- Intro to recursion
- 3 step to write recursive code
- tracing | Fact
  print
  Fib
- TC
- SC
- Fib

recap of recursive 1 & recursive 2 intermediate sessions in one session

## why Recursion?

- Merge Sort, Quick sort
- Binary Tree, BST, segment tree, tries
- Dynamic Programming
- Backtracking
- Graphs

Recursion? function that calls itself

Solving problem using smaller instace of the problem

sub problem

$\sum_{n}$ sum(4) = 4+3+2+1 = 4 + sum(3)

sum(3)

sum(n) $1+\cdots+n$

sum(n) = sum(n-1) + n

fact(4) = 4x3x2x1 = 4x fact(3)

fact(3)

factorial ( )!

fact(n) = n x fact(n-1)

How to write recursive code?

① Assumption decide what your func. does

② Main Logic solve problem using subproblems

③ Base Condition Input for which we need stop

Sum of n natural numbers:

ex
```
int sum(n) {
  if(n==1) ret 1
  ret n+sum(n-1)
}
```
✓① Assumption
✓② Main Logic
✓③ Base Condition

```
int fact(n) {
  if(n==0) ret 1
  ret n x fact(n-1)
}
```
✓① Assumption
✓② Main Logic
✓③ Base Condition

---

function Call tracing

```
int add(a,b){
  ret a+b
}

int mul(x,y){
  ret x*y
}

int sub(P,Q){
  ret P-Q
}

int div(a,b){
  ret a/b
}
```

```
main(){
  X=10, Y=20
  print(sub(mul(add(X,Y),30),75))
}
```

825 sub(mul(add(X,Y),30),75)

900 mul(add(X,Y),30)

30 add(X,Y) 10+20

10 20

output: 825

# Related data structure

| | |
|---|---|
| ~~add(x,y)~~ | 10+20=30 |
| ~~mul(add(x,y),30)~~ | 30×30=900 |
| ~~sub(mul(add(x,y),30),75)~~ | 900-75 |
| print(sub(mul(add(x,y),30),75)) | 825 |

Stack

① we add on top

② we remove from top

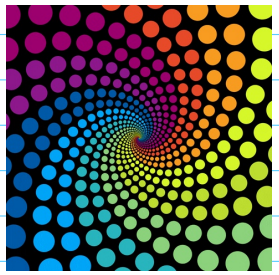
$n \geq 0$

| Fibonacci | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| numbers | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | ... |

+

```
int Fib(n){
    if(n==0) ret 0
    if(n==1) ret 1
    ret Fib(n-1) + Fib(n-2)
}
```

} if(n<=1) ret n



① Assumption ✓

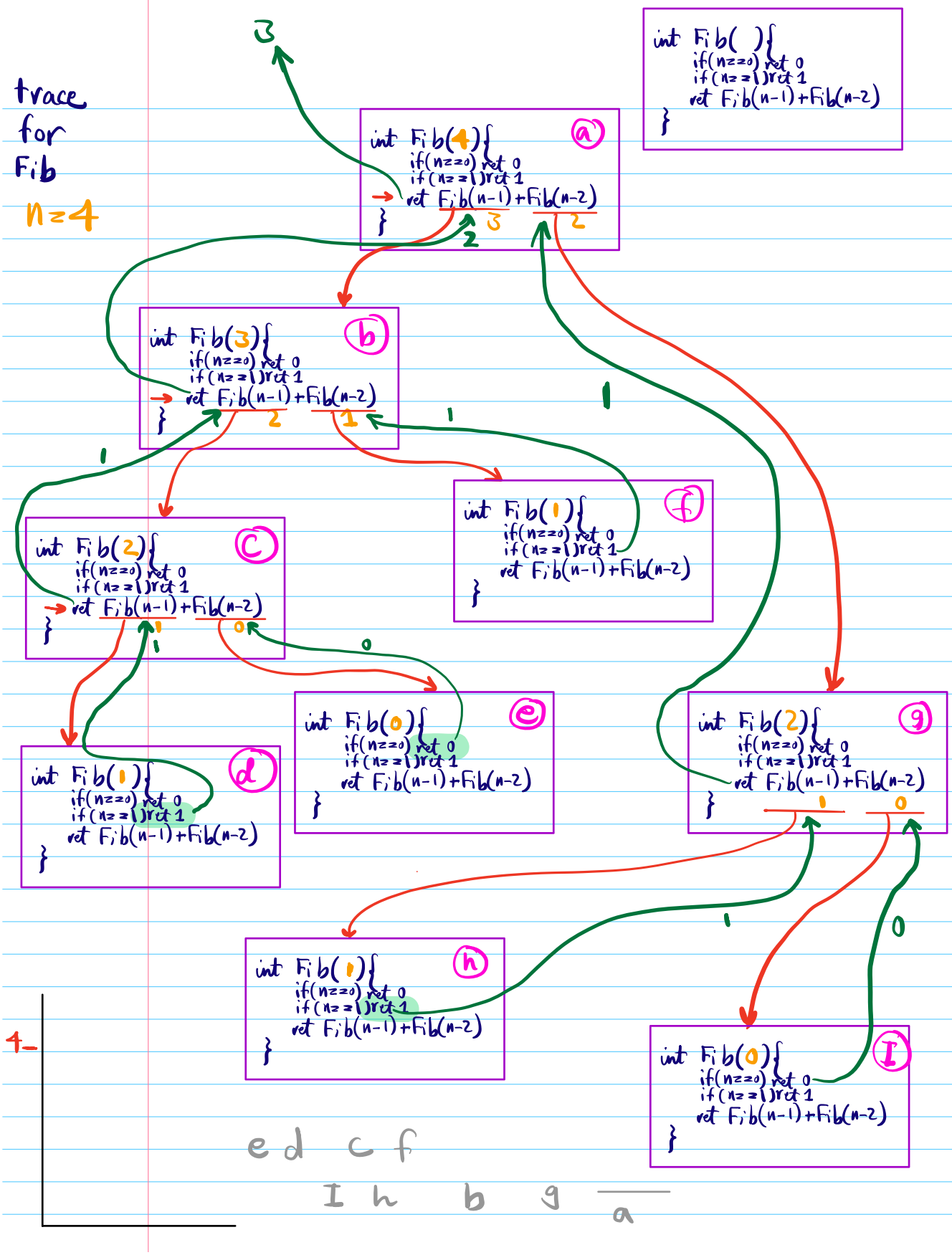
② Main Logic ✓

③ Base Condition ✓

$F(2) = F(1) + F(0)$

trace for Fib

n=4

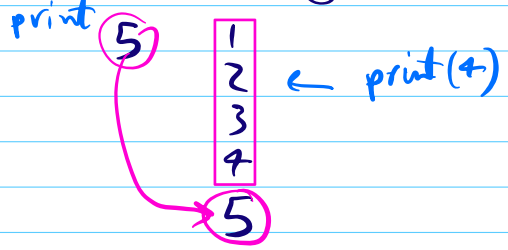


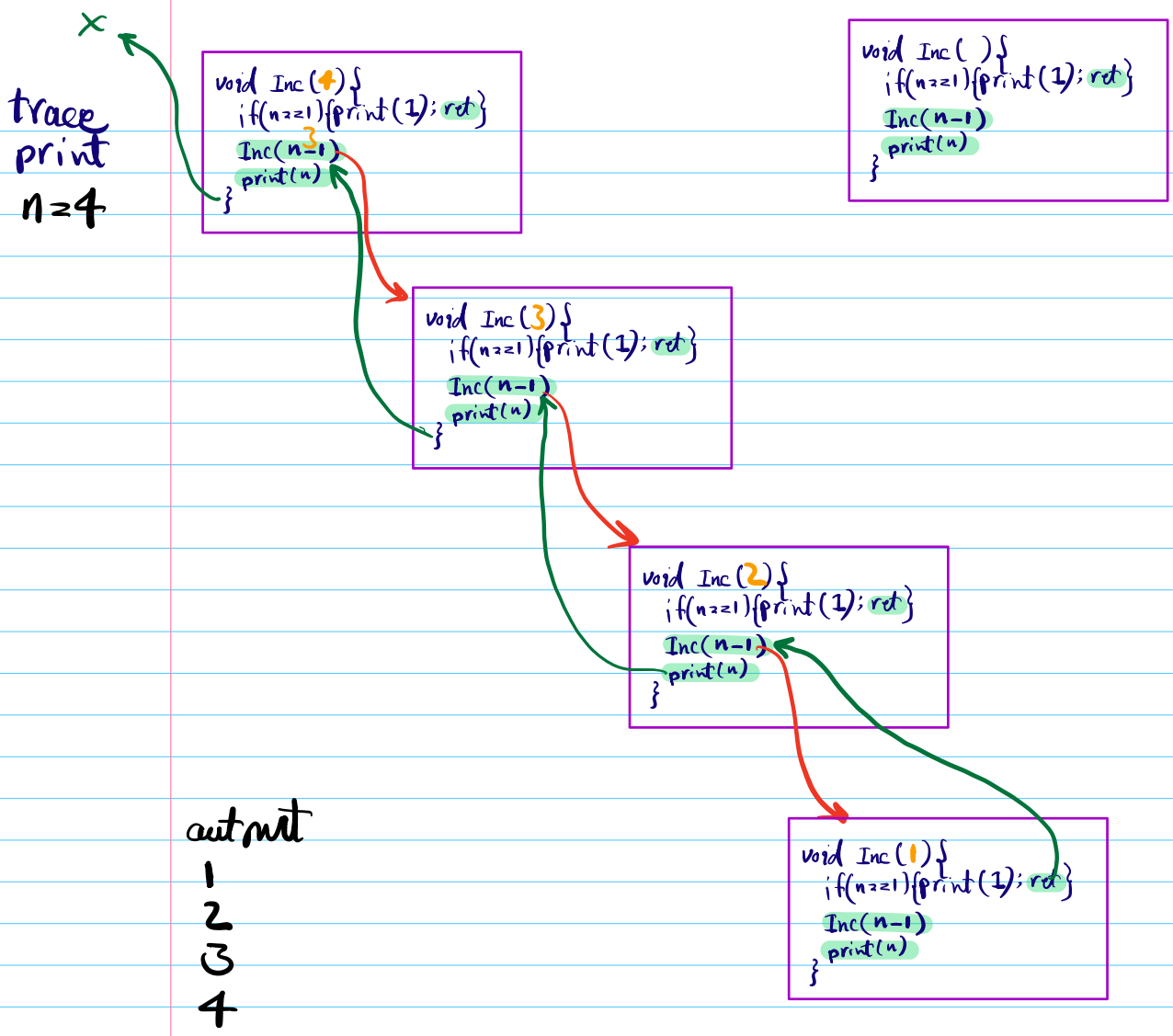
P1 Given N, print all numbers from 1 to N in increasing order; use recursive solutions

```
void Inc(n){
  if(n==1){print(1); ret}
  Inc(n-1)
  print(n)
}
```

swap what will happen

✓ ① Assumption
② Main Logic
③ Base Condition




trace
print
n=4
x
void Inc(4){
if(n>=1){print(1); ret}
Inc(n-1)
print(n)
}
void Inc(3){
if(n>=1){print(1); ret}
Inc(n-1)
print(n)
}
void Inc(2){
if(n>=1){print(1); ret}
Inc(n-1)
print(n)
}
void Inc(1){
if(n>=1){print(1); ret}
Inc(n-1)
print(n)
}
void Inc( ){
if(n>=1){print(1); ret}
Inc(n-1)
print(n)
}
output
1
2
3
4

# * fast power

```
int pow (int a, int n){
  if(n==0) ret 1 ;  p=pow(a, ⌊n/2⌋)
  if(n%2==0){ even?
    ret PxP
  }else{ odd?
    PxPxa
  }
}
```

$a^n = a^{n-1} \times a$

$a^5 = a^3 \times a^2$

$= a^1 \times a^4$

$pow(a,n) = pow(a,n-1) \times a$

$a^{16} = a^{15} \times a$

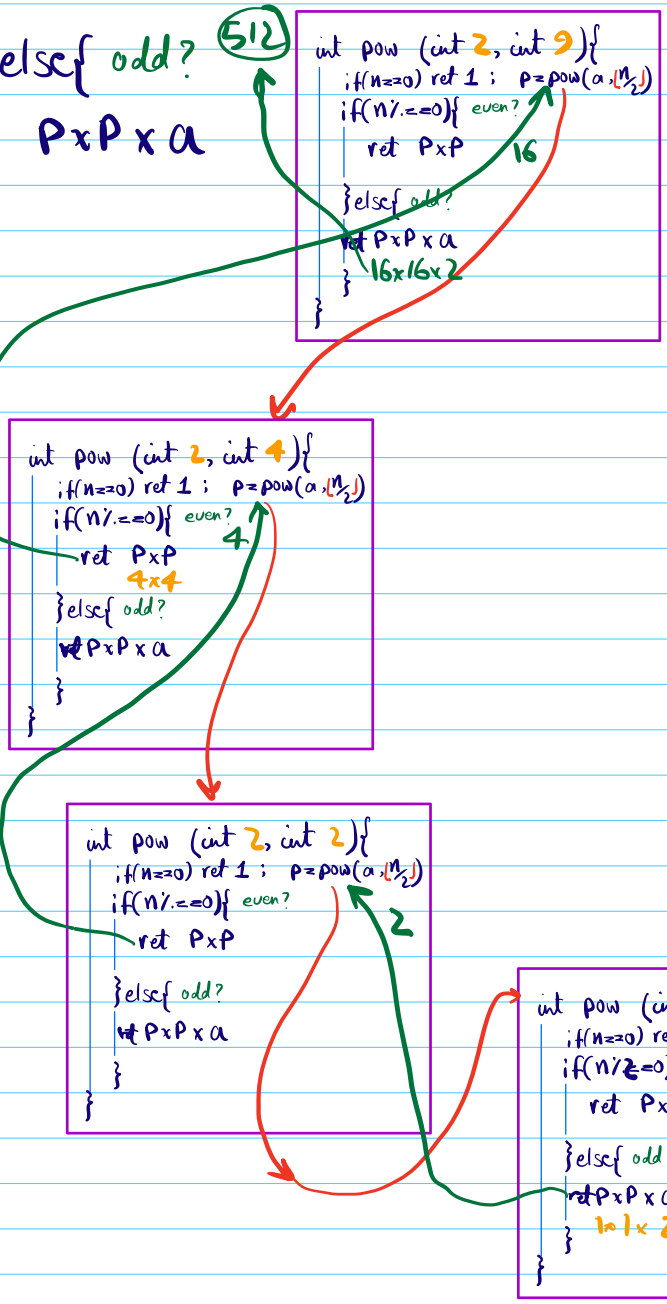
$= a^8 \times a^8$

ret V x V

$a^{17} = a^8 \times a^8 \times a$

V x V x a

Tracing of pow3(2,9)



# TC for recursive code using recursive relation

ex: sum

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

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

| | Step |
|---|---|
| $T(n) = T(n-1) + 1$ | 1 |
| $= T(n-2) + 1 + 1$ | 2 |
| $= T(n-3) + 1 + 1 + 1$ | 3 |
| $\vdots$ | |
| $=$ | |
| $= T(1) + n - 1$ | $n-1$ |

$T(n) = 1 + n - 1 = n$

$T(n) = n$ $O(n)$ TC

$n - (n-1)$

ex: fact

```
int fact(n){
    if(n==1) ret 1
    ret  n*fact(n-1)
}
```

TC: $O(n)$

```
int pow (int a, int n){
  if(n==0) ret 1
  int p=pow(a, n/2)
  if(n%2==0){
    ret p x p
  }else{
    ret p x p x a
  }
}
```

$T(N) = T(\frac{N}{2}) + 1$

| steps |
|---|
| 1 |

$T(n) = T(\frac{n}{2}) + 1$

$= T(\frac{n}{4}) + 1 + 1$ — 2

$= T(\frac{n}{8}) + 1 + 1 + 1$ — 3

$= T(\frac{n}{16}) + 1 + 1 + 1 + 1$ — 4

$\vdots$

$= T(\frac{n}{2^k}) + k$ — k

$\frac{n}{2^k} = 1$ → 1

$k = \log_2 n$

$T(n) = 1 + k = 1 + \log_2 n$

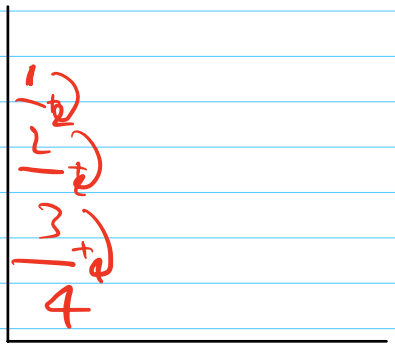
$TC: O(\log_2 n)$

# Space Complexity for recursion?

sum

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

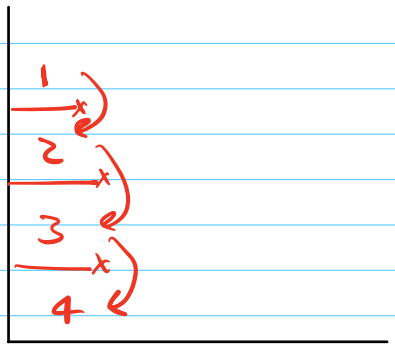
SC: O(n)




fact

```
int fact(n){
    if(n==1) ret 1
    ret  n*fact(n-1)
}
```

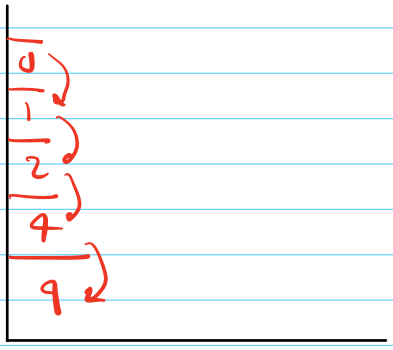
SC: O(n)




pow

```
int pow (int a, int n){
    if(n==0) ret 1
    int p=pow(a, n/2)
    if(n%2==0){
      ret p x p
    }else{
      ret p x p x a
    }
}
```

SC: $O(\log_2(n))$



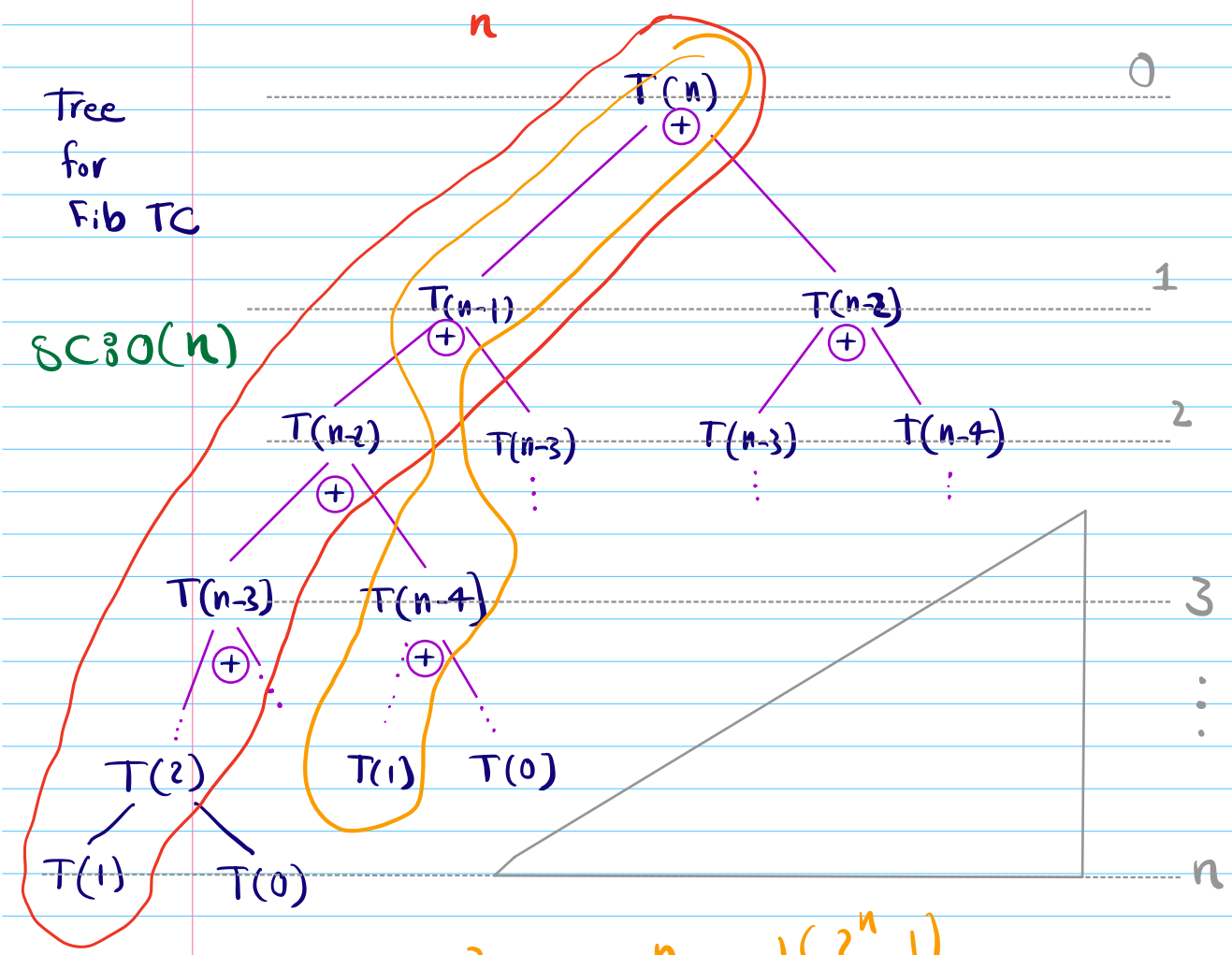
TC, SC
Fib

$F(n) = F(n-1) + F(n-2)$ $\quad F(0)=0$, $F(1)=1$

```
int Fib(n){
    if(n==0) ret 0
    if(n==1) ret 1
    ret Fib(n-1)+Fib(n-2)
}
```

$T(n)$

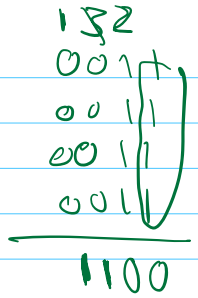
Tree for Fib TC




SC: $O(n)$

$2^0 + 2^1 + 2^2 + 2^3 + \cdots + 2^n = \frac{1(2^n - 1)}{2-1}$

TC $\leq O(2^n)$




4

4 % 2 = 0

4 / 2 = 2