하노이탑(tower of hanoi)

하노이 탑은 **퍼즐**의 일종이다.(위키 백과 참조)

3개의 기둥과 이 기둥에 꽂을 수 있는 크기가 다양한 원판들이 있고,
퍼즐을 시작하기 전에는 한 기둥에 원판들이 작은 것이 위에 있도록 순서대로 쌓여 있다.

게임의 목적은 다음 두 가지 조건을 만족시키면서,
한 기둥에 꽂힌 원판들을 그 순서 그대로 다른 기둥으로 옮겨서 다시 쌓는 것이다.

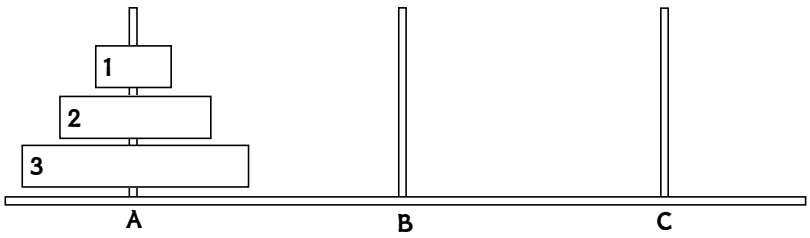
한 번에 하나의 원판만 옮길 수 있다.
큰 원판이 작은 원판 위에 있어서는 안 된다.

하노이탑 문제는 재귀호출을 이용하여 풀 수 있는 가장 유명한 예제 중의 하나이다.
그렇기 때문에 프로그래밍 수업에서 알고리즘 예제로 많이 사용한다.
일반적으로 원판이 n개 일 때, 2^n-1번의 이동으로 원판을 모두 옮길 수 있다.
2^n-1는 **메르센 수**라고 부른다.

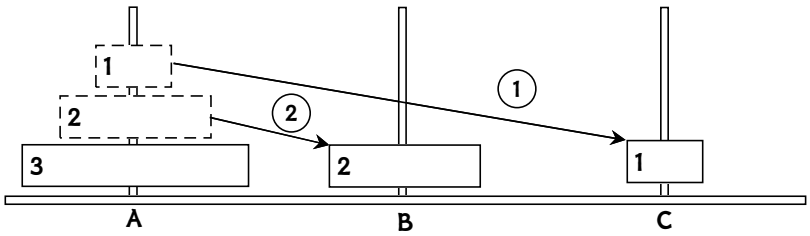
참고로, 64개의 원판을 옮기는 데 $2^n-1 \fallingdotseq 18446744073709551615$번을 움직여야 하고,
한번 옮기는 시간을 1초로 하면, 64개의 원판을 옮기는 데 5849억 4241만 7355년 걸린다.

하노이탑 재귀함수의 시간복잡도는 $O(2^n)$이다. 즉, **지수시간**이다.

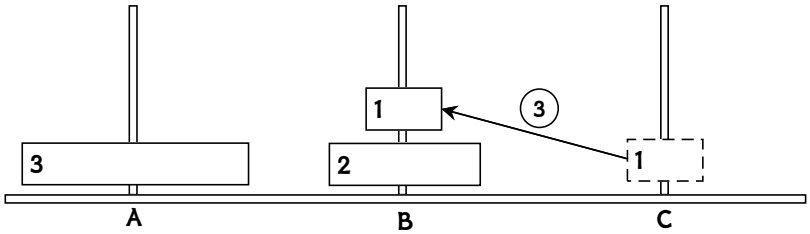
◈ 하노이 탑에서 원판이 3개인 경우 - 7번 이동이 필요

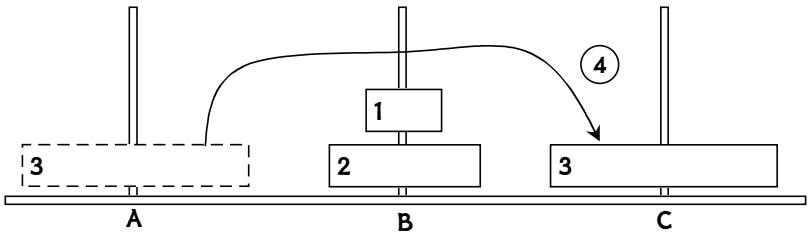


초기 상태

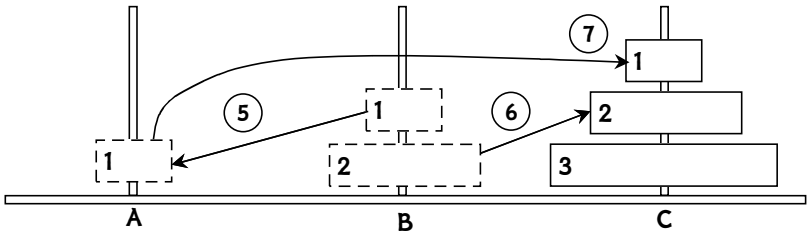


→ 원판 1을 기둥 B로, 원판 2를 기둥 C로 이동해도 된다.



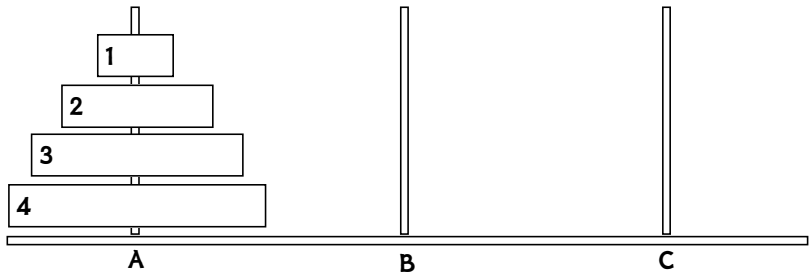


가장 큰 원판 이동

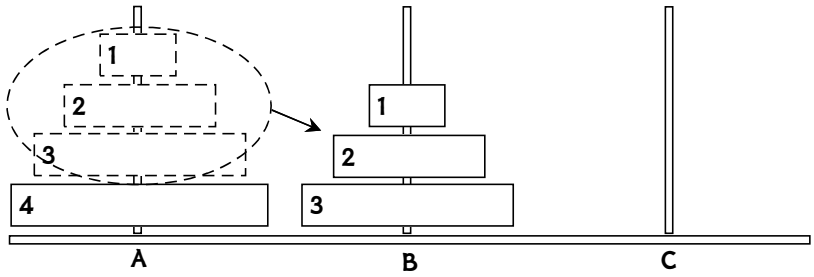


이동 완료

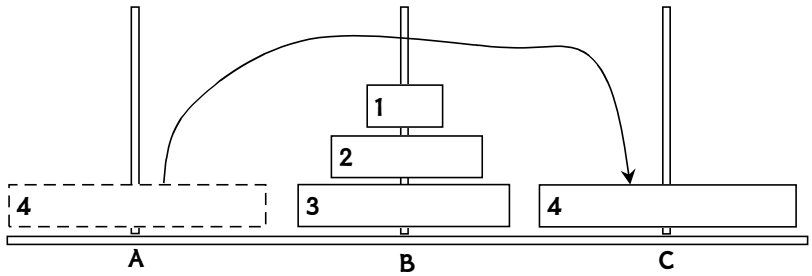
◈ 하노이 탑에서 원판이 4개인 경우 – 15번 이동이 필요



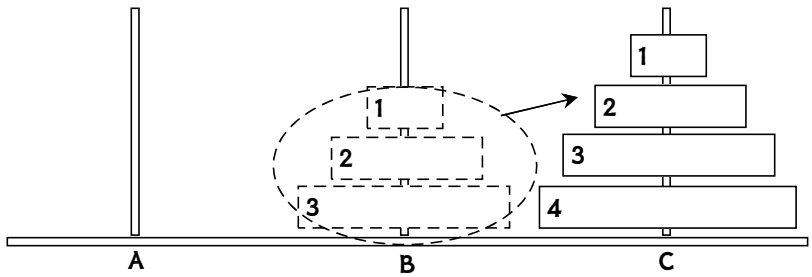
초기 상태



7번 이동이 필요



가장 큰 원판 이동



7번 이동이 필요

• 가장 큰 원판이 이동되는 시점이 전체 이동횟수에서 항상 중간이 된다.(8번째 이동)
• 이유는 가장 큰 원판 이동을 제외한 3개의 원판 이동횟수가 같으므로
• 이런 원리를 이해하면, 주어진 문제에서 답을 쉽게 찾을 수 있다.(같은 패턴 반복)

```
// 하노이탑 알고리즘(원판이 3개인 경우)
void hanoi(int n, char A, char B, char C)
{
    if(n==1)
      printf("원판 %d을 기둥 %c에서 기둥 %c로 옮긴다.\n", n, A, C);
    else
    {
      hanoi(n-1, A, C, B);    //n-1개의 원판을 A에서 B로 이동
      printf("원판 %d을 기둥 %c에서 기둥 %c로 옮긴다.\n", n, A, C); //위치 결정
      hanoi(n-1, B, A, C);    //n-1개의 원판을 B에서 C로 이동
    }
}
void main()
{
    hanoi(3, 'A', 'B', 'C');        //3개의 원판을 기둥 'A'에서 기둥 'C'로 이동
}
```

원판 1을 기둥 A에서 기둥 C로 옮긴다.
원판 2을 기둥 A에서 기둥 B로 옮긴다.
원판 1을 기둥 C에서 기둥 B로 옮긴다.
원판 3을 기둥 A에서 기둥 C로 옮긴다.
원판 1을 기둥 B에서 기둥 A로 옮긴다.
원판 2을 기둥 B에서 기둥 C로 옮긴다.
원판 1을 기둥 A에서 기둥 C로 옮긴다.

```
// 하노이탑 알고리즘에서 핵심 부분
 hanoi(n-1, A, C, B);       //n-1개의 원판을 A에서 B로 이동
 printf("원판 %d을 기둥 %c에서 기둥 %c로 옮긴다.\n", n, A, C); //위치 결정
 hanoi(n-1, B, A, C);       //n-1개의 원판을 B에서 C로 이동
            ↓
 hanoi(n-1, A, C, B);       //A, C, B 순서인데
 printf("원판 %d을 기둥 %c에서 기둥 %c로 옮긴다.\n", n, A, C);
 hanoi(n-1, B, A, C);       //B, A, C 순서임
```

```
// 하노이탑 알고리즘을 파이썬으로 구현
def hanoi(n, A, B, C):
    if n==1:                    # 원반이 1개이면 그냥 목적지로 옮기면 됨
        print('원반 {0} : 기둥 {1}에서 {2}로 이동'.format(n, A, C))
    else:
        hanoi(n-1, A, C, B)     # n-1개의 원반을 A에서 B로 이동
        print('원반 {0} : 기둥 {1}에서 {2}로 이동'.format(n, A, C))    # 가장 큰 원반 이동
        hanoi(n-1, B, A, C)     # n-1개의 원반을 B에서 C로 이동
hanoi(3, 'A', 'B', 'C')         # 기둥 "A"에 있는 원반 3개를 기둥 "C"로 이동
```

↓

↓출력

↓

원반 1 : 기둥 A에서 C로 이동
원반 2 : 기둥 A에서 B로 이동
원반 1 : 기둥 C에서 B로 이동
원반 3 : 기둥 A에서 C로 이동
원반 1 : 기둥 B에서 A로 이동
원반 2 : 기둥 B에서 C로 이동
원반 1 : 기둥 A에서 C로 이동

기출문제 분석

1. 다음 중 컴퓨터 프로그래밍을 통하여 문제를 해결하는데 최적의 시간복잡도가 가장 큰 것은? [2012년 국가 7급]

① 임의의 정수 100,000개를 선택정렬(selection sort)로 정렬하는 문제
② 정렬된 100,000개의 정수에서 임의의 정수를 이진탐색하는 문제
③ 반복호출로 피보나치수열 F0, F1, F2, … 에서 F100000 값을 구하는 문제
④ 재귀호출로 하노이 탑(tower of Hanoi)의 100,000개 원판 이동 문제

♤ **최적의 시간복잡도**

① 정수 100,000개를 선택정렬(selection sort) : $O(n^2) = O(100000^2)$
② 100,000개의 정수에서 임의 정수를 이진탐색 : $O(\log_2 n) = O(\log_2 100000)$
③ 반복호출로 피보나치수열의 100,000번째 수 구하기 : $O(n) = O(100000)$
④ 재귀호출로 하노이 탑의 100,000개 원판 이동 : $O(2^n) = O(2^{100000})$

정답 : ④

◈ **하노이탑 시간복잡도**

- 재귀함수 h()가 호출되는 총 횟수 T(n)이라고 하면
- T(1) = 1
  T(n) = 2×T(n-1) + 1
  → h(k-1)이 두 번 재귀호출 되고 : 2×T(n-1)
  → h(n)이 처음에 한번 호출되므로 : 1
- T(2) = 2×T(1) + 1 = 2×1 + 1 = 3 = 2^2 - 1
  T(3) = 2×T(2) + 1 = 2×3 + 1 = 7 = 2^3 - 1
  T(4) = 2×T(3) + 1 = 2×7 + 1 = 15 = 2^4 - 1
  T(5) = 2×T(4) + 1 = 2×15 + 1 = 31 = 2^5 - 1
  T(6) = 2×T(5) + 1 = 2×31 + 1 = 63 = 2^6 - 1
  :
  T(n) = 2×T(n-1) + 1 = 2^n - 1

- 시간복잡도는 O(2^n) → **지수시간**

2. 다음 "하노이타워" 프로그램을 수행한 결과에서 8번째 줄에 출력되는 문장으로 옳은 것은? [2013년 국가 7급]

```
#include <stdio.h>
void hanoi_tower(int n, char from, char tmp, char to)
{
  if( n==1 ) printf("원판1을 %c에서 %c로 옮긴다. \n", from, to);
  else {
    hanoi_tower(n-1, from, to, tmp);
    printf("원판 %d을 %c에서 %c로 옮긴다. \n", n, from, to);
    hanoi_tower(n-1, tmp, from, to);
  }
}
void main(){ hanoi_tower(4, 'A', 'B', 'C'); }
```

① 원판 1을 C에서 B로 옮긴다.
② 원판 4을 A에서 C로 옮긴다.
③ 원판 3을 A에서 B로 옮긴다.
④ 원판 2을 B에서 C로 옮긴다.

♤ **하노이타워(tower of hanoi) - 재귀호출**

• 실행 결과는 다음과 같다. 원판 1을 A에서 B로 옮긴다. 원판 2을 A에서 C로 옮긴다. 원판 1을 B에서 C로 옮긴다. 원판 3을 A에서 B로 옮긴다. 원판 1을 C에서 A로 옮긴다. 원판 2을 C에서 B로 옮긴다. 원판 1을 A에서 B로 옮긴다. **원판 4을 A에서 C로 옮긴다.** 원판 1을 B에서 C로 옮긴다. 원판 2을 B에서 A로 옮긴다. 원판 1을 C에서 A로 옮긴다. 원판 3을 B에서 C로 옮긴다. 원판 1을 A에서 B로 옮긴다. 원판 2을 A에서 C로 옮긴다. 원판 1을 B에서 C로 옮긴다.	◈ **하노이의 탑은 퍼즐의 일종** 3개의 기둥과 이 기둥에 꽂을 수 있는 크기가 서로 다른 다양한 원판들이 있고, 퍼즐 시작 전에는 한 기둥에 원판들이 큰 것이 아래에 있도록 쌓여 있다. ◈ **게임 목적** 한 기둥에 꽂힌 원판들을 순서 그대로 다른 기둥으로 옮겨서 다시 쌓는 것이다. 다음 2가지 조건을 만족해야 한다. ① 한 번에 하나의 원판만 옮길 수 있다. ② 큰 원판이 작은 원판 위에 있으면 안 된다. ◈ **하노이의 탑의 원리(애니메이션)** • 하노이의 탑 문제는 재귀호출을 이용하여 풀 수 있다. • 원판이 n개 일 때, 2^n-1번의 이동으로 원판을 모두 옮길 수 있다 → 원판이 4개이면 15번 이동이 필요하다.

• hanoi_tower(4, 'A', 'B', 'C'); → 원판은 **4**개이고, 'A', 'B', 'C'는 3개의 기둥을 의미한다.

정답 : ②

3. 다음 C 언어 코드를 실행한 후 5번째 및 7번째 줄에 출력되는 문장으로 옳은 것은? [2021년 국가 7급]

```
#include <stdio.h>
void hanoi(int n, char from, char tmp, char to)
{
  if( n==1 )
    printf("disk 1 from %c to %c\n", from, to);
  else {
    hanoi(n-1, from, to, tmp);
    printf("disk %d from %c to %c\n", n, from, to);
    hanoi(n-1, tmp, from, to);
  }
 }
 int main(int argc, char* argv[]) { hanoi(3, 'A', 'B', 'C'); }
```

	5번째 줄	7번째 줄
①	disk 1 from B to A	disk 1 from A to C
②	disk 1 from C to B	disk 2 from B to C
③	disk 2 from A to B	disk 3 from A to C
④	disk 3 from A to C	disk 2 from B to C

♤ **하노이타워(Tower of Hanoi) - 재귀호출**

• hanoi(3, 'A', 'B', 'C'); → 원판은 3개이고, 'A', 'B', 'C'는 3개의 기둥을 의미한다.

// 실행 결과는 다음과 같다.

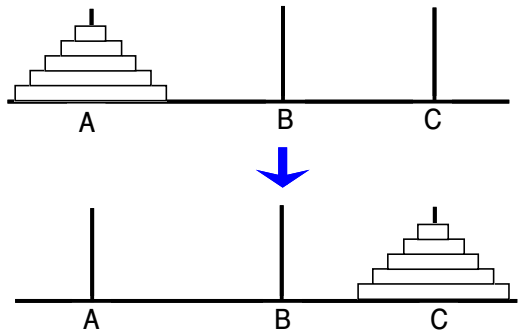
1번째 줄 : disk 1 from A to C → 가장 작은 원판인 disk 1이 기둥 A에서 기둥 C로 이동
2번째 줄 : disk 2 from A to B
3번째 줄 : disk 1 from C to B
4번째 줄 : disk 3 from A to C → **가장 큰 원판**인 disk 3이 **중간**에 이동된다.
5번째 줄 : disk 1 from B to A
6번째 줄 : disk 2 from B to C
7번째 줄 : disk 1 from A to C

• **하노이타워** 문제는 **가장 큰 원판**이 이동되는 시점이 전체 이동횟수에서 **항상 중간**이 된다.

정답 : ①

4. 〈보기 1〉과 같이 세 개의 막대 A, B, C가 주어져 있고, 막대 A에는 N개의 크기가 서로 다른 원판이 큰 것부터 아래에 놓이도록 차례로 쌓여 있다. 한 막대의 맨 위에 있는 원판 하나를 꺼내 다른 막대의 맨 위에 놓을 수 있는데, 이 때 작은 원판 위에 큰 원판을 올려놓을 수 없다. 막대 A에 있는 원판을 모두 막대 C로 옮기는 방법을 〈보기 2〉와 같이 재귀함수로 작성하고자 할 때, 〈보기 2〉의 (가)에 들어갈 내용은? (단, N은 3 이상이다. hanoi(N, start, to, via)는 막대 start의 맨 위에 있는 N개의 원판을 막대 to로 옮기는 함수이고, move(1, start, to)는 start의 맨 위 원판 1개를 to로 옮기는 함수이다) [2022년 서울 7급]

-----<보기 1>----------



-----<보기 2>----------

```
void hanoi(int N, int A, int C, int B) {
  if(N == 1)
    move(1, A, C);
  else {
      (가)
  }
}
```

---

① hanoi(N-1, A, B, C); move(1, A, B); hanoi(N-1, B, C, A);
② hanoi(N-1, A, B, C); move(1, A, C); hanoi(N-1, B, C, A);
③ hanoi(N-1, A, C, B); move(1, A, B); hanoi(N-1, C, A, B);
④ move(1, A, B); hanoi(N-1, A, C, B); move(1, B, C);

♤ 하노이탑

// 먼저, 주어진 문제에 맞도록 알고리즘을 완성하면 다음과 같다.

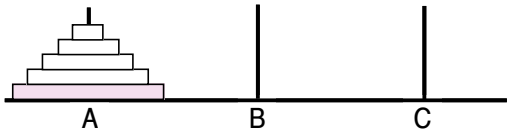
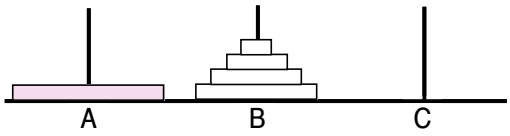
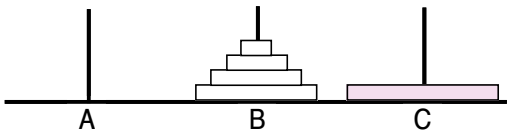
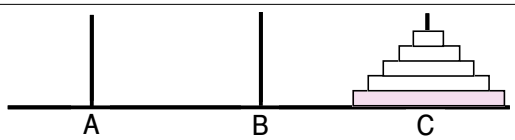
```
void move(int N, int A, int C)      //원판을 실제로 옮기는 함수
{
  printf("원판 %d을 기둥 %d에서 기둥 %d로 옮긴다.\n", N, A, C);
}
```

```
void hanoi(int N, int A, int C, int B) { //N은 원판수, A는 1번 기둥, B는 2번 기둥, C는 3번 기둥
    if(N==1)
       move(N, A, C);          //원판이 1개이면, A(1번기둥)에서 C(3번기둥)로 이동하고 종료
    else {
       hanoi(N-1, A, B, C);    //N-1개의 원판을 A에서 B로 이동
       move(N, A, C);          //문제에서 N을 1로 표현했는데, N이 합리적(가장 큰 원판 이동)
       hanoi(N-1, B, C, A);    //N-1개의 원판을 B에서 C로 이동
    }
}
void main(){ hanoi(3, 1, 3, 2); }       //3개의 원판을 1번 기둥에서 3번 기둥으로 이동
```

↓실행 결과

```
원판 1을 기둥 1에서 기둥 3로 옮긴다.
원판 2을 기둥 1에서 기둥 2로 옮긴다.
원판 1을 기둥 3에서 기둥 2로 옮긴다.
원판 3을 기둥 1에서 기둥 3로 옮긴다.      //항상 중간에 가장 큰 원판 이동
원판 1을 기둥 2에서 기둥 1로 옮긴다.
원판 2을 기둥 2에서 기둥 3로 옮긴다.
원판 1을 기둥 1에서 기둥 3로 옮긴다.
```

// 하노이탑 알고리즘 핵심 정리

초기상태		**목적** : 기둥 A의 원판을 C로 이동 hanoi(N, A, C, B);
단계 1		n-1개의 원판을 기둥 A에서 B로 옮긴 상태 hanoi(N-1, A, B, C);
단계 2		1개 남은 **가장 큰 원판**을 기둥 A에서 C로 옮긴 상태 move(N, A, C); //**원판 위치 확정**
단계 3		기둥 B에 있는 n-1개의 원판을 기둥 C로 옮긴 상태 – **목적 달성** hanoi(N-1, B, C, A);

• 기둥의 위치는 고정이지만, 기둥의 역할은 가변적이다.**(시작, 경유, 도착)**

정답 : ②