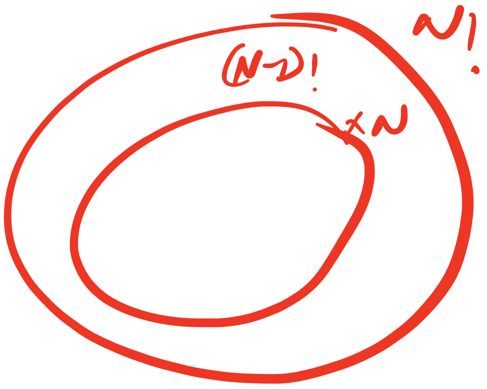
3. Base Case: ________

Q. Given N. Calculate N! [N ≥ 0]

N = 5 : $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

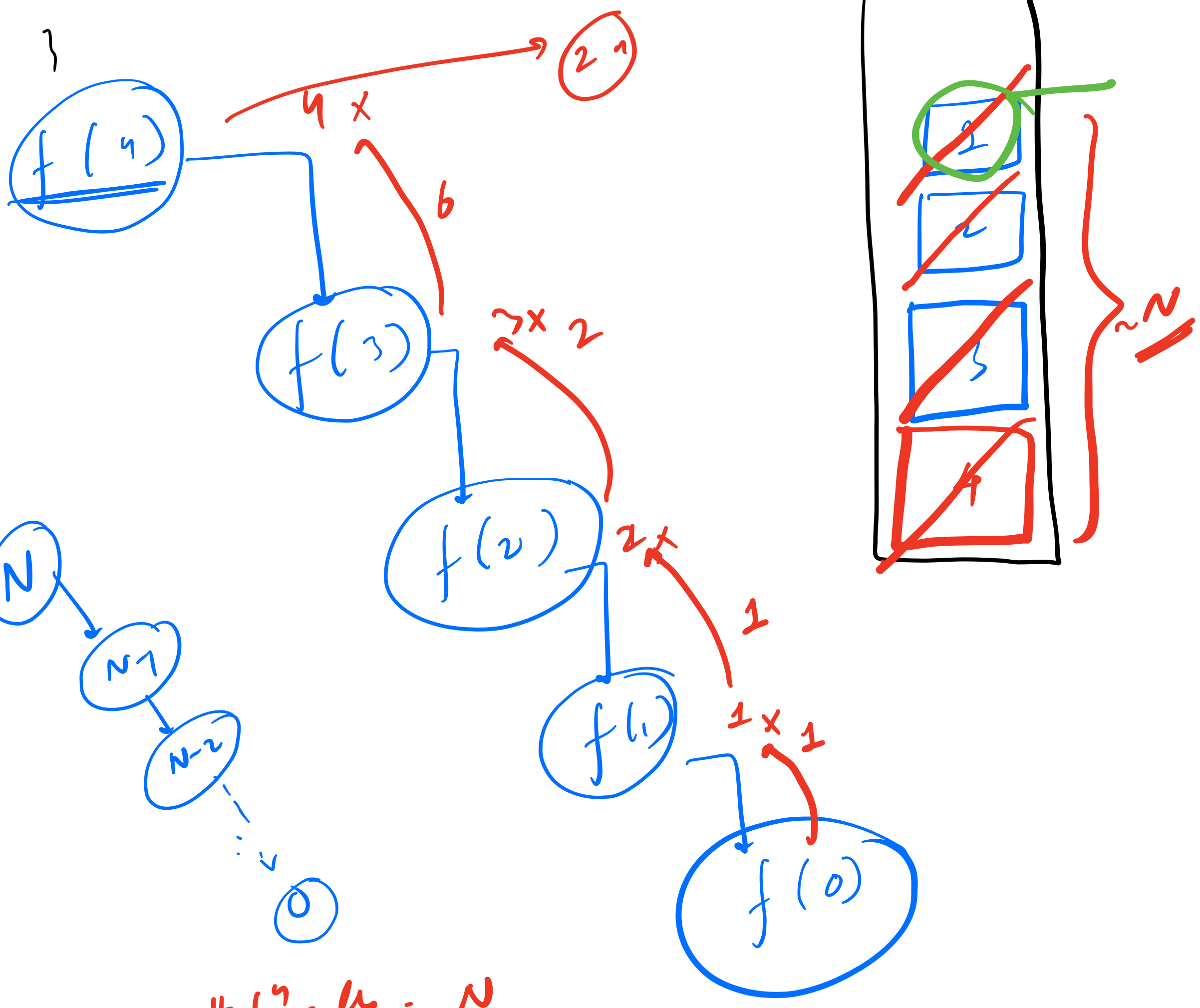
$$N! = \underbrace{1 \times 2 \times 3 \times \cdots \times (N-1)} \times N$$

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



```
int fact(N) {
  // ASS: return N!
    if(N==0) ret 1;
    ret fact(N-1) x N;
}
```

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



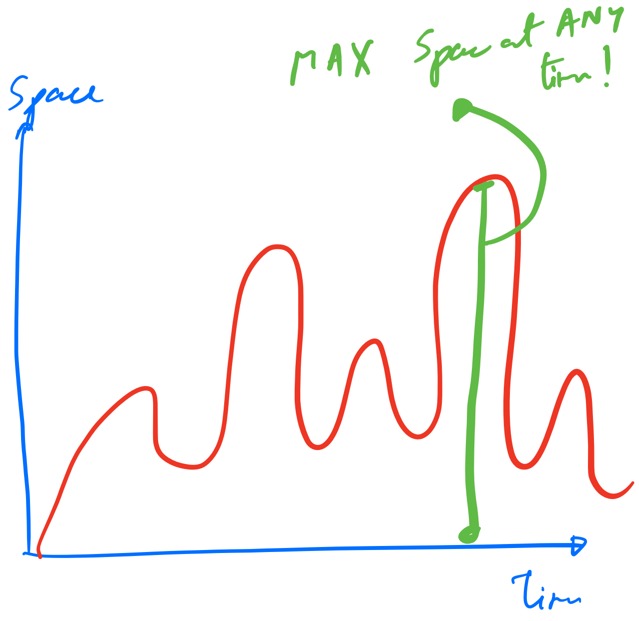
f^n calls = N

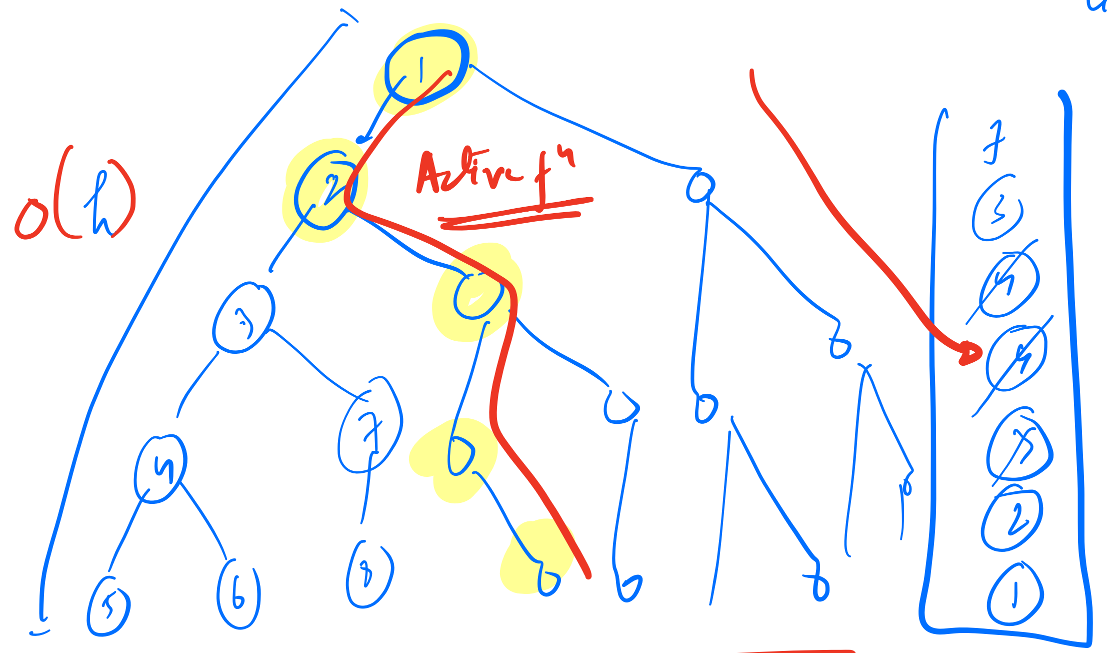
time taken / f^n call → O(1)

$\boxed{TC = O(N)}$

$$\boxed{TC = \# f^n \text{ calls} \times \text{time taken} / f^n \text{ call}}$$

$$\boxed{SC = O(N)}$$





$$\boxed{SC = \text{MAX } \# \text{ of Active } f^n \text{ calls} \times \text{space taken} / f^n}$$

Q

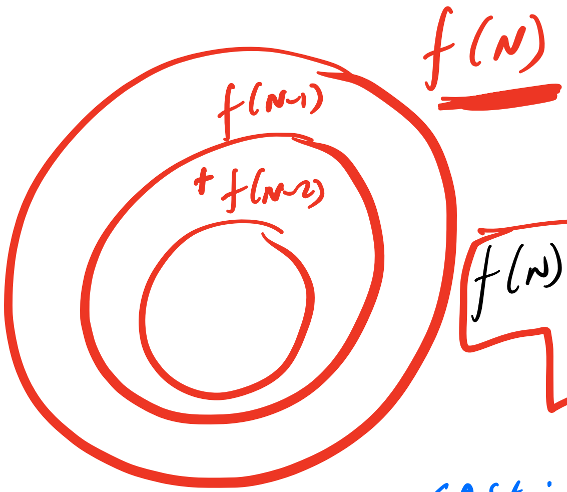
Fibonacci No $\longrightarrow$

				N-2	N-1	N				
1	2	3	4	5	6	7	8	9	10	. —
0	1	1	2	3	5	8	13	21	34	- —

Given. N . find N^{th} fib No!

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

}
```



RR

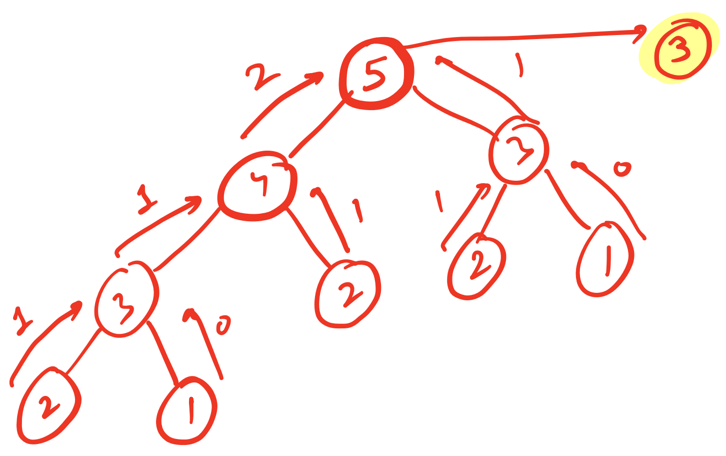
$$\boxed{f(N) = f(N-1) + f(N-2)}$$

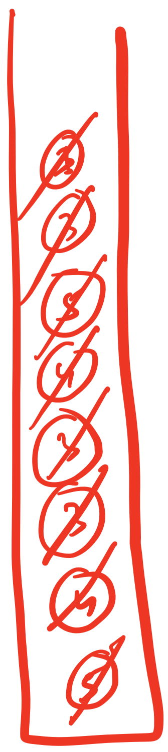
BASE CASE :

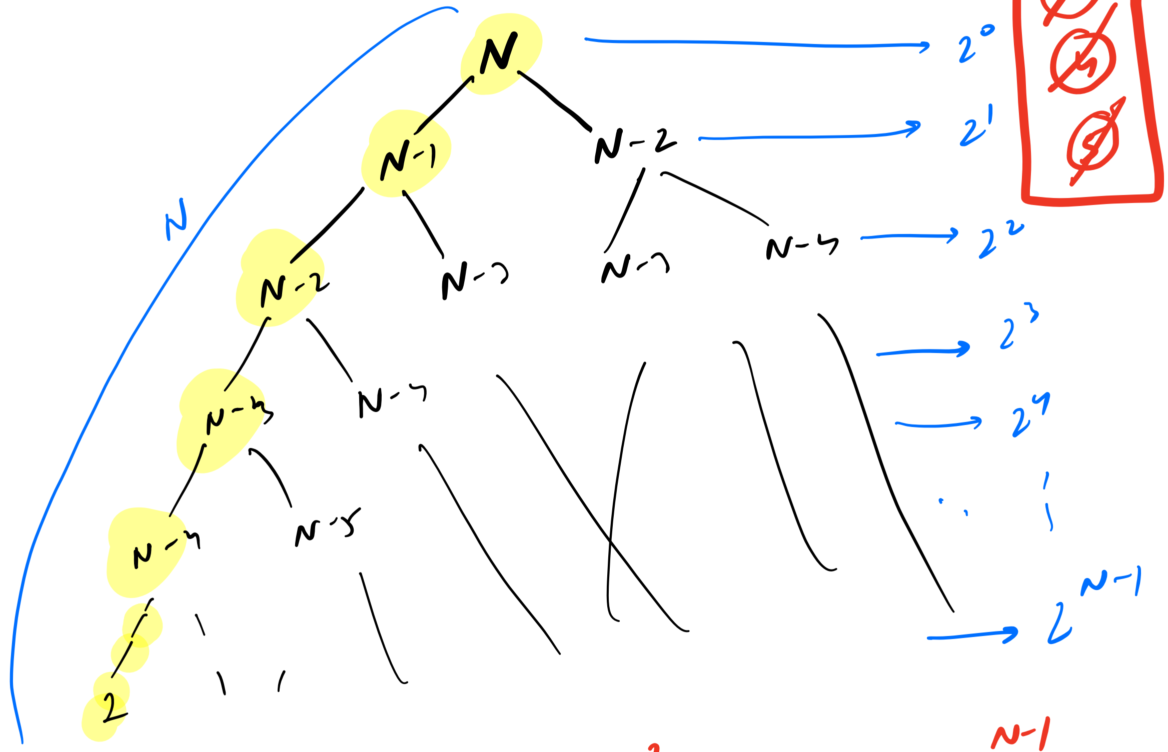
1) Cases Not defined by R.R.

✓ $f(2) = f(1) + f(0)$

✓ $f(1) = f(0) + f(-1)$







$\# f^n$ calls $= 2^0 + 2^1 + 2^2 + \text{---} + 2^{N-1}$

$= \boxed{2^N - 1}$

$TC = 2^N \times O(1)$

$\boxed{TC = O(2^N)}$

$N = 20$ $\sim 2^{20}$ $\sim 10^6$

SC

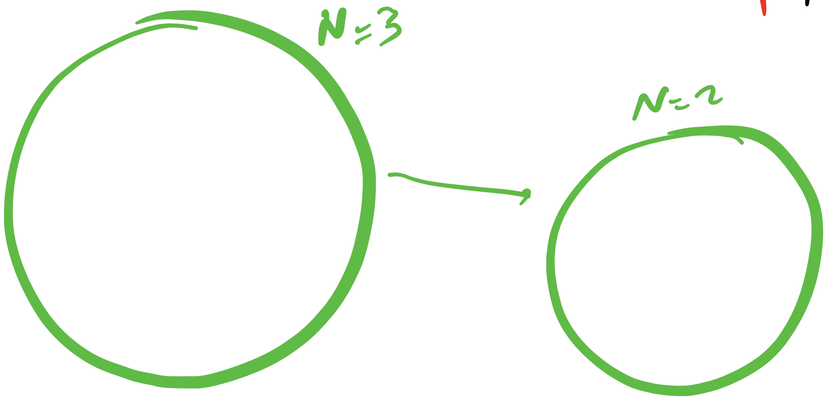
$\boxed{SC = O(N)}$

Q Given N. Generate all binary strings of len N. 2^N

N = 3 : { "000"
"001"
"010"
"011"
"100"
"101"
"110"
"111" }

N = 2

0	00
0	01
0	10
0	11
1	00
1	01
1	10
1	11



// N > 0

```
List<string> gen(N) {
  // Ass: Ret the list of All N sized binary strings!
  if(N==1) { ret {"0", "1"}; }

  List<string> P = gen(N-1);
  List<string> ans;
  f(s : P) {
      ans.add("0"+s);
  }

  f(s : P) {
      ans.add("1"+s);
  }
  ret ans;
}
```

N=3

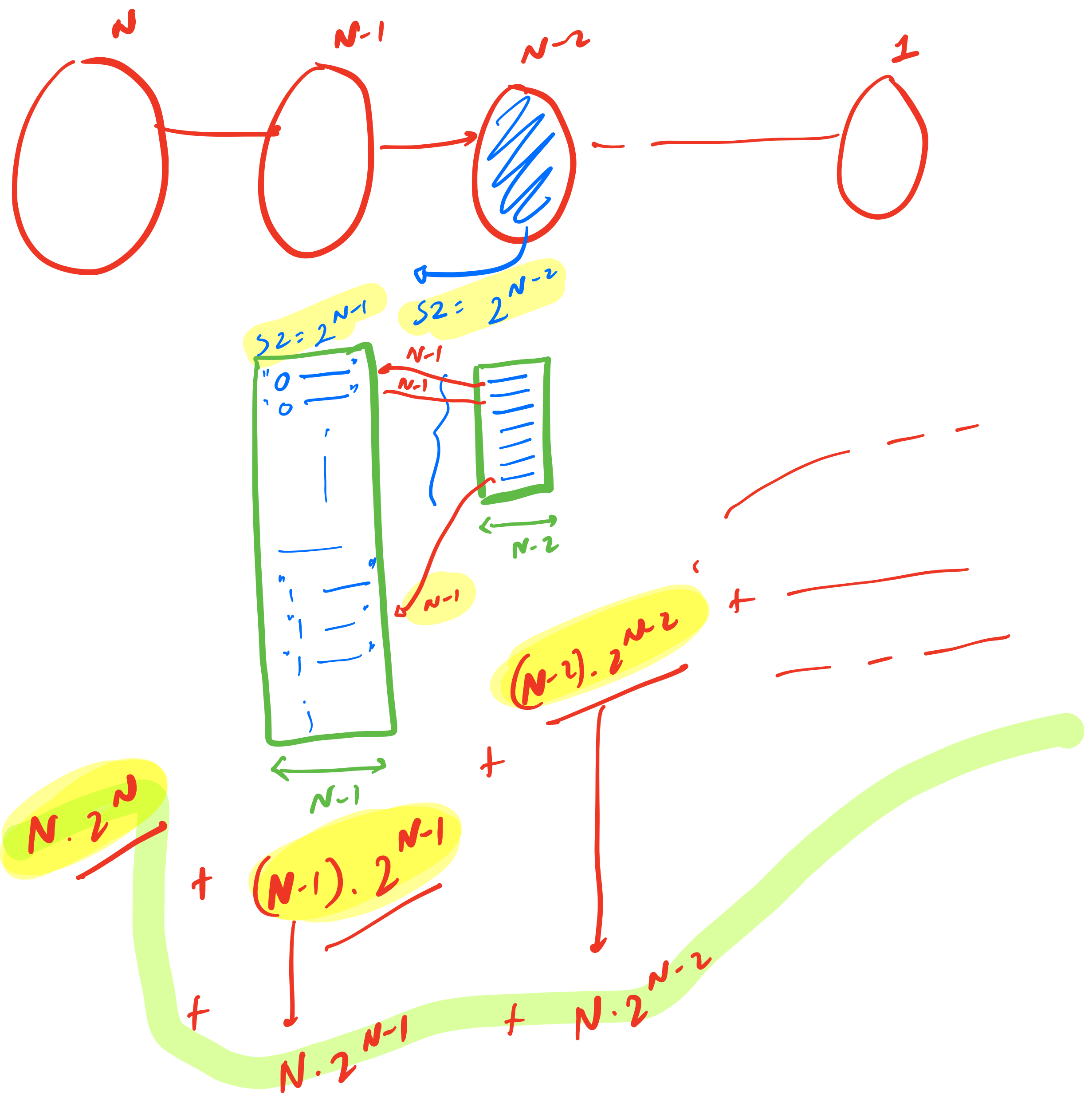
P

[00
01
10
11]

N=1

["0"
"1"]

```
if(N==0)
    ret {""}
```

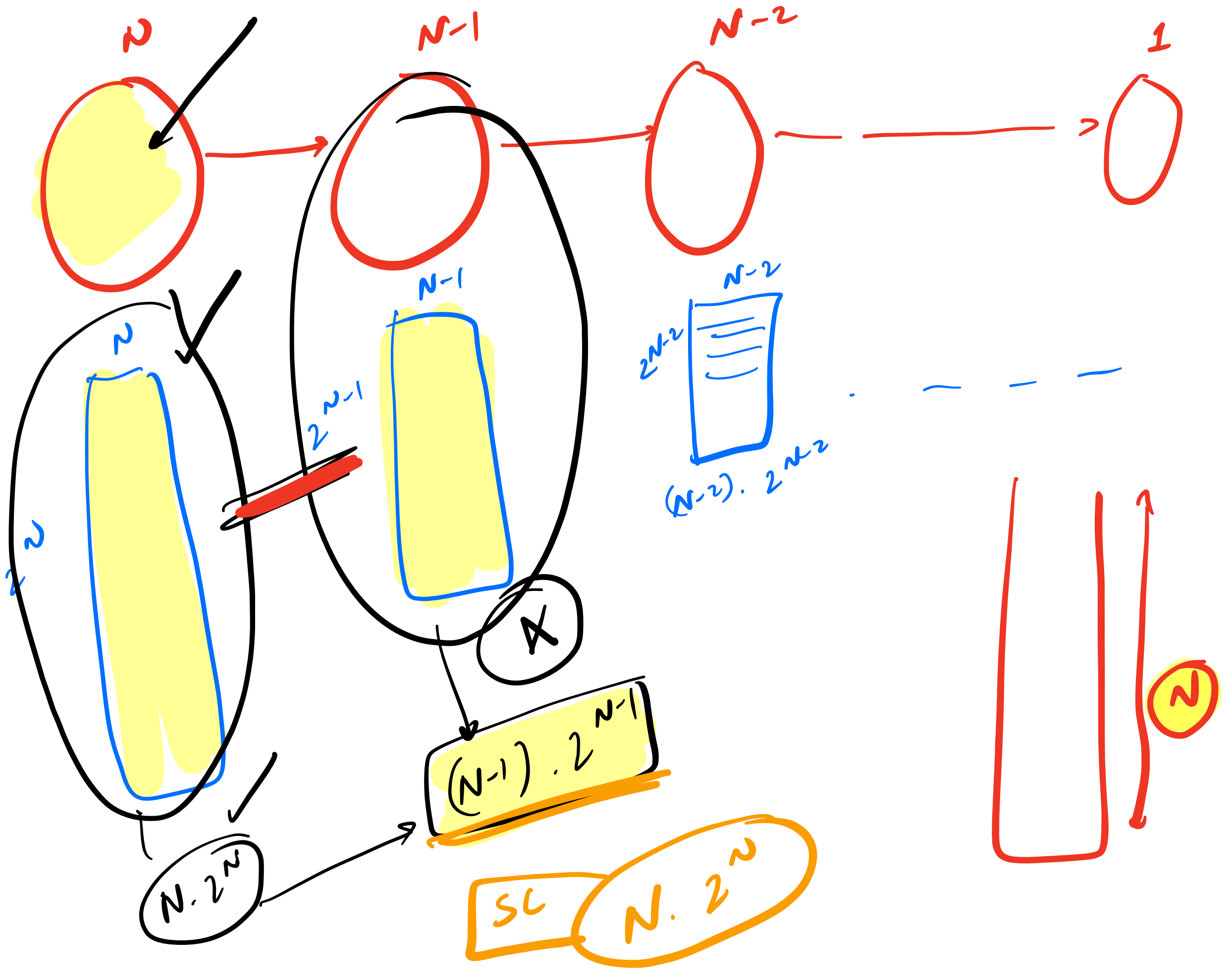


$N[2^N + 2^{N-1} + \text{---} \; 2^0]$

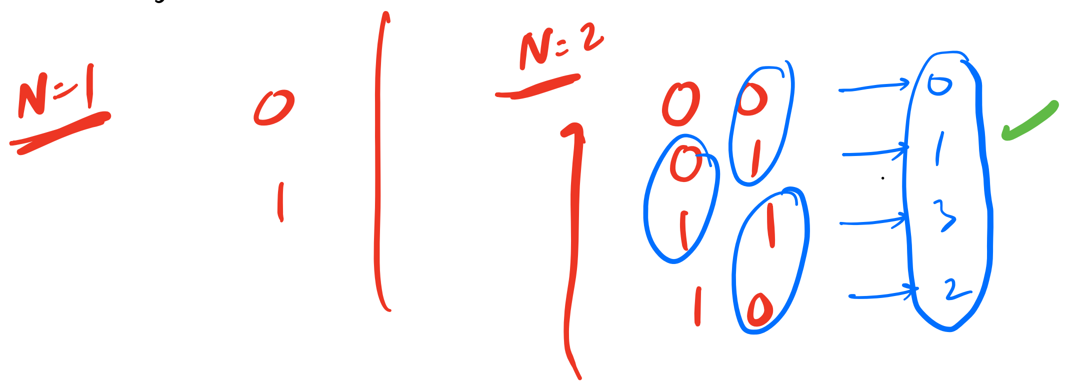
$N.[2^{N+1} - 1]$

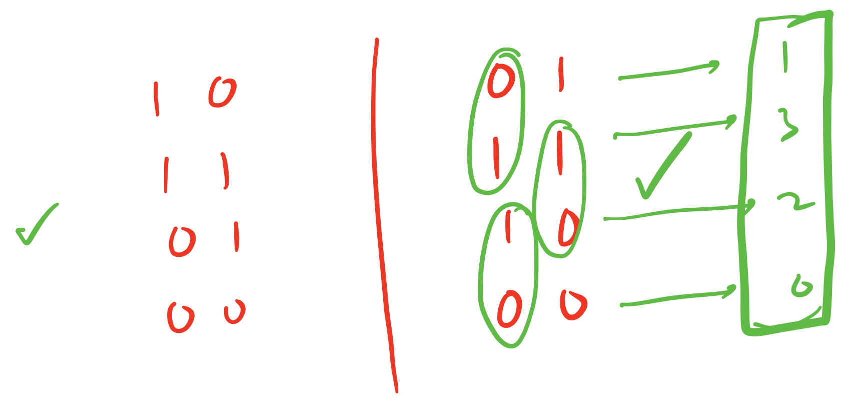
$\boxed{TC}$

$O(N.2^N)$



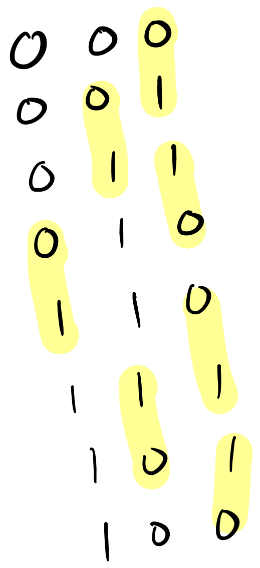
Q Given N. Generat all N bit gray code no's.
Adjacent no's in sequence should differ by 1 bit!



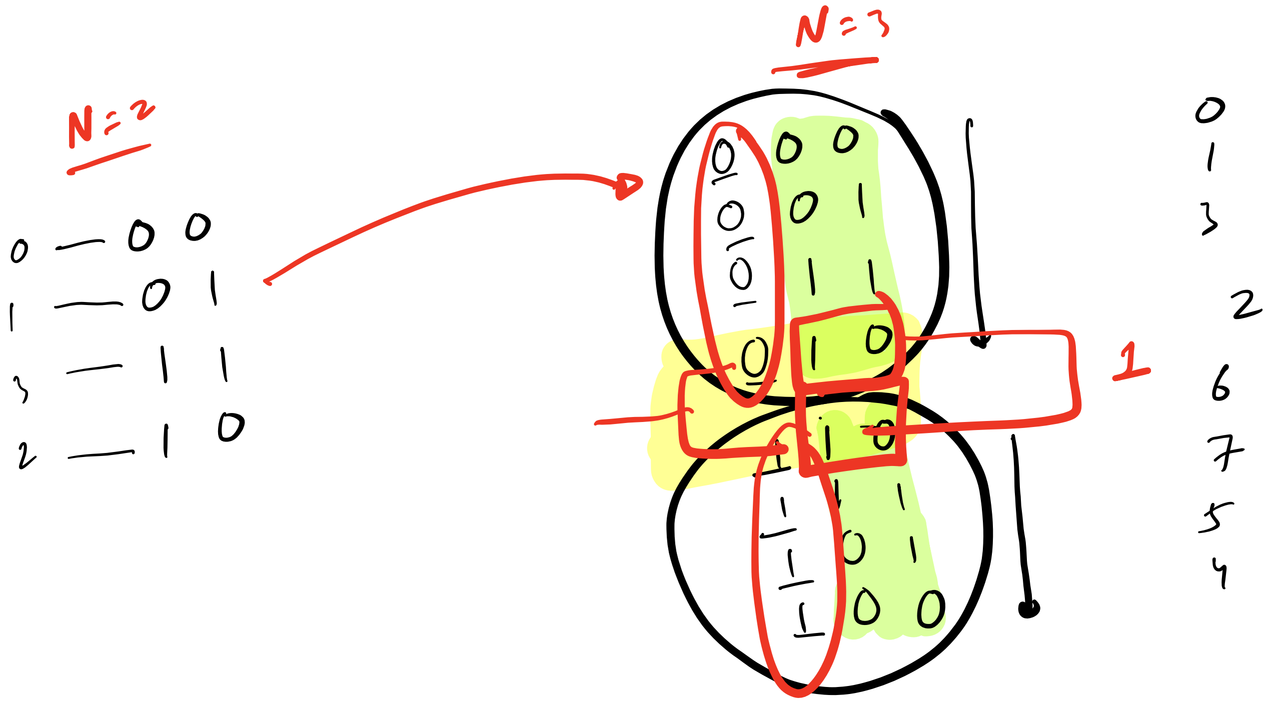


1 0
1 1
0 1
0 0
0 1
1 1
1 0
0 0
1
3
2
0

N=3



0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0

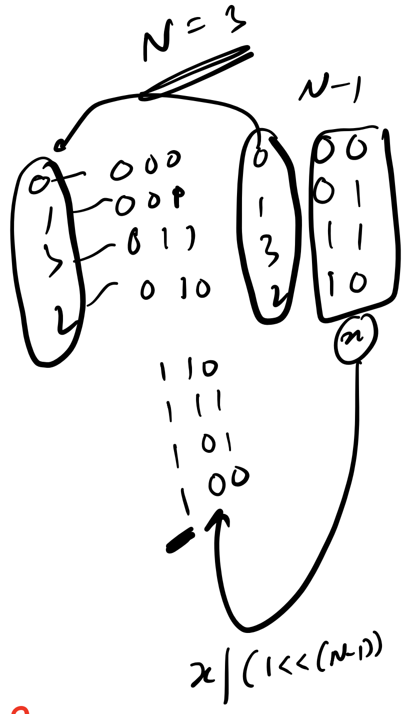


N=3
N=2
0 — 0 0
1 — 0 1
3 — 1 1
2 — 1 0
1
0
1
3
2
6
7
5
4

```
List <int> gen(N) {
   if(N==1) { ret (0,1);}
    List<int> p=gen(N-1);
    List<int> ans;
     f( s: p) {
         ans.add(s);
     }
     f( s: rev(p)) {
        ans.add( s|(1<<(N-1)));

     }
     ret ans;
}
```

N > 0



$\boxed{TC / SC}$?