



게임 자료구조와 알고리즘


-CHAPTER1_HANOI-

SOULSEEK



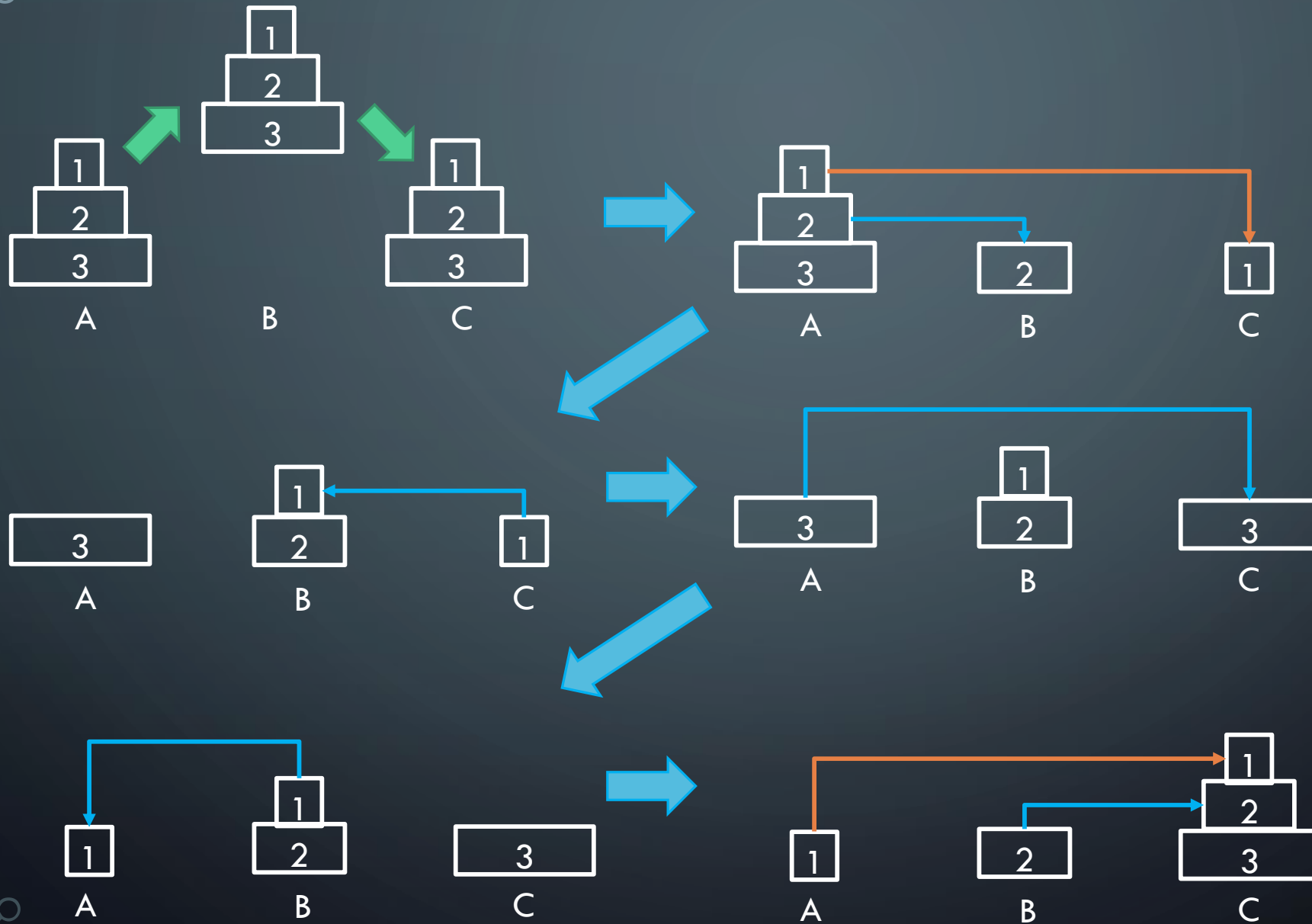
목차

1. 하노이 타워



하노이 타워

1. 하노이 타워



1. 하노이 타워

4개로 늘린다면?

- 1, 2, 3을 B로 옮기고 4를 C로 옮겨야 한다. – 이미 3개를 옮기는 패턴은 알고 있다.

그래서 공식패턴은?

- 작은 원반 3개(맨 아래에 있는 원반을 제외한)를 A에서 B로 이동
- 큰 원반(맨 아래 원반) 1개를 A에서 C로 이동
- 작은 원반(B로 옮겨진 원반) 3개를 B에서 C로 이동

공식을 일반화 해보자.(원반이 n 개일 때)

- 작은 원반 $n-1$ 개를 A에서 B로 이동
- 큰 원반 1개를 A에서 C로 이동
- 작은 원반 $n-1$ 개를 B에서 C로 이동

1. 하노이 타워

Code로 표현해보자.

- From에 꽂혀있는 num개의 원반을 by를 거쳐서 to로 이동

```
void HanoiTowerMove(int num, char form, char by, char to)
{
    num개의 원반을 by를 거쳐서(by)를 이용해서 from에서 to로 이동한다.
}
```

- 원반이 1개일 경우 한번 옮기면 끝나기 때문에 탈출 조건이 된다.

```
void HanoiTowerMove(int num, char form, char by, char to)
{
    if(num == 1)
    {
        cout << "원반1을 " << form << "에서 " << to << "로 이동" << endl;
    }
    else
    {
        // ...
    }
}
```

1. 하노이 타워

- 작은 원반 $n - 1$ 개를 **A**에서 **B**로 이동.

```
void HanoiTowerMove(int num, char form, char by, char to)
{
    if(num == 1)
    {
        cout << "원반1을 " << form << "에서 " << to << "로 이동" << endl;
    }
    else
    {
        //3단계 중 1단계
        HanoiTowerMove(num - 1, from, to, by);
    }
}
```

1. 하노이 타워

- 큰 원반 **1**개를 **A**에서 **C**로 이동, 작은 원반은 **n - 1**개를 **B**에서 **C**로 이동.

```
void HanoiTowerMove(int num, char form, char by, char to)
{
    if(num == 1)
    {
        cout << "원반1을 " << form << "에서 " << to << "로 이동" << endl;
    }
    else
    {
        //3단계 중 1단계
        HanoiTowerMove(num - 1, from, to, by);
        //3단계 중 2단계
        cout << "원반" << num << "을(를) " << from << "에서 " << to << "로 이동" << endl;
        //3단계 중 3단계
        HanoiTowerMove(num - 1, by, from, to);
    }
}
```