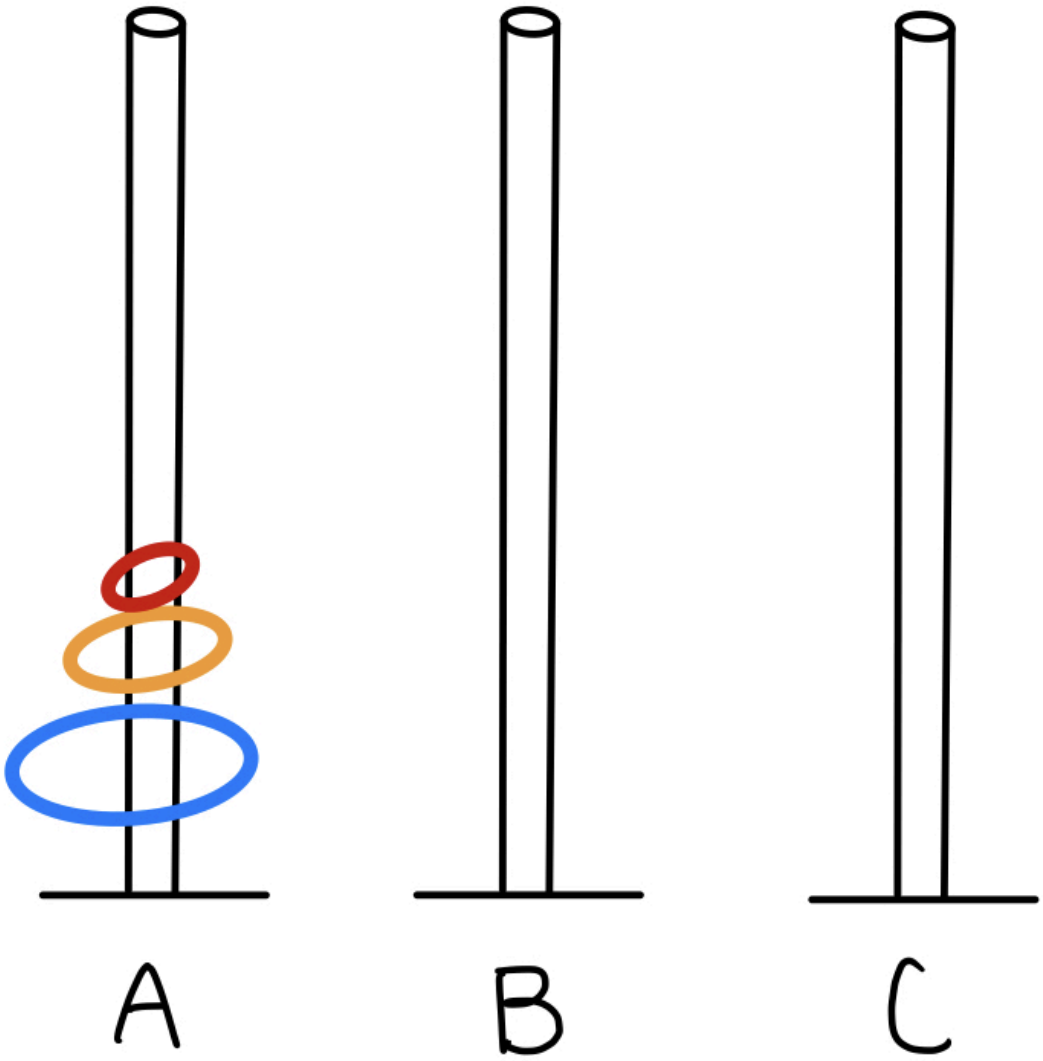
Recursion

1. Tower of Hanoi



題目

給定整數 `N`，代表從最左柱 `A` 將 `N` 個圓盤移到最右柱 `C`，可使用中間柱 `B`。一次只能移動一個圓盤，且不可將較大的圓盤壓在較小的上面。請輸出：

1. 最少移動次數 `2^N - 1`。
2. 逐步移動序列，每行一筆，格式：`A → C`。

輸入格式

- 單一整數 `N` （`1 ≤ N ≤ 20`）。

輸出格式

- 第 1 行：最少移動次數。
- 之後每行：一次合法移動，格式 `X → Y`，其中 `X, Y ∈ {A, B, C}`。

範例

```
class Hanoi:
    def __init__(self):
        self.moves = []  # e.g., [("A","C"), ("A","B"), ...]

    def solve(self, n, src="A", aux="B", dst="C"):


                pass

if __name__ == "__main__":
    n = 7
    h = Hanoi()
    h.solve(n, "A", "B", "C")

    total = (1 << n) - 1
    print(total)
    for a, b in h.moves:
        print(f"{a} → {b}")
```

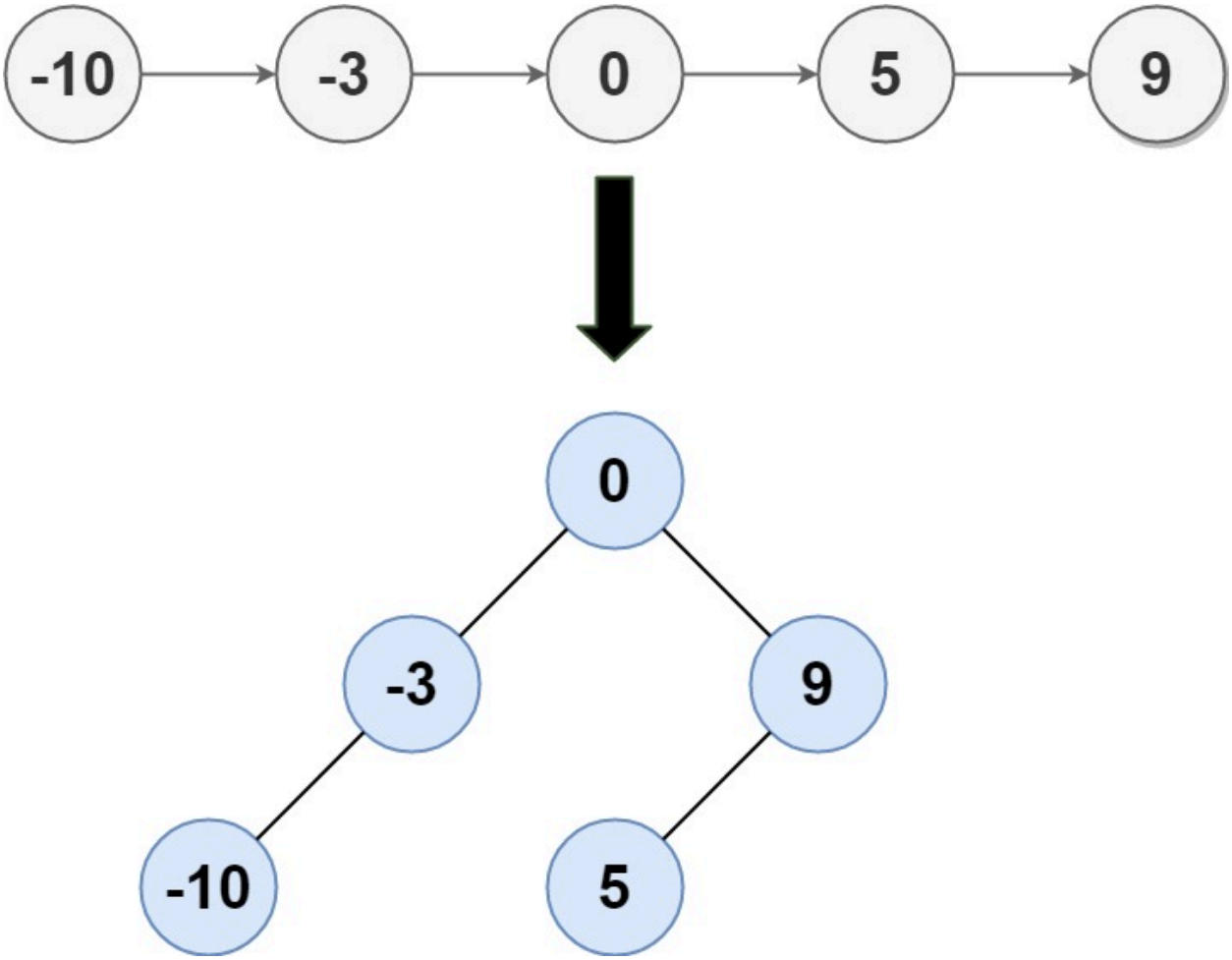
輸入

```
3
```

輸出

```
7
A → C
A → B
C → B
A → C
B → A
B → C
A → C
```

2.將排序鏈結串列轉換為平衡二元搜尋樹



給定一個**已排序（升冪）的單向鏈結串列的頭節點** `head`**，請將其轉換為高度平衡的二元搜尋樹（BST）**。

範例 1：

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
```

```
#         self.left = left
#         self.right = right
class Solution:
    def sortedListToBST(self, head: Optional[ListNode]) -> Optional[TreeNode]:
```

輸入：

`head = [-10, -3, 0, 5, 9]`

輸出：

`[0, -3, 9, -10, null, 5]`

說明：

其中一個可能的答案是 `[0, -3, 9, -10, null, 5]`，代表以下這棵高度平衡的 BST：

#注意! 這題輸出可能會依照不同方式而不同, 但是一定符合BST.