재귀(Recursion)

목차

- 재귀란
- 분할 정복 알고리즘
- 하노이 탑
- 팁?

재귀란

- 자기 자신을 호출하는 함수

Q. 반복문이나 stack을 사용하면 해결할 수 있는데 사용하는 이유는?

A. 재귀적으로 코드를 구현했을 때, 코드가 깔끔한 경우가 있다.

A. 다른 사람들이 코드를 봤을 때, 편하게 이해할 수 있다.

- 대표적인 예제(팩토리얼)

→ 반복문 사용했을 경우

```
def Fact(num) :
    a = 1
    while (num > 1) :
        a = a * num
        num = num - 1
    num = a
    return num
print(Fact(20))
```

→ 재귀를 사용하는 경우

```
def Fact(num) :
    if num > 1 :
        return (num * Facto(num-1))
    else :
        return 1
print(Fact(20))
```

- 주의할 점

1. 종료 조건을 잘 설정해야 한다.
2. 문제의 정의를 선언하는 부분을 잘 이해해야 한다.

분할 정복 알고리즘

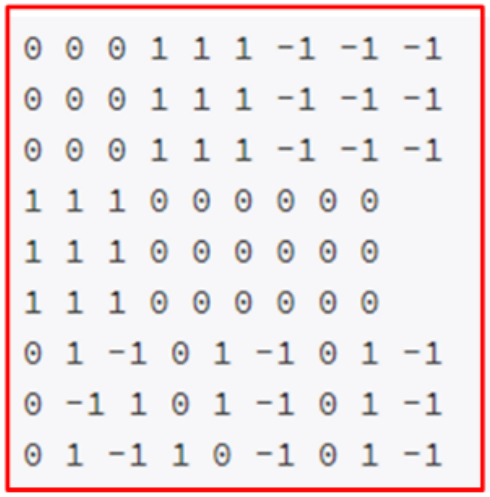
- 그대로 해결할 수 없는 문제를 작은 문제로 분할하여 문제를 해결하는 방법(알고리즘)

- 예제(백준 1780번)

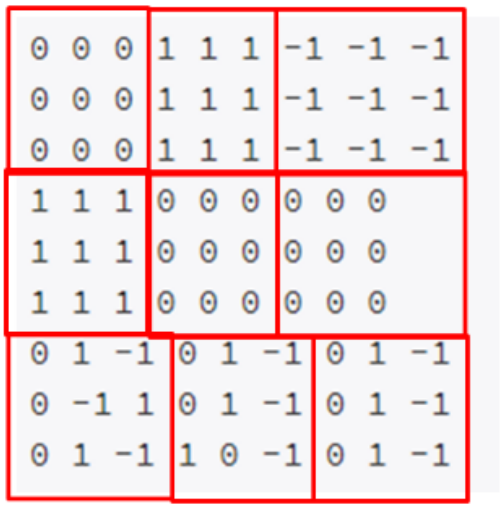
1780번: 종이의 개수

이와 같이 종이를 잘랐을 때, -1로만 채워진 종이의 개수, 0으로만 채워진 종이의 개수, 1로만 채워진 종이의 개수를 구해내는 프로그램을 작성하시오.

https://www.acmicpc.net/problem/1780



```
0 0 0 1 1 1 -1 -1 -1
0 0 0 1 1 1 -1 -1 -1
0 0 0 1 1 1 -1 -1 -1
1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0
0 1 -1 0 1 -1 0 1 -1
0 -1 1 0 1 -1 0 1 -1
0 1 -1 1 0 -1 0 1 -1
```



```
0 0 0 1 1 1 -1 -1 -1
0 0 0 1 1 1 -1 -1 -1
0 0 0 1 1 1 -1 -1 -1
1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0
0 1 -1 0 1 -1 0 1 -1
0 -1 1 0 1 -1 0 1 -1
0 1 -1 1 0 -1 0 1 -1
```

하노이 탑

- 하노이 탑: 세 가지의 기둥이 존재하고, 다양한 크기의 n개의 원판이 하나의 기둥에 작은 것들이 위에 있도록 순서대로 쌓여 있을 때, 다음 조건을 만족하며 다른 기둥으로 옮기는 문제

1. 한 번에 하나의 원판만 옮길 수 있다.
2. 큰 원판이 작은 원판 위에 있을 수 없다.

- 해답

1. N개의 원판을 옮기기 위해서는 N-1개의 원판을 이웃한 기둥으로 옮겨야 한다.
2. 남아 있는 가장 큰 원판이 최종 목적 기둥으로 옮겨진다.
3. 1번을 하고 남아 있는 N-1개의 원판을 가지고 1, 2을 반복한다.

팁

- Recursion Error 발생할 때!

→ 이유: 파이썬의 기본 재귀함수 깊이가 1000으로 한도가 설정되어 있기 때문이다.

- 해결방법

→ sys.setrecursionlimit(깊이) 메소드를 사용하면 재귀함수의 깊으를 원하는 만큼 설정할 수 있다.