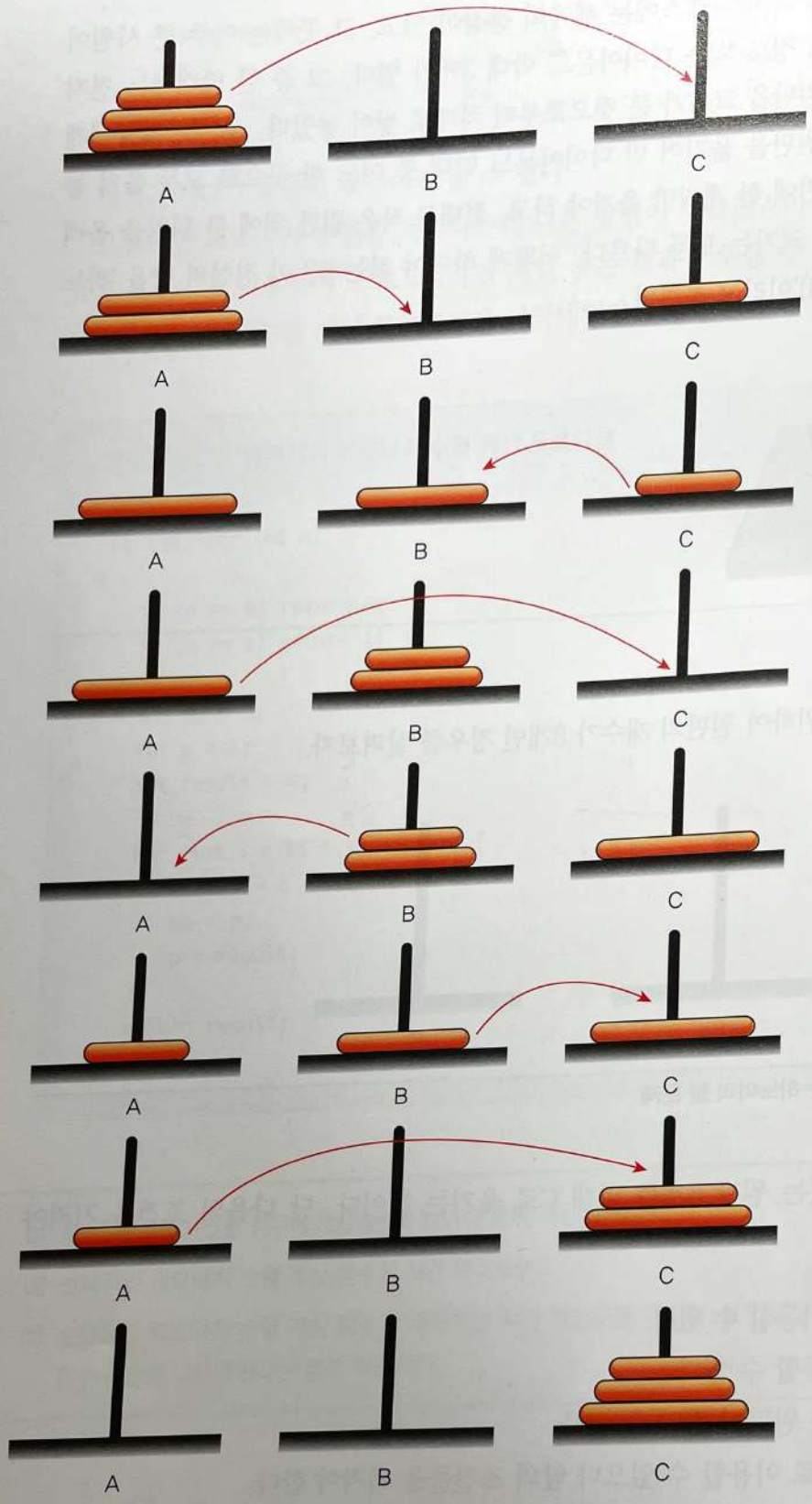
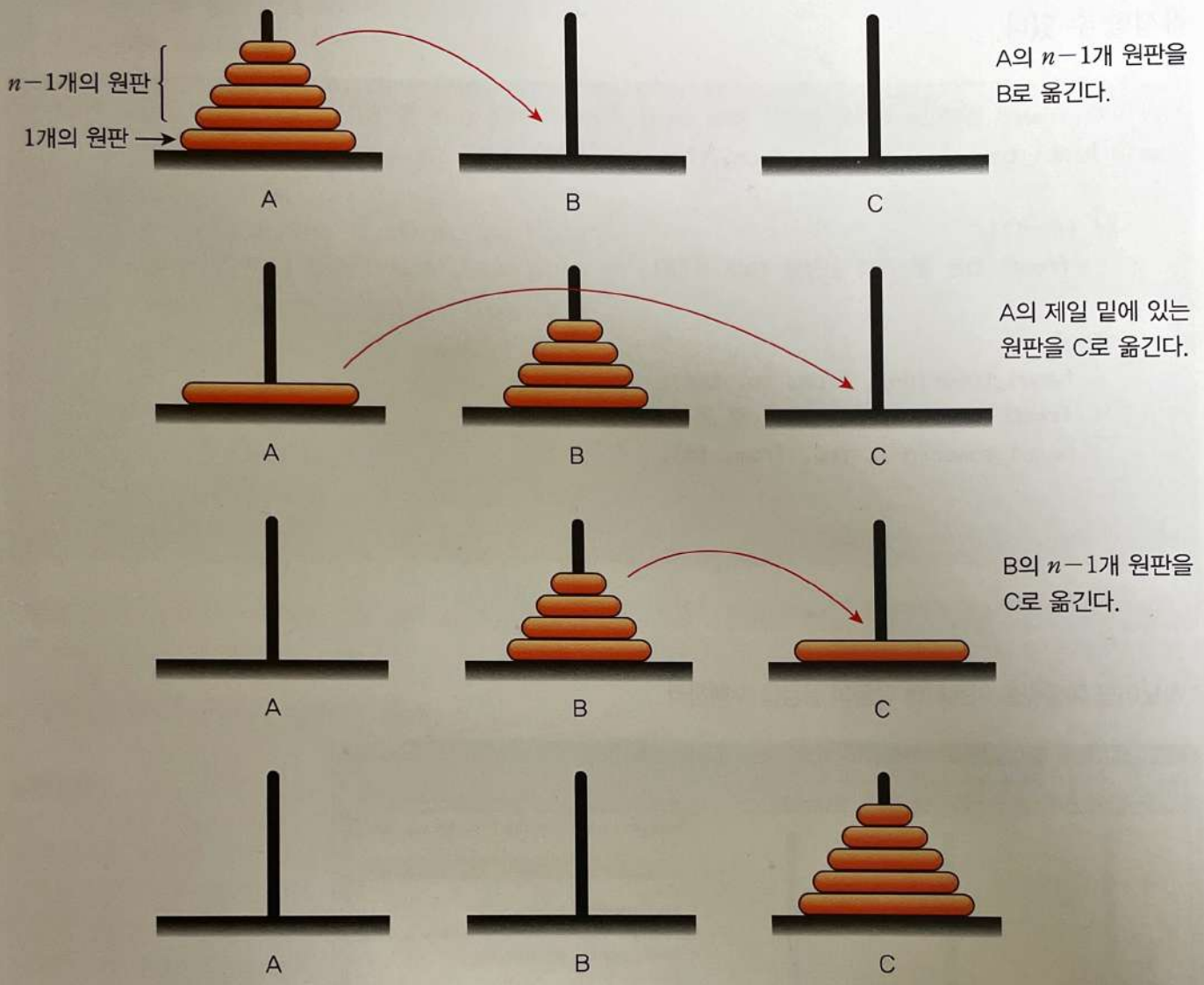


[그림 2-14] 3개의 원판을 가지는 하노이의 탑 문제의 해답

4개의 원판이 있는 경우에는 조금 더 복잡해진다. 더 나아가 n개의 원판이 있는 경우를 해결하려면 상당히 복잡해진다.

이 문제는 순환적으로 생각하면 쉽게 해결할 수 있다. 순환이 일어날수록 문제의 크기가 작아져야 한다. 여기서의 문제의 크기는 이동하여야 하는 디스크의 개수가 된다. 다음과 같이 문제를

나누어 생각하여 보자. n개의 원판이 A에 쌓여있는 경우, 먼저 위에 쌓여 있는 $n-1$개의 원판을 B로 옮긴 다음, 제일 밑에 있는 원판을 C로 옮긴다. 이어서 B에 있던 $n-1$개의 원판을 C로 옮긴다. 자 이제 문제는 B에 쌓여있던 $n-1$개의 원판을 어떻게 C로 옮기느냐이다. 이 문제를 다음과 같이 알고리즘을 만들어서 생각하여보자.



[그림 2-15] n개의 원판을 가지는 하노이의 탑 문제의 해답

```
// 막대 from에 쌓여있는 n개의 원판을 막대 tmp를 사용하여 막대 to로 옮긴다.
void hanoi_tower(int n, char from, char tmp, char to)
{
    if (n == 1){
        from에 있는 한 개의 원판을 to로 옮긴다.
    }
    else{
        ① from의 맨 밑의 원판을 제외한 나머지 원판들을 tmp로 옮긴다.
        ② from에 있는 한 개의 원판을 to로 옮긴다.
        ③ tmp의 원판들을 to로 옮긴다.
    }
}
```

위의 의사 코드 중에서 ②은 1개의 원판을 이동하는 것이므로 매우 쉽고 문제는 $n-1$개의 원판을 이동하여야 하는 ①과 ③을 어떻게 해결하는냐 이다. 그러나 자세히 살펴보면 ①과 ③은 막대의 위치만 달라졌을 뿐 원래의 문제의 축소된 형태라는 것을 발견할 수 있다. 즉 ①은 to를 사용하여 from에서 tmp로 $n-1$개의 원판을 이동하는 문제이고 ③은 from을 사용하여 tmp에서 to로 $n-1$개의 원판을 이동하는 문제이다. 따라서 순환호출을 사용할 수 있어서 다음과 같이 다시 작성할 수 있다.

```
// 막대 from에 쌓여있는 n개의 원판을 막대 tmp를 사용하여 막대 to로 옮긴다.
void hanoi_tower(int n, char from, char tmp, char to)
{
    if (n==1){
        from에 있는 한 개의 원판을 to로 옮긴다.
    }
    else{
        hanoi_tower(n-1, from, to, tmp);
        from에 있는 한 개의 원판을 to로 옮긴다.
        hanoi_tower(n-1, tmp, from, to);
    }
}
```