

알고리즘 재귀(1)

팀장 : 조영래

📌 프로그래머스

1. 1. 타켓 넘버

문제

n개의 음이 아닌 정수가 있습니다. 이 수를 적절히 더하거나 빼서 타겟 넘버를 만들려고 합니다. 예를 들어 [1, 1, 1, 1, 1]로 숫자 3을 만들려면 다음 다섯 방법을 쓸 수 있습니다.

사용할 수 있는 숫자가 담긴 배열 numbers, 타겟 넘버 target이 매개변수로 주어질 때 숫자를 적절히 더하고 빼서 타겟 넘버를 만드는 방법의 수를 return 하도록 solution 함수를 작성해주세요.

```
-1+1+1+1+1 = 3
+1-1+1+1+1 = 3
+1+1-1+1+1 = 3
+1+1+1-1+1 = 3
+1+1+1+1-1 = 3
```

- **입력 및 출력**

numbers	target	return
[1, 1, 1, 1, 1]	3	5

제한사항

- 주어지는 숫자의 개수는 2개 이상 20개 이하입니다.
- 각 숫자는 1 이상 50 이하인 자연수입니다.
- 타겟 넘버는 1 이상 1000 이하인 자연수입니다.

구현 아이디어

1. 모든 가능한 조합의 연산자 조합을 만든다.
2. 1번의 과정을 연산자 문자열을 배열에 담음으로써 수행한다.
3. 모든 연산자 조합에 대해 연산을 시행한다.

코드

```
def recursive(arr, m, operators_list):
    if m == len(arr):
        operators_list.append(list(arr)) # 직접 넣으면 operators_list에 입력된 arr이
        return                           # 모두 같아진다. 그래서 복사해서 넣음

    arr.append('+')
    recursive(arr, m, operators_list)
    arr.pop()
    arr.append('-')
    recursive(arr, m, operators_list)
    arr.pop()

def solution(numbers, target):
    answer = 0
    operators_list = []
    recursive([], len(numbers), operators_list)
    for operators in operators_list:
        result = 0
        for i in range(len(operators)):
            if operators[i] == '+':
```

```
                result += numbers[i]
            else:
                result -= numbers[i]
        if result == target:
            answer += 1
    return answer
```

---

1. 2. 전화번호 목록

문제

하노이 탑(Tower of Hanoi)은 퍼즐의 일종입니다. 세 개의 기둥과 이 기동에 꽂을 수 있는 크기가 다양한 원판들이 있고, 퍼즐을 시작하기 전에는 한 기둥에 원판들이 작은 것이 위에 있도록 순서대로 쌓여 있습니다. 게임의 목적은 다음 두 가지 조건을 만족시키면서, 한 기둥에 꽂힌 원판들을 그 순서 그대로 다른 기둥으로 옮겨서 다시 쌓는 것입니다.

1. 한 번에 하나의 원판만 옮길 수 있습니다.
2. 큰 원판이 작은 원판 위에 있어서는 안됩니다.

하노이 탑의 세 개의 기둥을 왼쪽 부터 1번, 2번, 3번이라고 하겠습니다. 1번에는 n개의 원판이 있고 이 n개의 원판을 3번 원판으로 최소 횟수로 옮기려고 합니다.

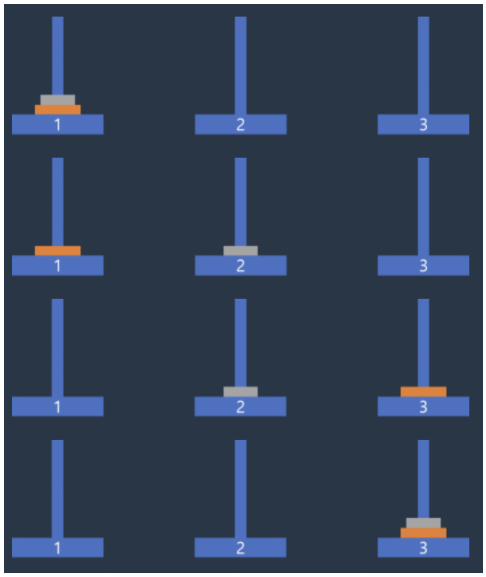
1번 기둥에 있는 원판의 개수 n이 매개변수로 주어질 때, n개의 원판을 3번 원판으로 최소로 옮기는 방법을 return하는 solution를 완성해주세요.

- **입력 및 출력**

n	result
2	[[1,2], [1,3], [2,3]]

제한사항

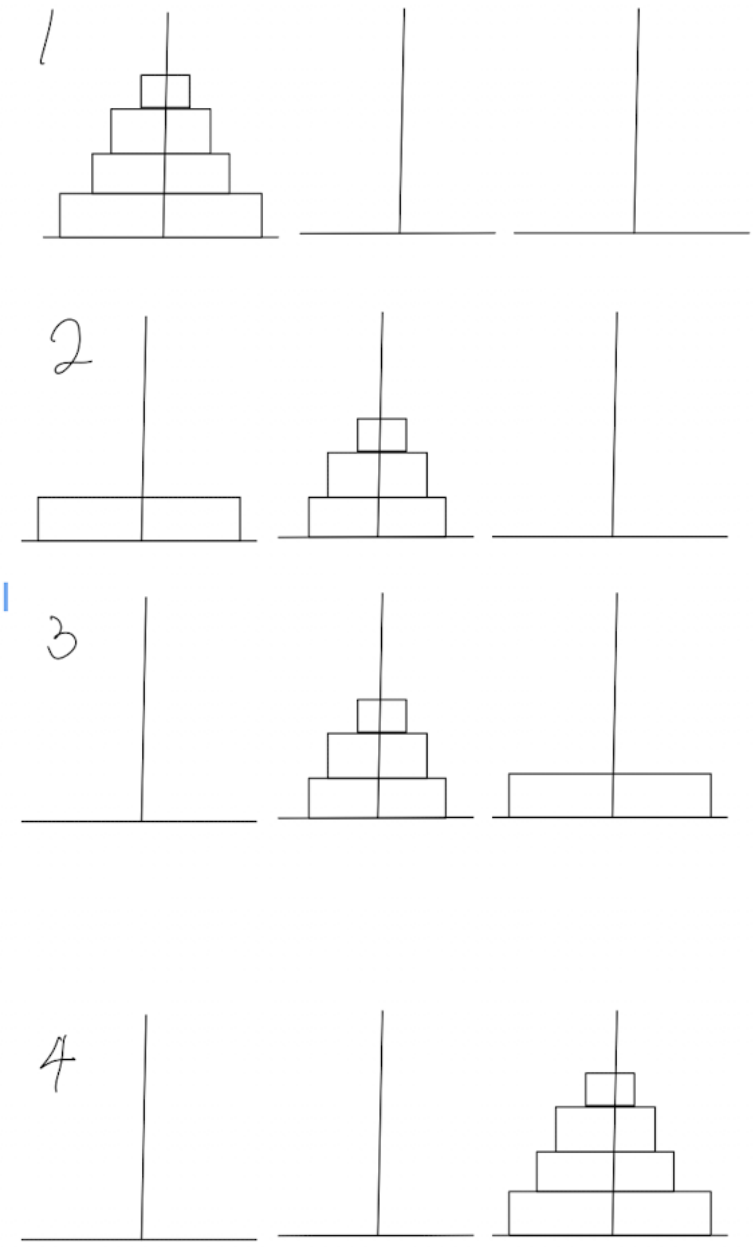
- n은 15이하의 자연수 입니다.



구현 아이디어

1. 처음 상태
    - 옮겨야 할 원판 : 4개
    - 1번 기둥 => 3번 기둥
2. 가장 큰 원판을 한 번만 옮기면 되는 상황
    - 옮겨야 할 원판 : 3개
    - 1번 기둥 => 3번 기둥
3. 가장 큰 원판을 옮긴 상황
4. 나머지 원판들을 가장 큰 원판 위로 옮기는 상황
    - 옮겨야 할 원판 : 3개
    - 2번 기둥 => 3번 기둥

위의 상황을 일반화 하면 n개의 원판을 가진 기둥을 옮기려면 n-1개의 원판을 옮기는 것을 한번 한 뒤 가장 큰 원판을 옮기고 n-1개의 원판을 옮겨야 한다.




```
def hanoi(n, f, b, t, result):
    if n == 1:
        result.append([f,t])
    else:
        hanoi(n-1, f, t, b, result)
        result.append([f, t])
        hanoi(n-1, b, f, t, result)

def solution(n):
    result = []
    hanoi(n, 1,2,3, result)
    return result
```