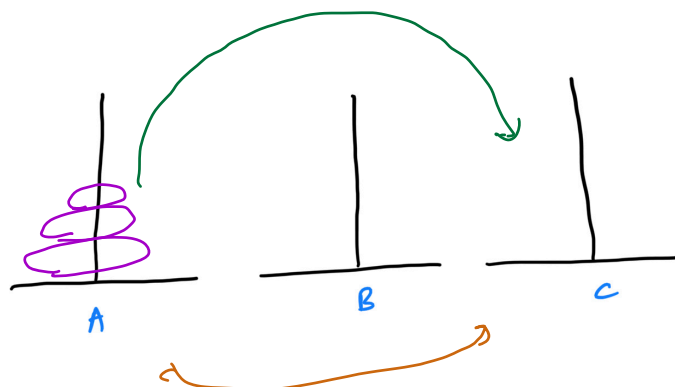


Towers of Hanoi

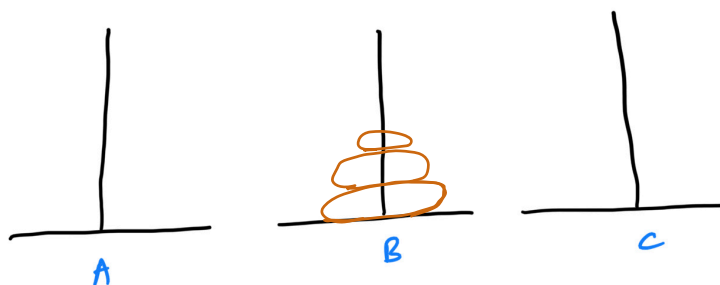


Rules

- 1) Move one disc at a time
- 2) Larger disc should not be put on top of a smaller disc

$N = 3$

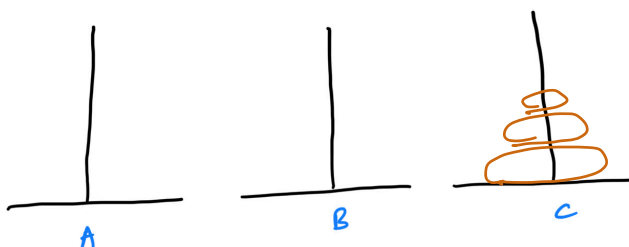
X



$A \rightarrow B$
 $A \rightarrow C$
 $B \rightarrow C$
 $A \rightarrow B$
 $C \rightarrow A$
 $C \rightarrow B$
 $A \rightarrow B$

}

✓

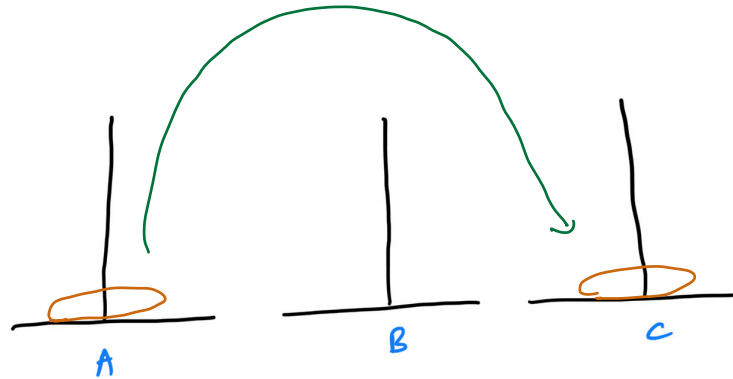


$A \rightarrow C$
 $A \rightarrow B$
 $C \rightarrow B$
 $A \rightarrow C$

$B \rightarrow A$
 $B \rightarrow C$
 $A \rightarrow C$

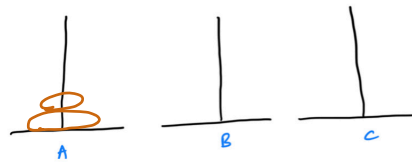
N discs, Move from tower A to C and print all the steps

N=1

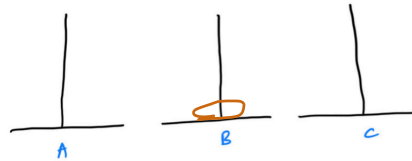


A → C

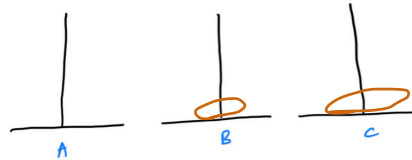
N=2



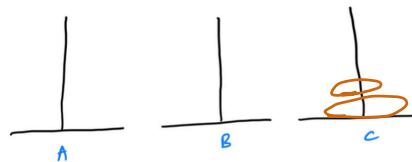
A → B



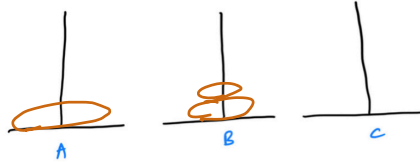
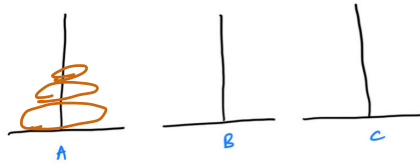
A → C



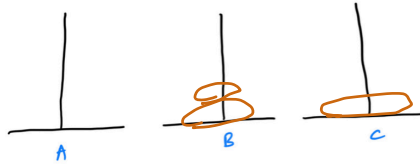
B → C



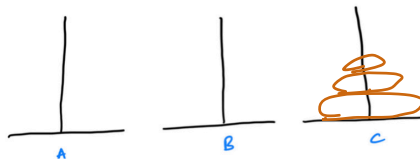
$N=3$



$A \rightarrow C$
 $A \rightarrow B$
 $C \rightarrow B$

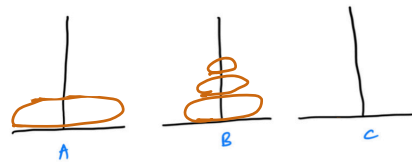
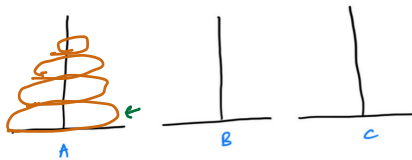


$A \rightarrow C$

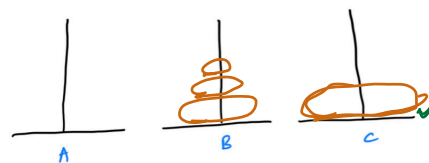


$B \rightarrow A$
 $B \rightarrow C$
 $A \rightarrow C$

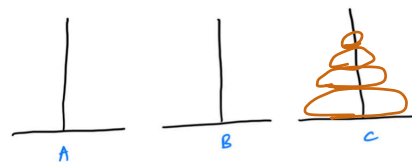
$N=4$



Step 1: Move 3 discs from $A \rightarrow B$
 $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$
 $A \rightarrow B$
 $C \rightarrow A$, $C \rightarrow B$, $A \rightarrow B$

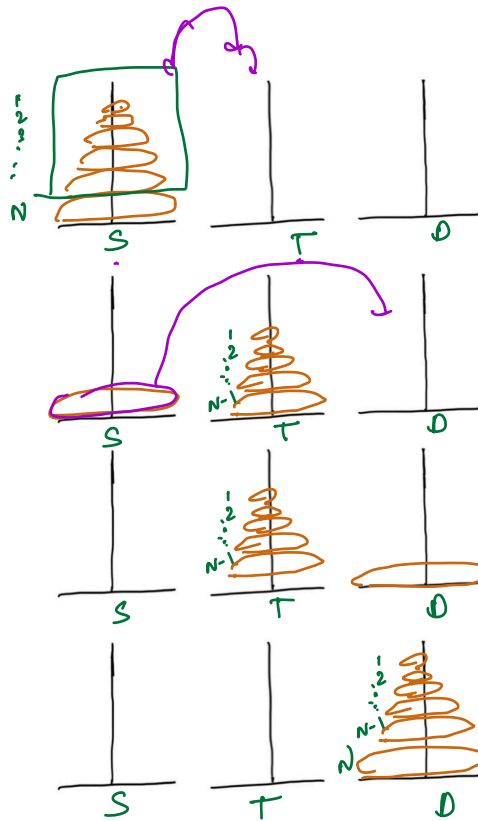


Step 2: Move largest disc from $A \rightarrow C$
 $A \rightarrow C$



Step 3: Move 3 discs from $B \rightarrow C$
 $B \rightarrow C$, $B \rightarrow A$, $C \rightarrow A$
 $B \rightarrow C$
 $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$

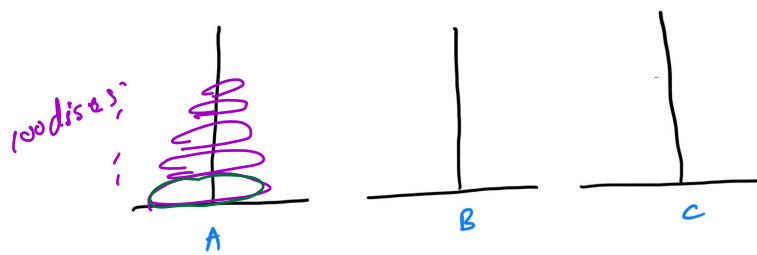
Problem: Move N discs from S to D using T



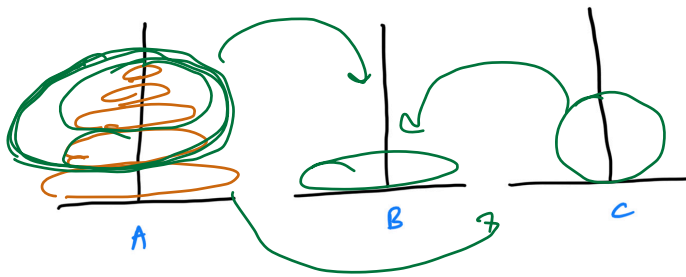
Step 1: Move $(N-1)$ discs from $S \rightarrow T$ using D

Step 2: Move $S \rightarrow D$ ✓

Step 3: Move $(N-1)$ discs from $T \rightarrow D$ using S



→ Move 100 discs from $A \rightarrow B$
 ↳ Move 99 discs from $A \rightarrow C$

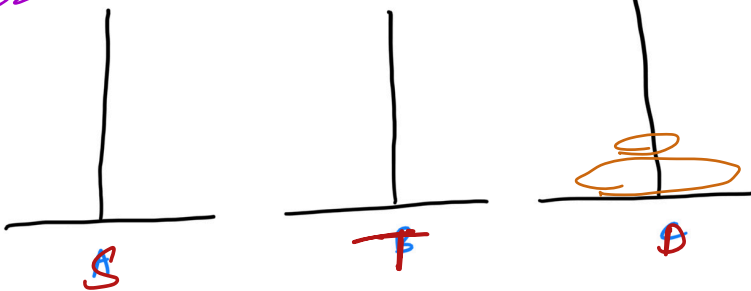


- Move 5 disks from A to C
- 1) Move 4 disks from A to B
 - 2) $A \rightarrow C$
 - 3) Move 4 disks from B to C

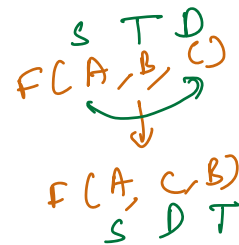
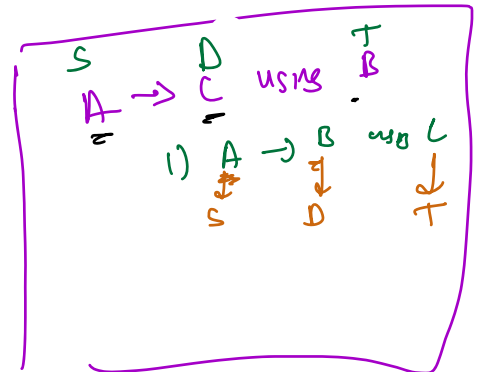
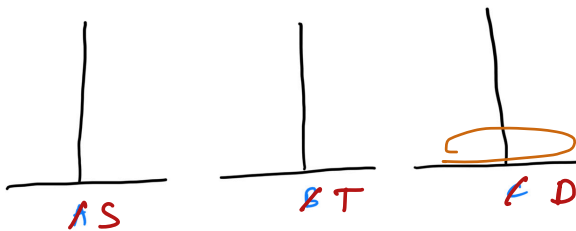
Pseudocode

- 1) Move $(N-1)$ disks from S to T using D
- 2) $S \rightarrow D$
- 3) Move $(N-1)$ disks from T to D using S

$N=2$



$N=1$



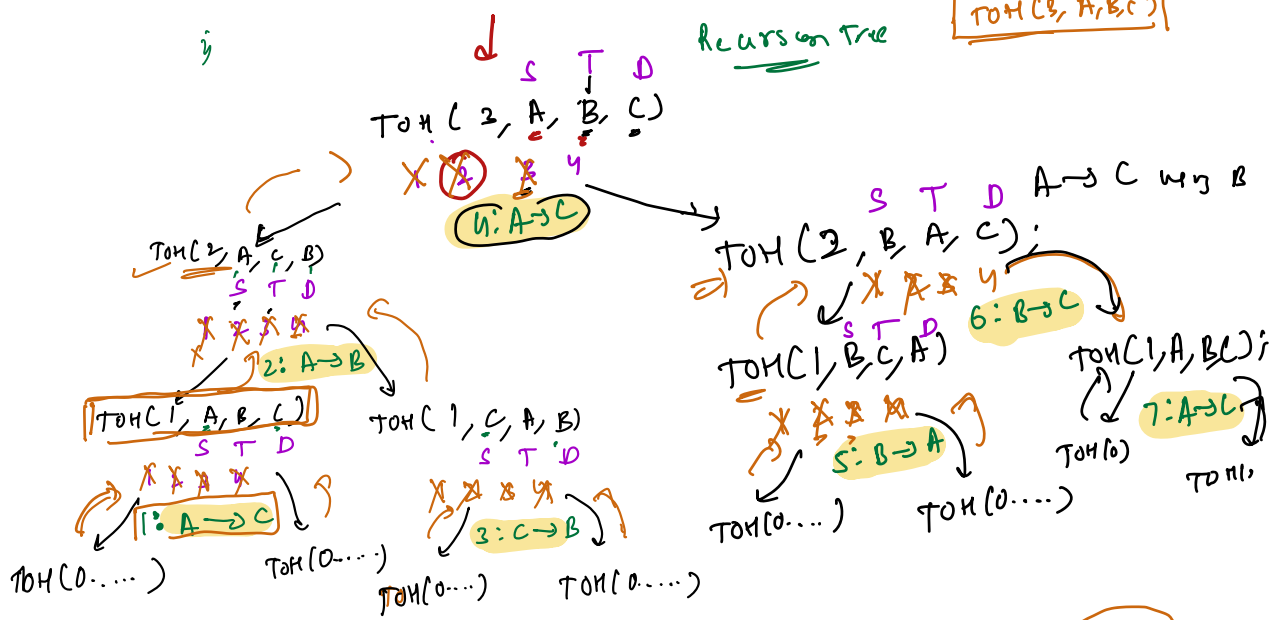
$A \rightarrow C$ using B

Assumption: $T_{OHC}(N, S, T, D)$ moves N discs from $S \rightarrow D$ using T

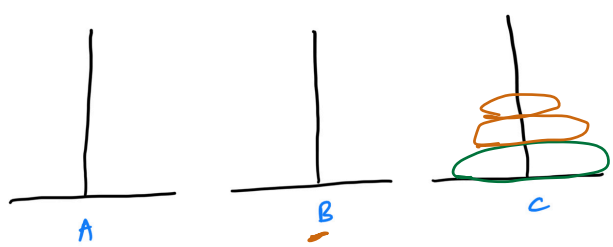
$$T(N) = T(N-1) + O(1) + T(N) \\ = 2T(N-1) + O(1) \\ O(2^N)$$

1. $T(N) = 0$ when $N=0$
2. $T(N-1, S, D, T);$
3. $print("S \rightarrow D");$
4. $T(N-1, T, S, D);$

$T_{OHC}(2, A, C, B)$
 $T_{OHC}(2, A, B, C)$

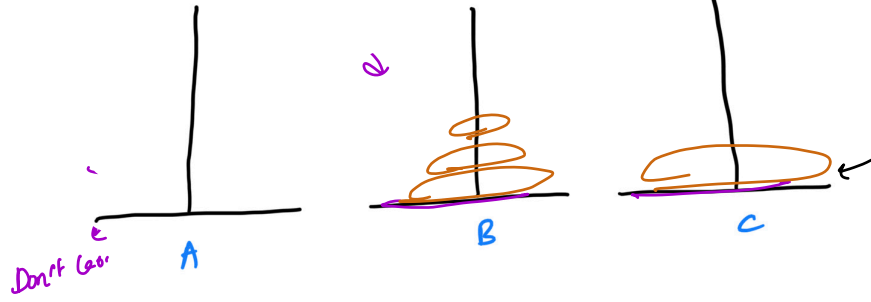


$$1 + 2^1 + 2^2 + \dots + 2^N \Rightarrow O(2^{N+1})$$



- $A \rightarrow C$
- $A \rightarrow B$
- $C \rightarrow D$
- $A \rightarrow C$
- $B \rightarrow A$
- $B \rightarrow C$
- $A \rightarrow C$

9 mins



A → C using B

Question: No of valid parentheses
 Given N, find no of valid parentheses
 using N pairs of brackets

N=1 () [1]
 N=2 () () , (() [2]
 N=3 () () () (() () ((()) [5]
 () (()) (() ()

Length of parentheses: 2N

C — — — — — — — — —
 1 2 3 4 5 2N-1 2N

Observation

- 1) length: 2N
- 2) 1st pos is always '('
- 3) corresponding closing bracket would only be at even position

(ans:-

0 pairs (N-1)
C) — — — — — — — —
 1 2 3 4 5 2N-1 2N
 => Find No. of parentheses with (N-1) pairs of brackets
F(0) * F(N-1)

Case 2:

Diagram illustrating the recursive step for calculating the number of ways to climb stairs:

The diagram shows a sequence of steps labeled 1, 2, 3, ..., 2N-1, 2N. A green line connects step 1 to step 2, and a red line connects step 2 to step 3. A green arrow points from step 1 to step 2, labeled "1 pair" and "x ways". A red arrow points from step 2 to step 3, labeled "1 pair" and "y ways".

Below the diagram, the text $F(1) \times F(N-2)$ is written, indicating the recursive formula for the number of ways to climb stairs.

Case 3:

$F(2) \times F(N-3)$

Diagram illustrating the recursive step for Case 3:

Input sequence: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Step 1: Pairing 1 and 2, 3 and 4, 5 and 6, 7 and 8, 9 and 10. This results in 5 pairs.

Step 2: The sequence is reduced to 1, 3, 5, 7, 9.

Step 3: Pairing 1 and 3, 5 and 7, 9 and 10. This results in 3 pairs.

Step 4: The sequence is reduced to 1, 5, 9.

Step 5: Pairing 1 and 5, 9 and 10. This results in 2 pairs.

Step 6: The sequence is reduced to 1, 9.

Step 7: Pairing 1 and 9. This results in 1 pair.

Step 8: The sequence is reduced to 1.

Final result: 1.

(ans 4: $f(3) \times f(N-4)$)

Diagram illustrating the recursive process for calculating the Fibonacci sequence:

- The sequence is represented as a series of terms: $\frac{C}{1}, \frac{1}{2}, \frac{1}{3}, \dots, \frac{1}{2N-1}, \frac{1}{2N}$.
- The term $\frac{1}{2N}$ is labeled "open" with a downward arrow.
- A bracket under the terms from $\frac{1}{2}$ to $\frac{1}{2N-1}$ is labeled "(N-1) part".
- Below the diagram, the recursive formula is given: $F(i) \times F(N-i-1)$.
- Indices are specified: $i=0$ to $N-i-1$ and $i=1$.

$$F(N) = F(0) \times F(N-1) + F(1) \times F(N-2) + F(2) \times F(N-3) + \dots + F(N-1) \times F(0)$$

Assumption: $F(N)$ returns no. of balanced parentheses using N pairs

Main Logic:

```

ans = 0;
for (i = 0; i < N; i++)
    ans += F(i) * F(N-i-1);

```

Base Condition:

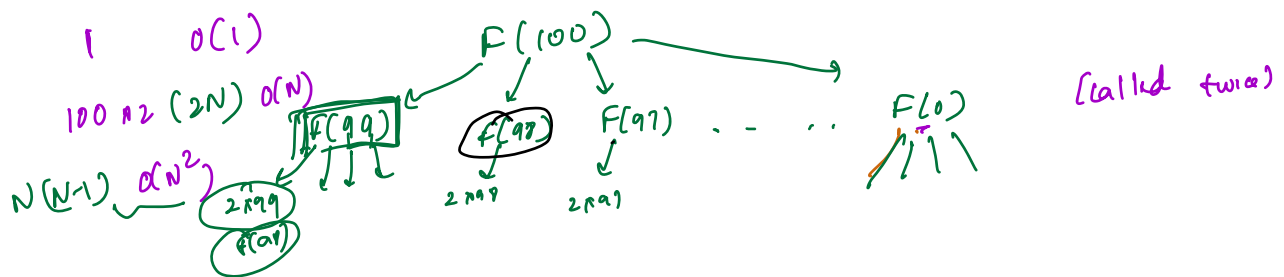
```

if (N == 0)
    return 1;

```

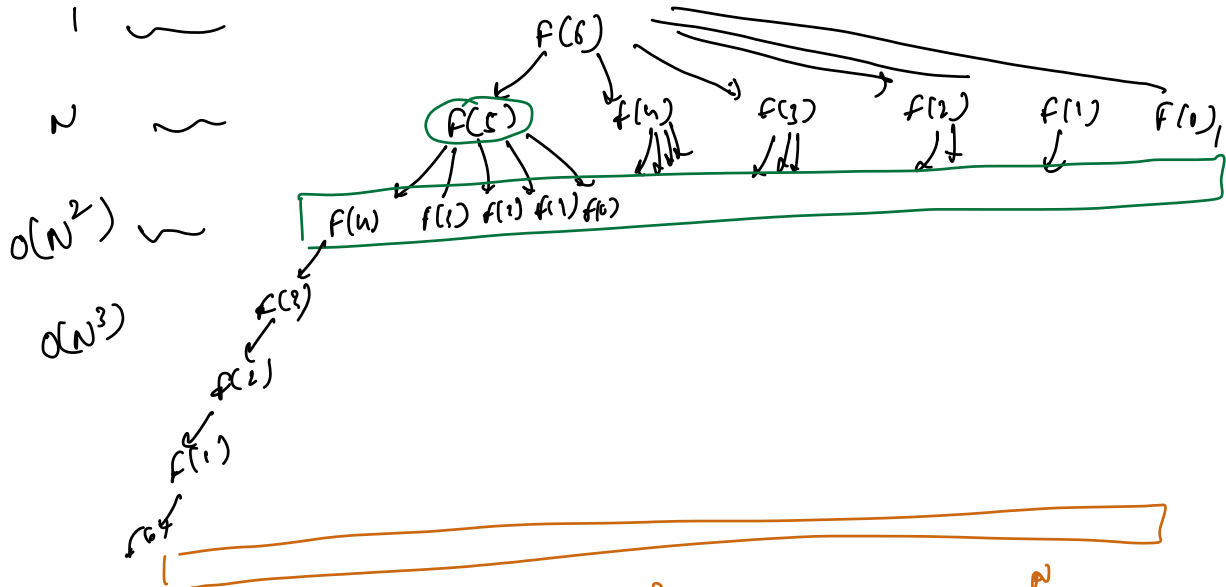
$$F(N) = F(0) \cdot F(N-1) + F(1) \cdot F(N-2) + F(2) \cdot F(N-3) + \dots + F(N-1) \cdot F(0)$$

↓ Catalan Number = $\frac{2N C_N}{N+1}$



$$2[N-1 + N-2 + N-3 + \dots + 0]$$

$$\frac{2[(N-1)N]}{2} = N(N-1)$$



$$1 + N + N^2 + N^3 + \dots + N^N$$

$$\frac{a(r^k - 1)}{r - 1} = \frac{1(N^N - 1)}{N - 1} = O(N^N)$$

Exponential

DP: $O(N^N) \rightarrow O(N^2)$

$$2^N C_N \Rightarrow \frac{2N!}{N! \times N!} \Rightarrow O(N)$$

$O(N)$

Master's Theorem

$$T(N) = a \cdot T\left(\frac{N}{b}\right) + O(N^d) \quad \begin{matrix} \downarrow & \downarrow & \downarrow \\ a > 0 & b > 1 & d \geq 0 \end{matrix}$$

Case 1:	$b^d > a$:	$O(N^d)$	}
Case 2:	$b^d = a$:	$O(N^d \log_b(N))$	
Case 3:	$b^d < a$:	$O(N^{\log_b a})$	

Ex: $T(N) = T(N/3) + O(1)$
 1. $T(N/3) + O(N^0)$

$$\begin{matrix} a = 1 \\ b = 3 \\ d = 0 \end{matrix}$$

$$b^d = 3^0 = 1$$

Case 2: $= O(N^d \log_b N)$

$$= O(N^0 \cdot \log_{3/2}^N)$$

$$= O(\log_{3/2}^N)$$

Ex2:

$$T(N) = 2T\left(\frac{N}{2}\right) + O(1) \quad \checkmark$$

$$a = 2$$

$$b = 2$$

$$d = 0$$

$$b^d = 2^0 = 1$$

$$b^d < a$$

$$O(n^{\log_b a})$$

$$= O(n^{\log_2^2})$$

$$= O(n)$$

Case 3:

Ex3: $T(N) = 2T\left(\frac{N}{2}\right) + O(N)$ [Merge Sort]

$$a = 2$$

$$b = 2$$

$$d = 1$$

$$b^d = 2^1 = 2$$

$$a = b^d$$

Case 2: $O(n^d \log_b n)$

$$= O(n^1 \log_2 n)$$

$$= O(n \log n)$$

Sorting? → Sunday Morn IST / Sat Night EST
 → Mon Morn IST / Sun Night

Sorting 2 →

Wed	Morning	IST / Tue	Night	EST
Tue	Morning	IST / Mon	Night	EST

- Insertion
- Selection
- Bubble sort
- Merge sort
- Quick sort
- Count sort
- Bucket sort
- Radix sort

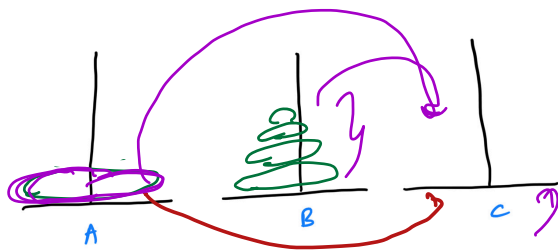
```
int sum(int N) {
    if (N == 0) return 0;
    return N + sum(N-1);
}
```

↓
last operation

```
int sum(int N, int r) {
    if (N == 0) return r;
    r = N + r;
    return sum(N-1, r);
}
```

Doubt
 Last ch.

N-1, r



- 1) Move from $A \rightarrow B$ using C
- 2) Move from $B \rightarrow C$ using A

- ✓ 1) $A \rightarrow B$ using C
- 2) $\text{print}(A \rightarrow B)$
- 3) $B \rightarrow C$ using A