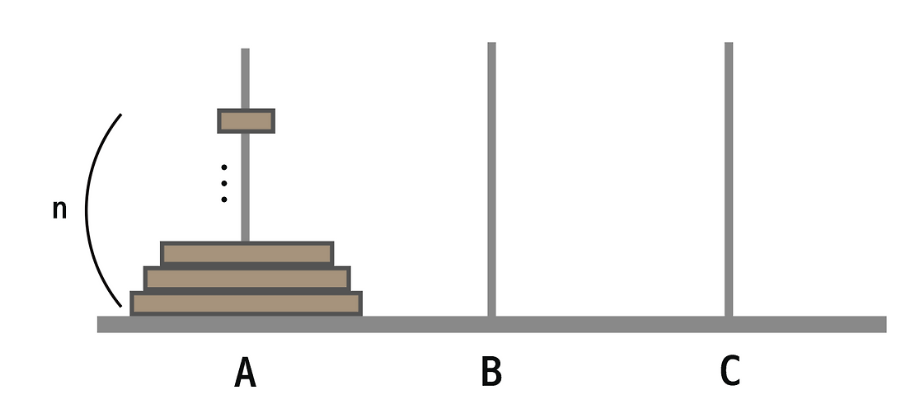
하노이탑 알고리즘 구현 및 분석 보고서 작성

1. 서론

하노이의 탑(Tower of Hanoi)은 유명한 재귀 알고리즘 문제 중 하나로, 세 개의 기둥과 여러 개의 원판을 이용하여 특정한 규칙을 만족하면서 원판을 이동시키는 문제이다. 본 보고서에서는 하노이탑 알고리즘의 개념을 설명하고, 이를 코드로 구현하는 과정을 설명한다.

1. 본론

하노이탑 개수와 상관없이, 일정 규칙을 찾고, 최소 단위를 생각한다면 어렵지 않게 풀 수 있다. 일단 하노이 탑 공식부터 알아보자.



n개의 원판이 있다고 가정하자

1. **가장 큰 원판을 C 로 옮기기 위해서는 n-1 개의 원판이 A 에서 B 로 간다.**
2. **A에 있는 가장 큰 원판이 C로 간다.**
3. **B의 n-1 개의 원판이 C로 간다.**

n = 1일 때

A C

n = 2 일때

A B (A의 n-1를 B에 옮긴다)

A C (가장 큰 원판 C로)

B C ( B의 n-1개의 원판 C로)

N = 3 일때

A C/A B/C A (A의 n-1를 B에 옮긴다)

A C (가장 큰 원판 C로)

B A/B C/A C ( B의 n-1개의 원판 C로)

private static void Hanoi(int N, char start, char mid, char to) {  
 // 이동할 원반의 수가 1개라면?  
 if (N == 1) {  
 *sb*.append(start + " " + to + "\n");  
 return;  
 }  
  
 // STEP 1 : N-1개를 A에서 B로 이동   
 *Hanoi*(N - 1, start, to, mid);  
  
 // STEP 2 : 1개를 A에서 C로 이동   
 *sb*.append(start +" "+ to + "\n");  
  
 // STEP 3 : N-1개를 B에서 C로 이동   
 *Hanoi*(N - 1, mid, start, to);

1. if(N==1)sb.append(start =>to) 1개의 원판을 start -> end로 옮긴다.
2. Hanoi(N-1, start, to, mid) 는 N-1개의 원판은 start -> mid(남는기둥)에 옮긴다.
3. Sb.append(start => to) 는 N번째 원판을 start -> to로 옮긴다.
4. Hanoi(N-1, mid, start, to)는 N-1의 원판을 mid - > to로 욺긴다.

N=1 일때

1개의 원판이 A -> C로 간다.

N=2 일때

2번 원판이 A - > C로 가야한다.

1번 원판이 A -> B 로 간다.

2번 원판이 A -> C 로 간다.

1번 원판 아까 mid로 갔던걸 mid->to로 이동하면 B->C

재귀를 들어갈 때 start -> mid 였던게 나올 때 mid -> to가 된다.

1. 결론