

컴프2 7주차 : 재귀의 이해

과제 목표 : 재귀를 이해하고 문제를 해결할 수 있다.

학번 : 201902722

이름 : 유형곤

1번 과제 하노이 탑

문제해결과정 및 코드 설명 :

하노이 탑을 처음 보았을 때, 우선 하노이탑의 규칙을 잘 모르기 때문에

<https://www.mathsisfun.com/games/towerofhanoi.html>

위 사이트에 들어가서 약 20분간 시뮬레이션 해보았습니다. 그랬더니 문제의 부분적인 구조가 보였습니다. 바로 출발지(from)에서 도착지(to)로 n개의 고리를 옮기려면 임시적으로 다른 장소(another)에 n-1개의 고리를 놓아둔 후, 1개의 고리를 from->to로 이동시키고, 다시 n-1개의 고리를 another->to로 이동시키는 것입니다. 그렇게 문제를 이해하고 나니 이 문제를 재귀적으로 풀 수 있겠다는 생각이 들었습니다.

```
static StringBuilder sb;
public static void hanoi(int from, int to, int n, int depth) {
    if(n == 1) {
        sb.append(depth + ": ").append((char) (from-1+'A') + " -> " + (char) (to-1+'A')).append("\n");
        return;
    }
    int another = 6 - from - to; //1 + 2 + 3 = 6
    hanoi(from, another, n-1, depth-1);
    hanoi(from, to, 1, depth);
    hanoi(another, to, n-1, depth-1);
}
```

<하노이 탑 소스코드>

그래서 소스코드를 보시면 기본단계($n = 1$)부분과 재귀적 단계로 나뉘어져 있는데,

기본 단계는 $n = 1$ 인 경우 A->C처럼 고리의 이동을 출력하는 것이고,

```
hanoi(from, another, n-1, depth-1);
```

```
hanoi(from, to, 1, depth);
```

```
hanoi(another, to, n-1, depth-1);
```

재귀적 단계는 처음 부분에서 설명 드린 것처럼 $n-1$ 개를 우선 옮기고, 1개를 옮긴 후, 다시 $n-1$ 개를 옮기도록 작성했습니다.

여기서 depth는 n 단계에서 수행했음을 말합니다. 다시 말하면

4단계는 3단계를 수행하고 (1~3단계에 해당), 블록을 1개 옮기고 (4단계에 해당), 다시 3단계를 수행한 것입니다. depth를 단계에 따라 나열해보면 다음과 같은 형태가 됩니다.

N = 2) 1 2 1

N = 3) 1 2 1 3 1 2 1

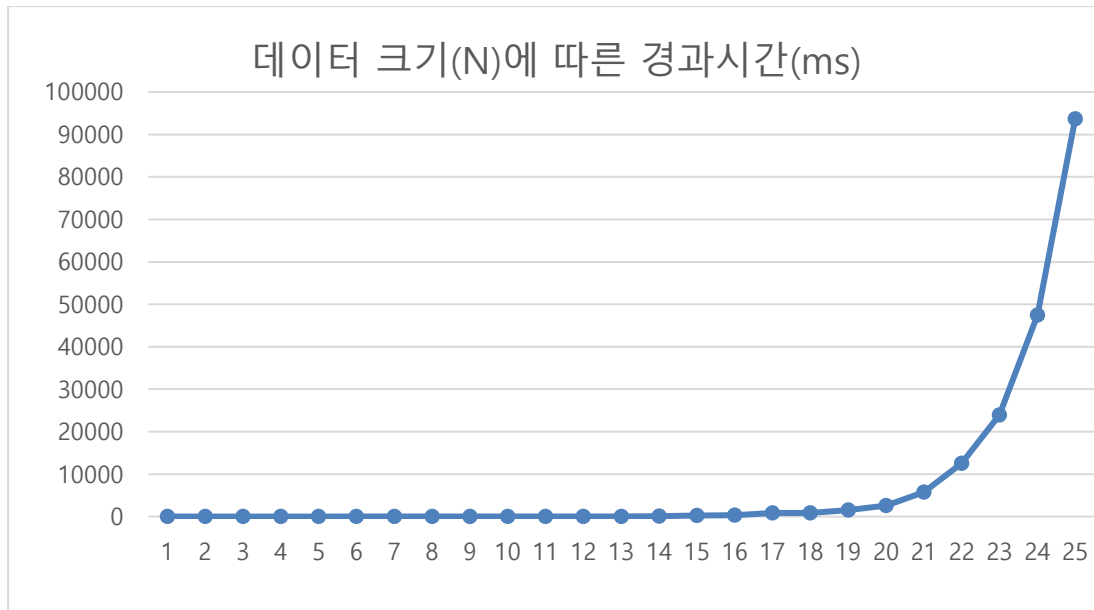
N = 4) 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1

```
Console
<terminated> Hanoi (1) [Java Application] C:\Program Files\Java\jre1.8.0_211\bin\javaw.exe (2019. 10. 18. 오전 11:43:05)
2 번째 하노이 탑
1: A -> B
2: A -> C
1: B -> C

3 번째 하노이 탑
1: A -> C
2: A -> B
1: C -> B
3: A -> C
1: B -> A
2: B -> C
1: A -> C
```

<수행결과>

output.csv를 excel에서 그래프로 그려본 결과 다음과 같이 나왔습니다.



2번 과제 : Palindrome 검사

문제해결과정 및 코드 설명 :

Palindrome은 앞으로 읽어도, 뒤로 읽어도 같은 문자열로 ana, level, 소주죌주소 등의 문자열을 말합니다. 이 문제도 부분적인 문제로 나눌 수 있습니다.

예를들어 level은, 첫 글자와 마지막 글자가 같고, eve가 회문이라면 회문입니다.

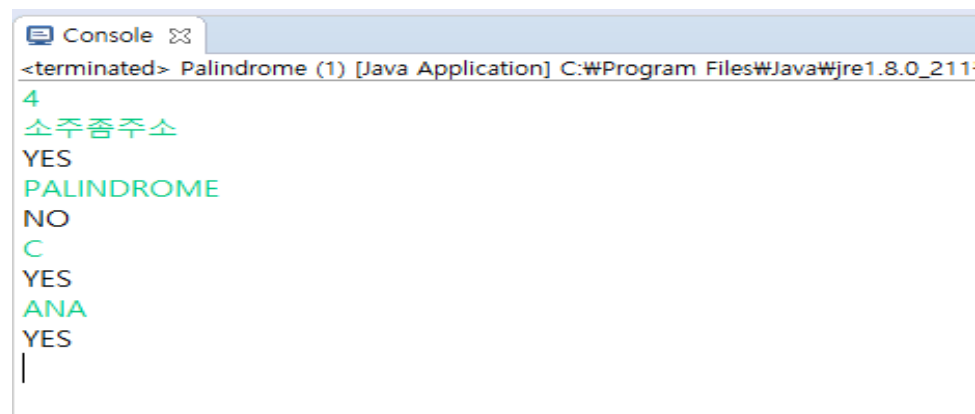
다시 eve는, 첫 글자와 마지막 글자가 같고, v가 회문이라면 회문입니다.

이런 방식으로 문제를 분할할 수 있으며, 재귀함수가 끝나게 되는 종료조건은

문자열의 길이가 1인 경우 true를, 길이가 2인 경우 두 문자가 같은지의 여부를 반환하여 문제를 해결합니다.

```
public static boolean isValidPalindrome(String str, int s, int e){
    int len = e-s + 1;
    if(len == 1) {
        return true;
    } else if(len == 2) {
        return (str.charAt(s) == str.charAt(e));
    } else {
        if(str.charAt(s) == str.charAt(e)) {
            return isValidPalindrome(str, s+1, e-1);
        } else {
            return false;
        }
    }
}
```

참고로, isValidPalindrome 메서드의 int s, int e는 문자열의 시작과 끝 인덱스를 의미합니다. 처음에는 substring으로 구현하려다가, substring은 매번 새로운 문자열을 만들기 때문에 인덱스를 활용한 방법으로 해결했습니다.



```
<terminated> Palindrome (1) [Java Application] C:\Program Files\Java\jre1.8.0_211\
4
소주죌주소
YES
PALINDROME
NO
C
YES
ANA
YES
|
```