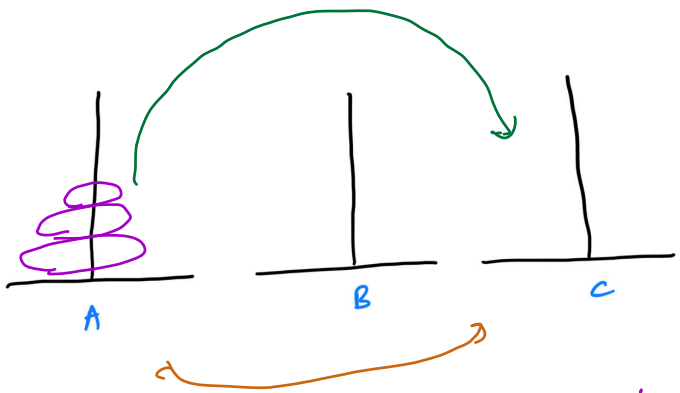
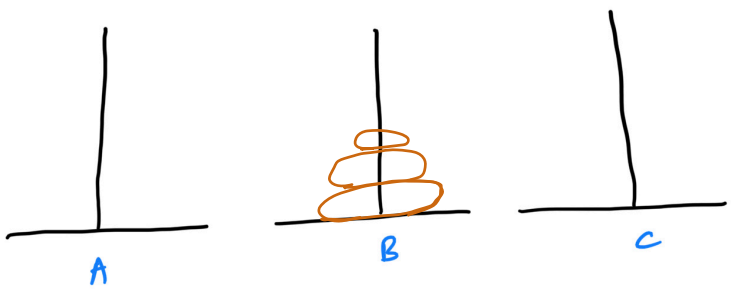
Towers of Hanoi



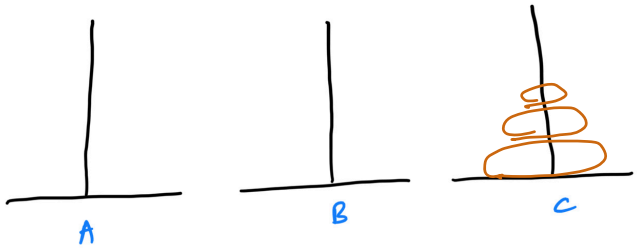
Rules

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

N = 3



A → B
A → C
B → C
A → B
C → A
C → B
A → B

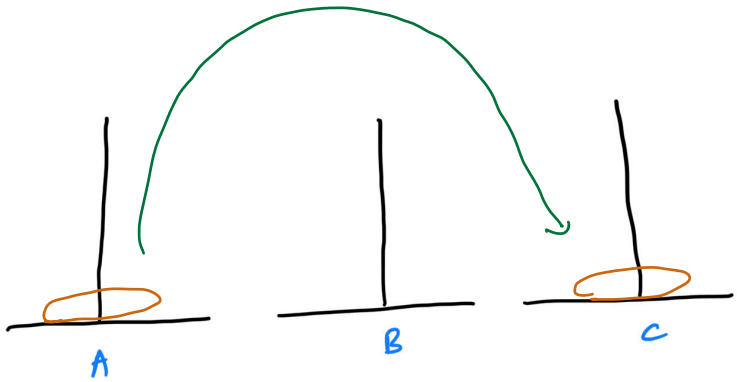


A → C
A → B
C → B
A → C

B → A
B → C
A → C

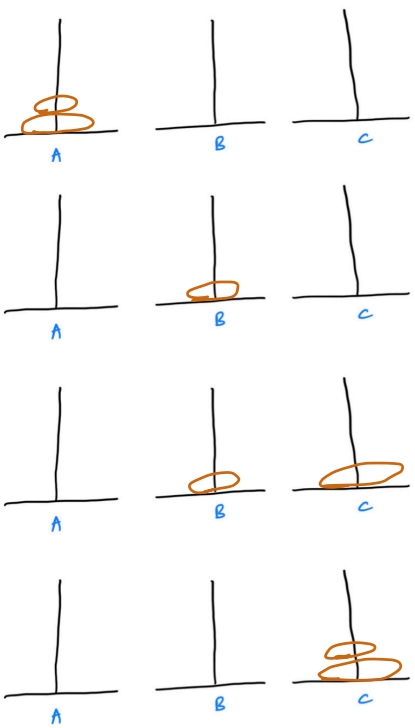
N discs, Move from Tower A to C and print all the steps

N = 1



A → C

N = 2

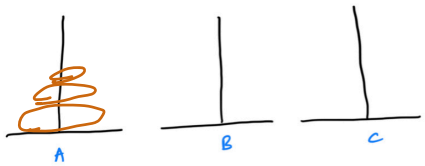


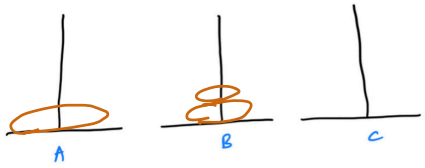
A → B

A → C

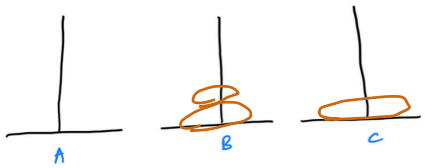
B → C

N=3





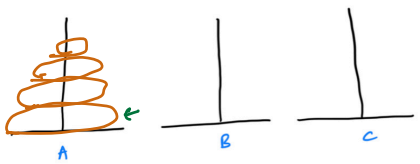
A → C
A → B
C → B



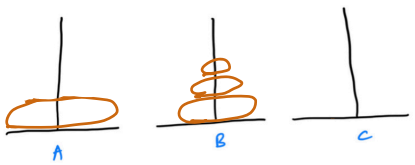
A → C

B → A
B → C
A → C

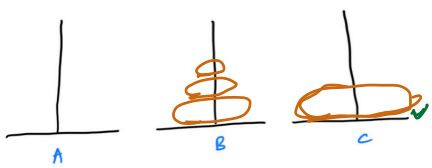
N = 4



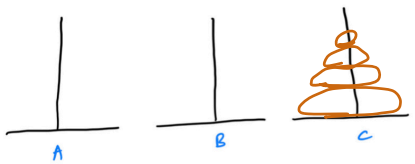
Step1: Move 3 discs from A → B
A → B, A → C, B → C
A → B
C → A, C → B, A → B



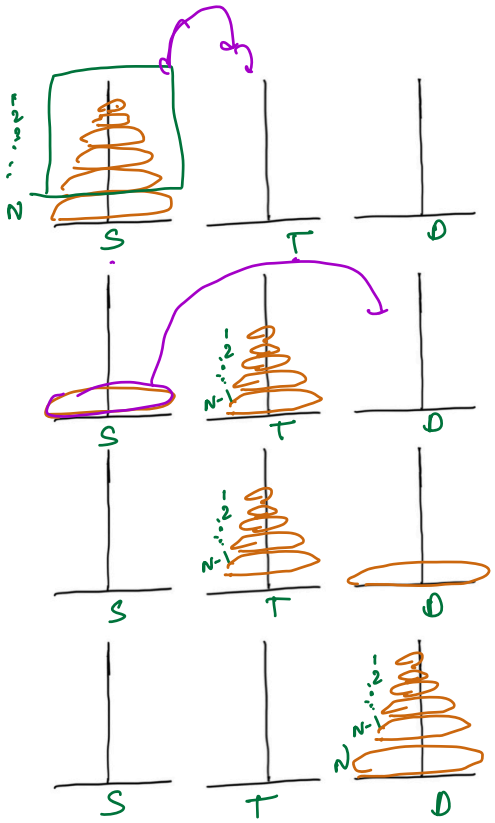
Step2: Move longest disc from A → C
A → C



Step3: Move 3 discs from B → C
B → C, B → A, C → A
B → C
A → B, A → C, B → C



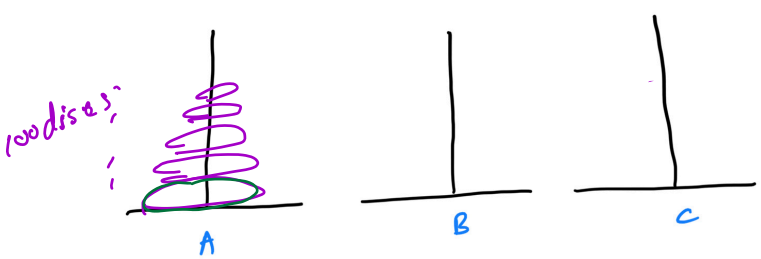
Problem: Move N discs from S to D using T



step1: Move (N-1) discs from S → T using D

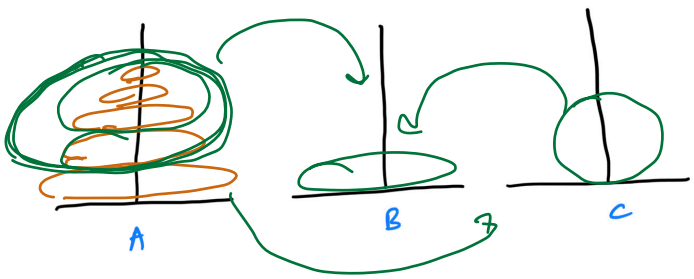
step2: Move S → D

step3: Move (N-1) discs from T → D using S



-> Move 100 discs from A → B

↳ Move 99 discs from A → C



Move 5 discs from A→C

1) Move 4 discs from A→B
2) A→C
3) Move 4 discs from B→C

Src Dst Temp

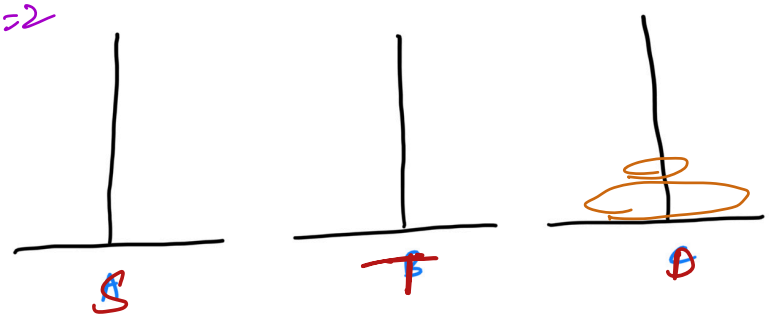
S to T using D

Pseudocode

1) Move (N-1) discs from S to T using D
2) S → D
3) Move (N-1) discs from T to D using S

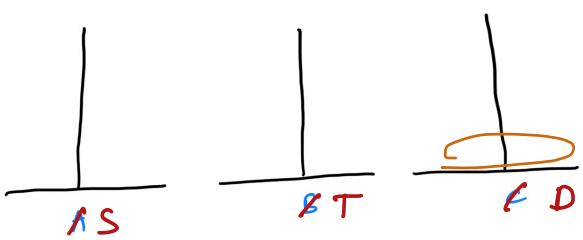
S D T
A → C using B
1) A → B using C
S D T

N=2



S T D
F(A, B, C)
F(A, C, B)
S D T

N=1



A→C using B

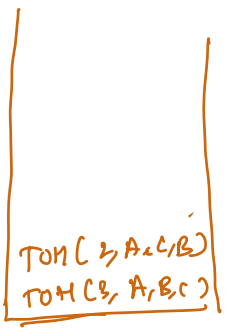
Assumption: TOH(N, S, T, D) moves N discs from S → D using T

A B C

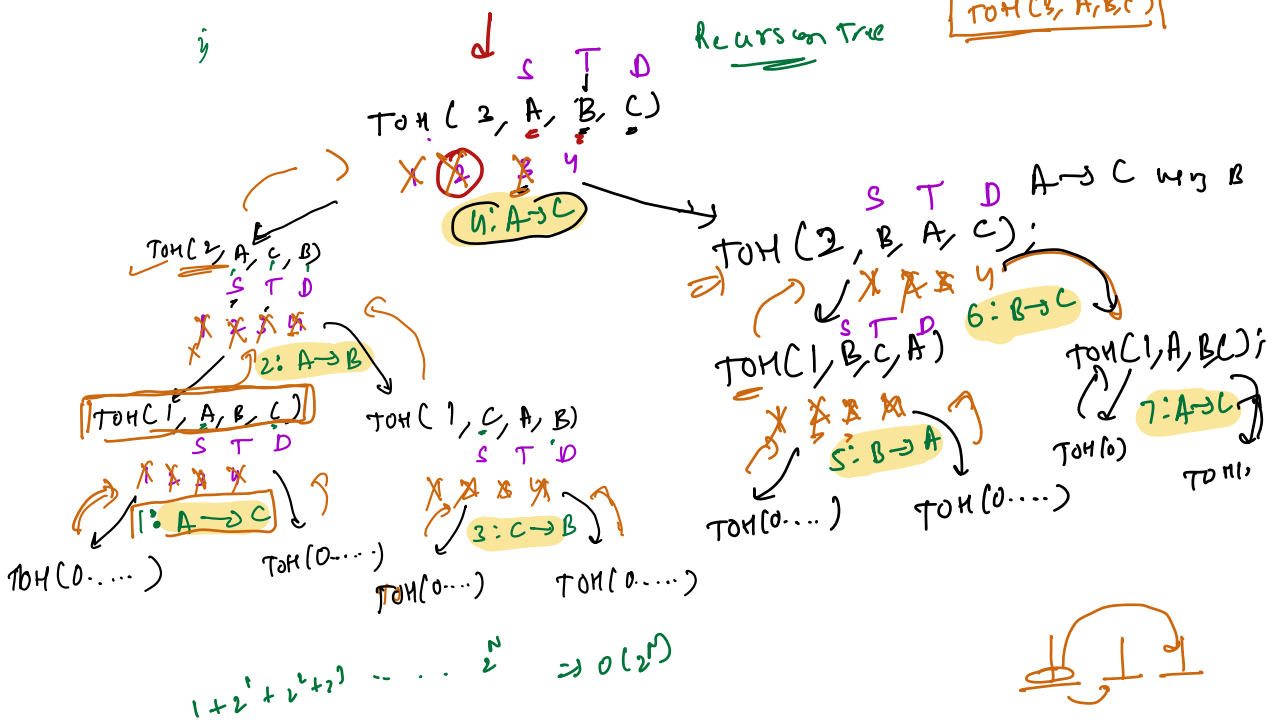
```
void TOH(int N, char S, char T, char D) {
1.  if(N == 0) return;
2.  TOH(N-1, S, D, T);
3.  print("S -> D");
4.  TOH(N-1, T, S, D);
}
```

Src: S, Dest: T, Temp: D

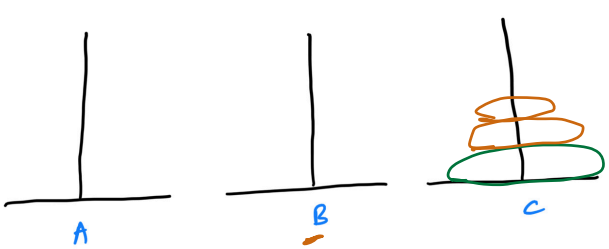
T(N) = T(N-1) + O(1) + T(N-1)
= 2T(N-1) + O(1)
$O(2^N)$



Recursion Tree



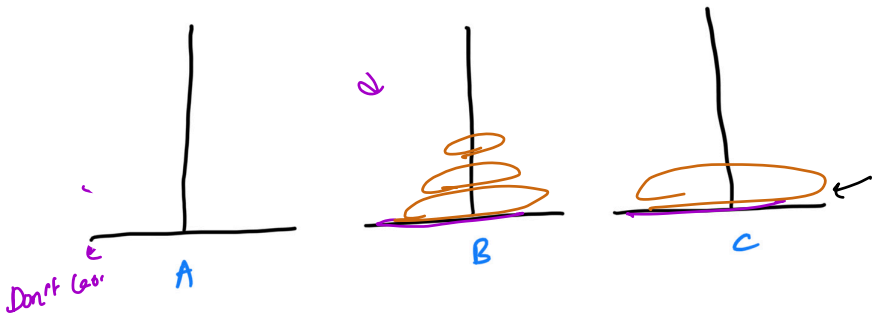
$1 + 2^1 + 2^{1+1} \dots 2^N \Rightarrow O(2^N)$



- A→C
A→B
C→B
A→C

B→A
B→C
A→C

9 mins



Don't Go.

A → C using B

Question: No of valid parentheses

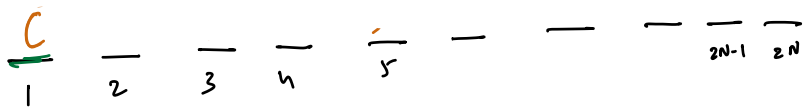
Given N, find nor of of valid parenthesis using N pairs of brackets (i)

N = 1 () [1]

N = 2 ()(), (()) [2]

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

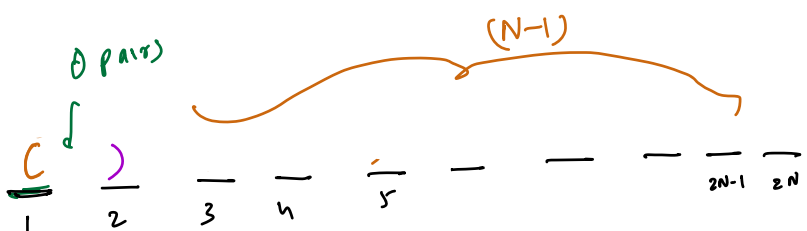
Length of Parenthesis: 2N



Observation

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

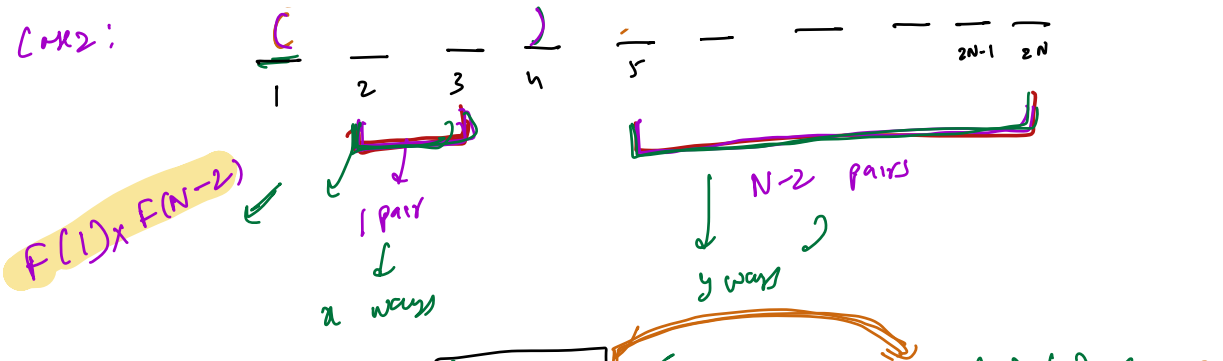
Case1:



=> Find No. of parenthesis with (N-1) pairs bracket

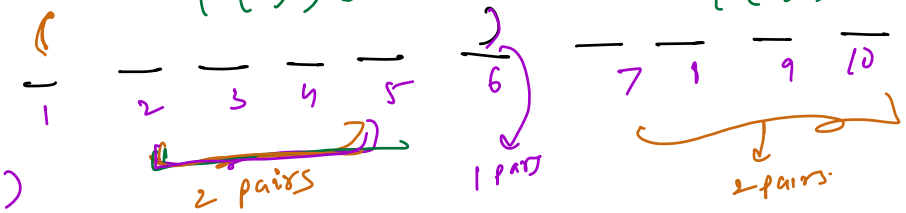
F(0) x F(N-1)

Case2:



F(1) x F(N-2)

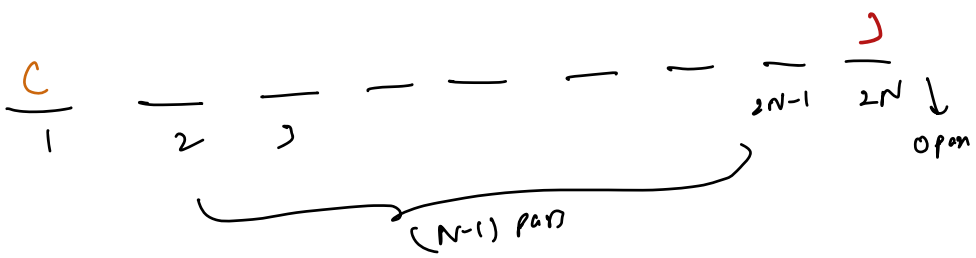
Case3:



F(2) x F(N-3)

Case4: F(3) x F(N-4)

CaseN:



$\sum_{i=0}^{N-i-1}$ $\sum_{i=1}^{}$ F(i) x F(N-i-1)

$F(N) = F(0) \times F(N-1) + F(1) \times F(N-2) + F(2) \times F(N-3) + F(3) \times F(N-4) + \ldots + F(N-1) \times F(0)$

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

Main Logic:

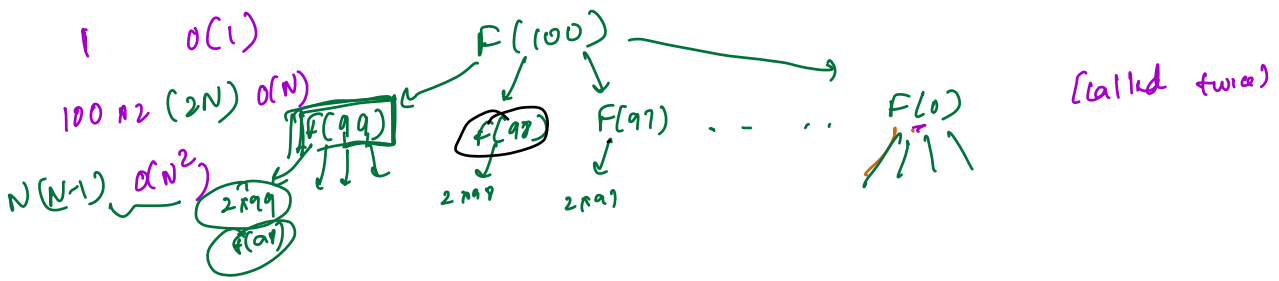
```
ans = 0;
for(i=0; i<N; i++)
    ans += F(i) * F(N-i-1);
```

Base condition:

```
if(N == 0)
    return 1;
```

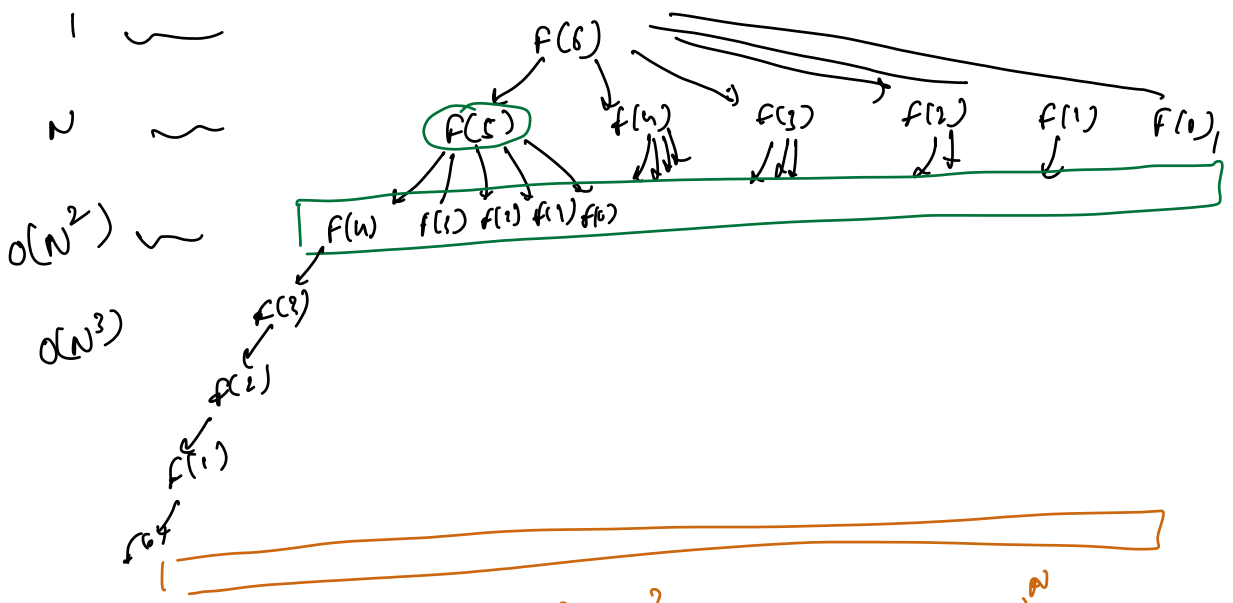
$F(N) = F(0)\cdot F(N-1) + F(1)\cdot F(N-2) + F(2)\cdot F(N-3) + \ldots + F(N-1)\cdot F(0)$

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



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

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



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

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

Exponential

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

$O(N)$

$2N_C{}_N \Rightarrow \frac{2N!}{N! \times N!} \Rightarrow O(N)$

$O(N)$ $O(N)$

Master's Theorem

$$T(N) = a \cdot T\left(\frac{N}{b}\right) + O(N^d)$$

$a > 0$, $b > 1$, $d \geq 0$

Case 1: $b^d > a$: $O(n^d)$

Case 2: $b^d = a$: $O(n^d \cdot \log_b(n))$

Case 3: $b^d < a$: $O(n^{\log_b a})$

Ex: $T(N) = T(2N/3) + O(1)$

$= 1 \cdot T(N/(3/2)) + O(N^0)$

$a = 1$

$b = 3/2$

$d = 0$

$b^d = (3/2)^0 = 1$

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

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

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

Ex2: $T(N) = 2T(\frac{N}{2}) + O(1)$ ✓

$a = 2$
$b = 2$
$d = 0$

$b^d = 2^0 = 1$

$b^d < a$

Case 3: $O(n^{\log_b a})$

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

$= O(n)$

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

$a = 2$
$b = 2$
$d = 1$

$b^d = 2^1 = 2$

$a = b^d$

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

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

$= \Theta(n \log_2 n)$

Sorting1 → Sunday Morning IST / Sat Night EST
→ Mon Morning IST / Sun Night

Sorting2 → 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

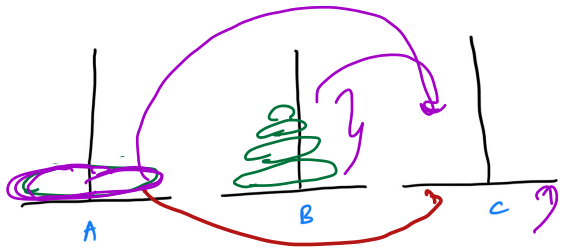
```
int sum(int N){
    if(N==0) return 0;
    return N + sum(N-1);
}
```

last operation

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

r = N+r, a, b, c

Doubt
Last class.



1) Move from A → B using C

2) Move from B to C using A

1) A → B using C
2) Print(A →)
3) B → C using A