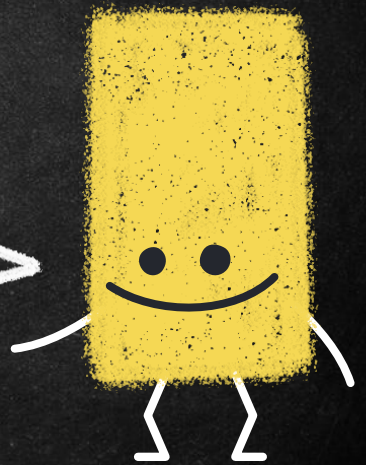
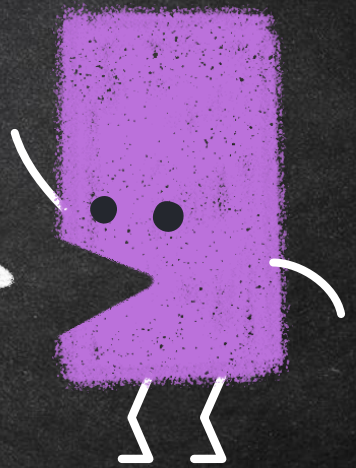
Algorithm Study For 2021

WEEK 4



“

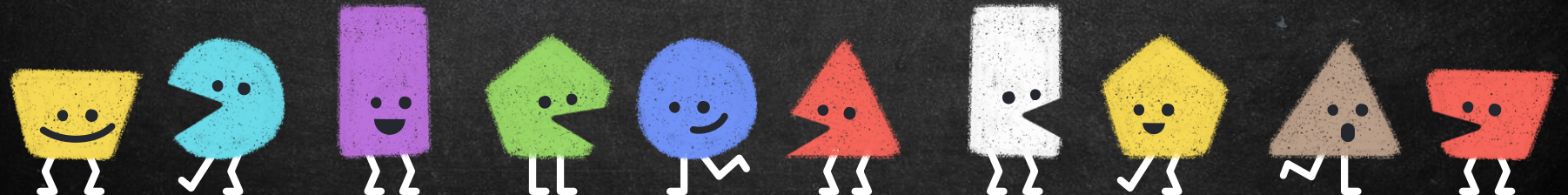
이번 주에는 중요한 내용인
재귀에 대해서 알아봅니다!!



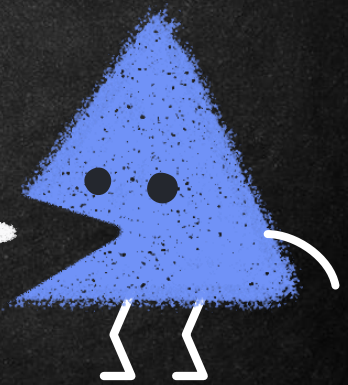
Recursion

재귀가 무엇인지 이해하고, 예시들에 적용해보며

문제까지 스스로 풀어봅니다.



ConTents

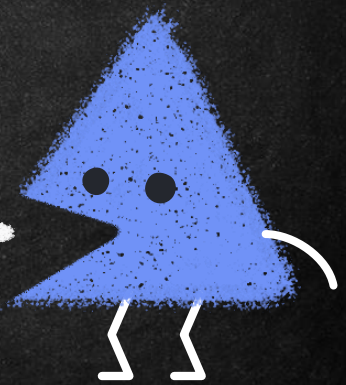


재귀함수란...?

Recursion

→ 자기 자신을 호출하는 함수!

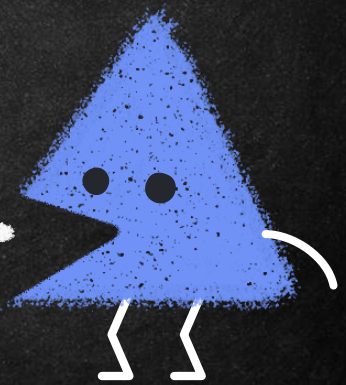
한 단계 낮은 문제가 해결된다면 그것을 바탕으로 답을 얻을 수 있다! (???)



재귀함수란...?

한 단계 낮은 문제가 해결된다면 그것을 바탕으로 답을 얻을 수 있다! (???)

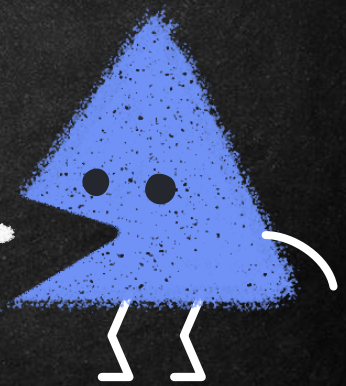
- ➜ 이게 무슨 말일까요...?
- ➜ 예를 들어서 생각해봅시다



팩토리얼 함수 만들기

Factorial(5) = 5*4*3*2*1입니다.
Factorial(4) = 4*3*2*1이므로 여기에 5를 곱해주면
Factorial(5) = Factorial(4)*5라는 등식이 성립합니다.

한단계 낮은(4!)문제의 해결을 통해서 5!을 구할 수 있는 셈이죠...

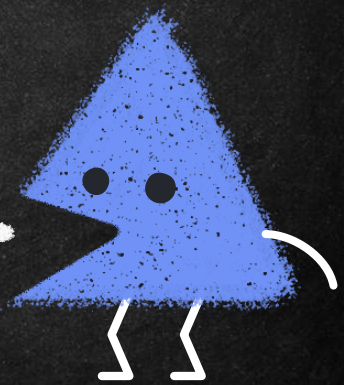


팩토리얼 만들기!

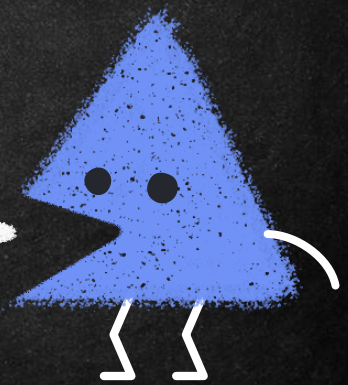
```cpp
#include <iostream>

using namespace std;

int factorial(int n)
{
    return factorial(n - 1) * n;
}

int main(void)
{
    cout << factorial(5) << endl;
    return 0;
}
```

이렇게 하면 될까요..?

자기 자신을
호출한다고 했으니
하나씩 줄여서
호출하고 자기 자신과
곱했습니다!!!



절대 안됩니다!!!!



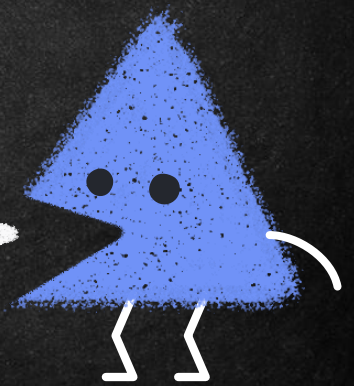
팩토리얼 만들기!

```cpp
#include <iostream>

using namespace std;

int factorial(int n)
{
    if (n == 1)
        return 1;
    return factorial(n - 1) * n;
}

int main(void)
{
    cout << factorial(5) << endl;
    return 0;
}
```

재귀함수는 자기 자신을 호출합니다!

-> 무한 루프에 빠질 수 있습니다.

=> 적어도 하나의 더 이상 자기 자신을 호출하지 않는 조건이 필요합니다!!!

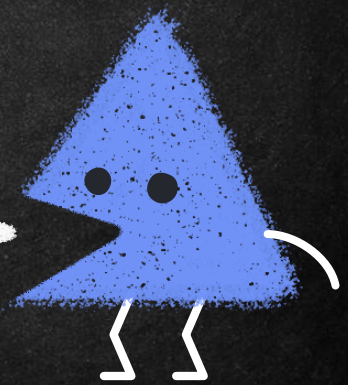


팩토리얼 만들기!

```cpp
#include <iostream>

using namespace std;

int factorial(int n)
{
    int result = 1;
    for (int i = 1; i <= n; i++)
    {
        result *= i;
    }
    return result;
}
int main(void)
{
    cout << factorial(5) << endl;
    return 0;
}
```

왼쪽과 같이 코드를 짤 수도 있습니다.

For문을 활용한 반복함수입니다.



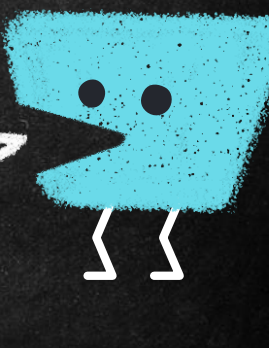
반복 vs 재귀

반복함수!!!

```cpp
#include <iostream>

using namespace std;

int factorial(int n)
{
    int result = 1;
    for (int i = 1; i <= n; i++)
    {
        result *= i;
    }
    return result;
}
int main(void)
{
    cout << factorial(5) << endl;
    return 0;
}
```

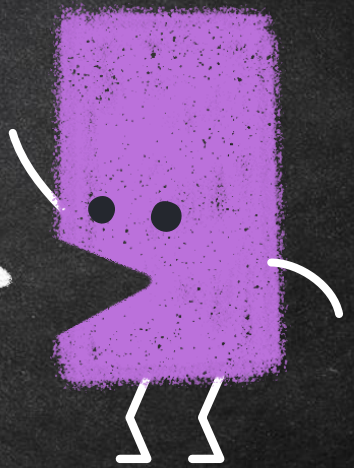
재귀함수!!!

```cpp
#include <iostream>

using namespace std;

int factorial(int n)
{
    if (n == 1)
        return 1;
    return factorial(n - 1) * n;
}

int main(void)
{
    cout << factorial(5) << endl;
    return 0;
}
```



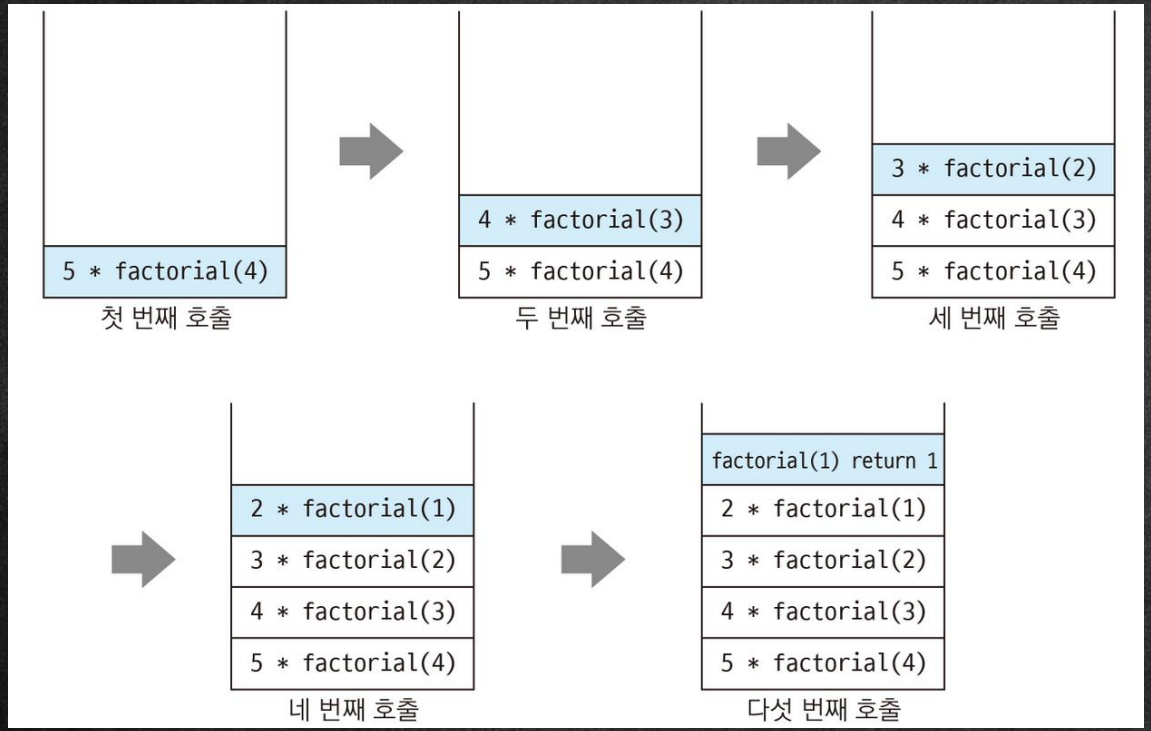
“

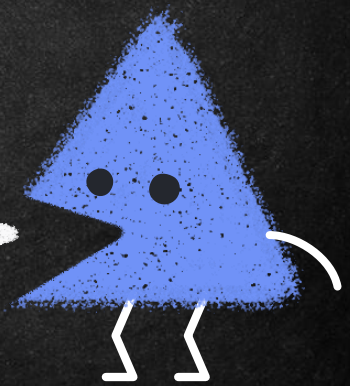
어떻게 다를까요..?!



재귀함수의 호출!

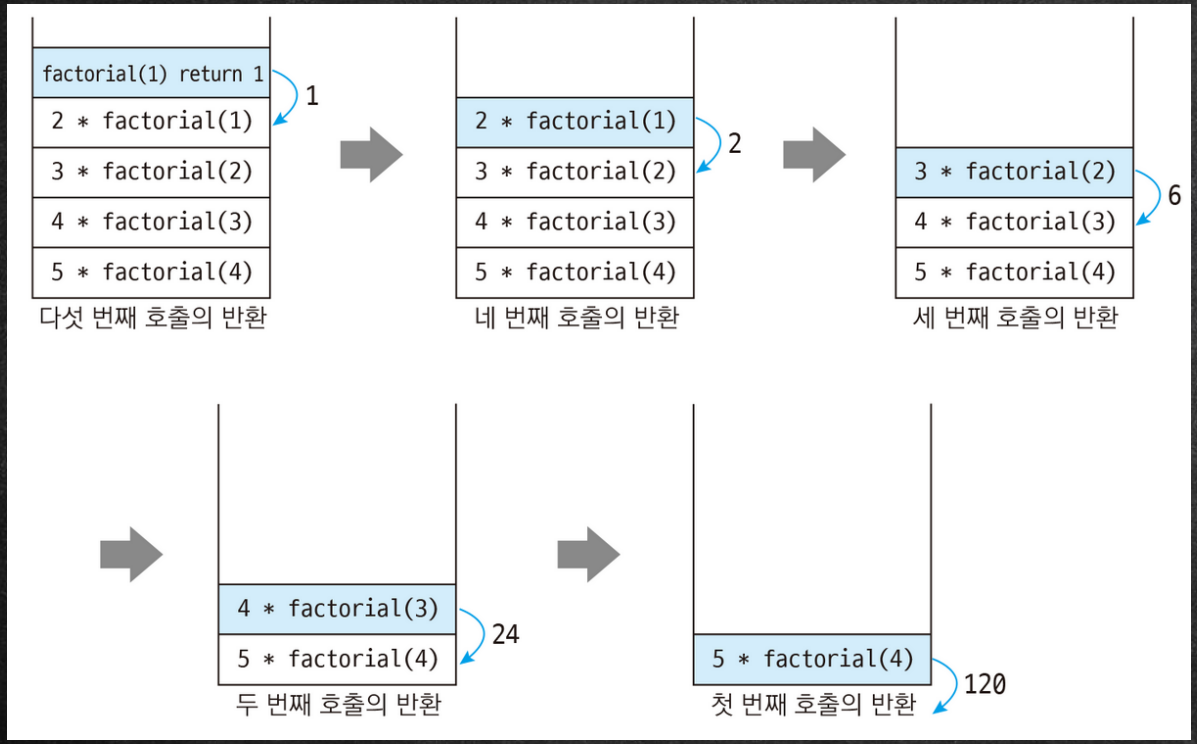
Factorial함수의 재귀 호출은 다음과 같습니다.

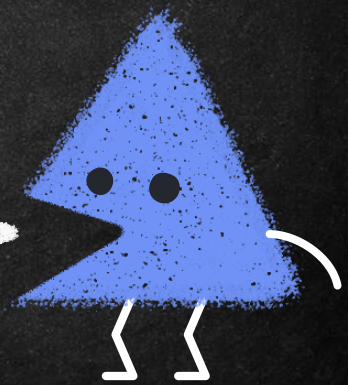




재귀함수의 반환!

Factorial함수의 반환은 다음과 같습니다.

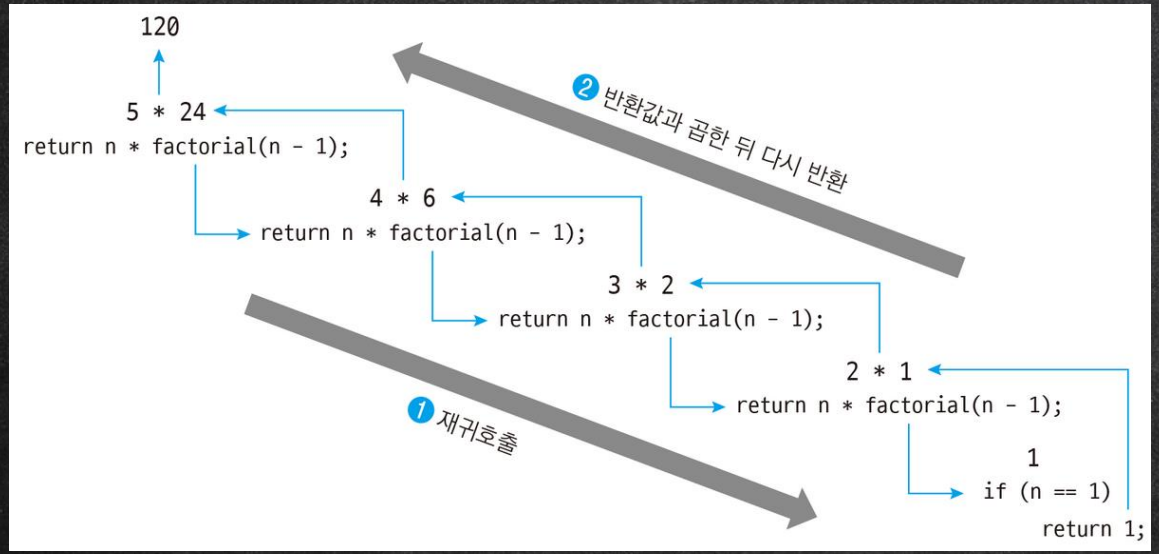


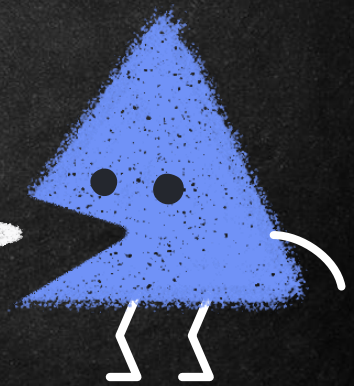


결론!

큰 숫자가 들어오게 되면 호출 스택의 오버플로우가 발생하기도 하며, 오버헤드가 발생합니다.

속도도 느리겠죠!!!

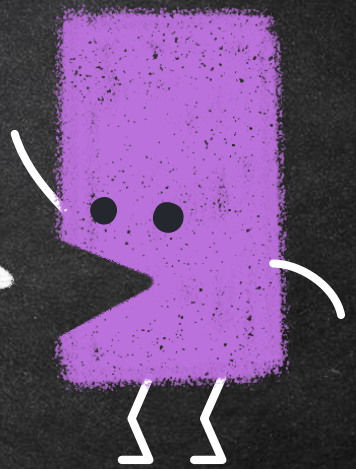




"

함수의 호출원리 및 재귀호출에 대해 깊이 알아봅시다..!

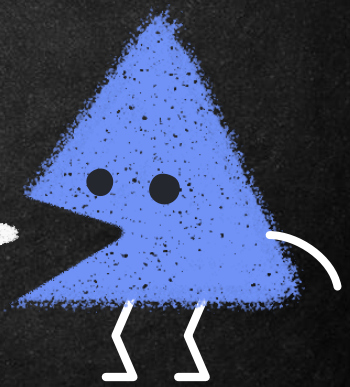
http://10bun.tv/beginner/episode-4/#%E1%84%92%E1%85%A2%E1%86%A8%E1%84%89%E1%85%B5%E1%86%B7-%E1%84%80%E1%85%A1%E1%86%BC%E1%84%8B%E1%85%B4



결론!

반복함수와 재귀함수를 비교해보면 다음과 같습니다.

	재귀 함수	반복문
장점	- 상대적으로 간결한 코드	- 속도가 상대적으로 빠름
단점	- 메모리를 많이 사용함 - 속도가 상대적으로 느림	- 상대적으로 복잡한 코드

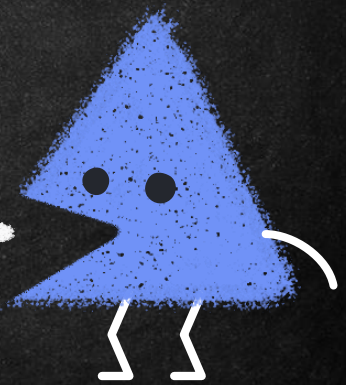


활용해봅시다!!!

피보나치 수열이 뭔가요?

$$F_0 = 0,\ F_1 = 1,\ F_{n+2} = F_{n+1} + F_n$$

사용자로부터 정수를 입력받고 거기까지의 수열을 보여주는 코드를 구현해봅시다!!!(Feat. 재귀)

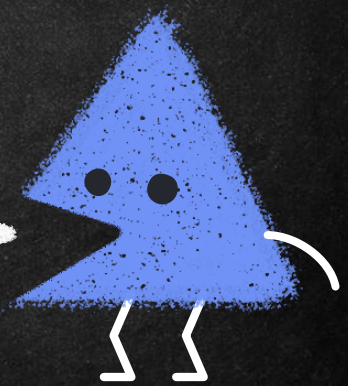


피보나치 수열~

```cpp
#include <iostream>
using namespace std;
int fibonacci(int n)
{
    if (n < 2)
        return n;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);
}
int main(void)
{
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cout << fibonacci(i);
    }
    return 0;
}
```

```
5
11235
```

재귀함수가 어떻게
사용되었는지
이해해보세요!

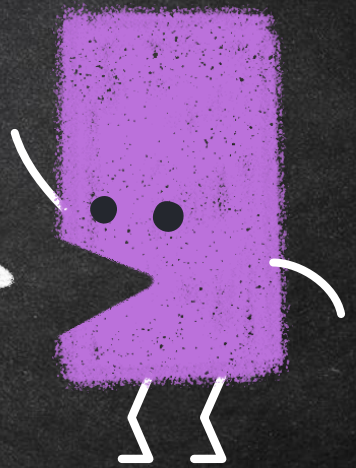


"

지금까지는 너무 쉬운 예시였습니다. 이제 좀 머리를 써볼게요.

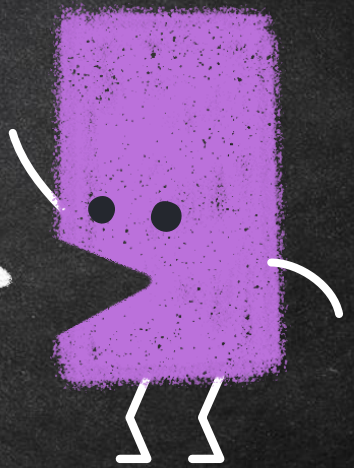
유명한 하노이의 탑 문제랑

트리순회 문제를 해결해봅시다!



“

하노이의 탑 문제가 뭔가요..!?



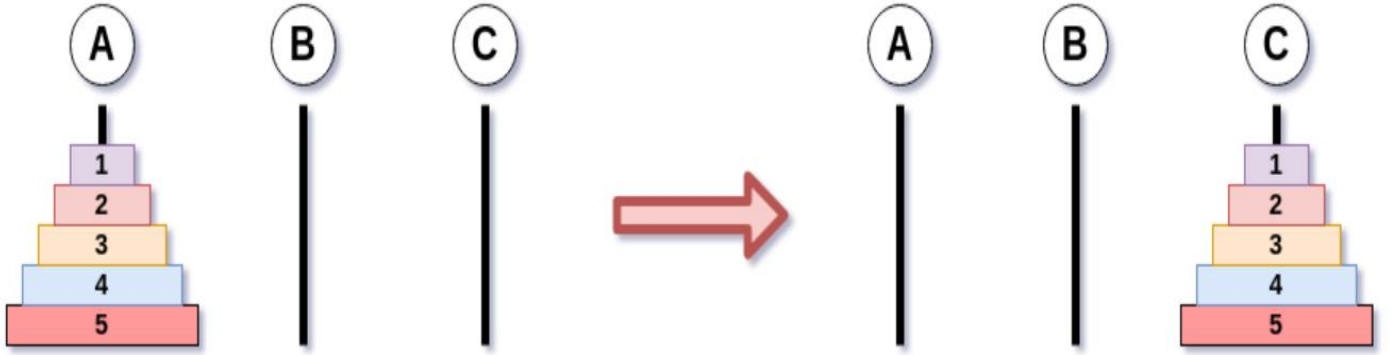
하노이의 탑!!!

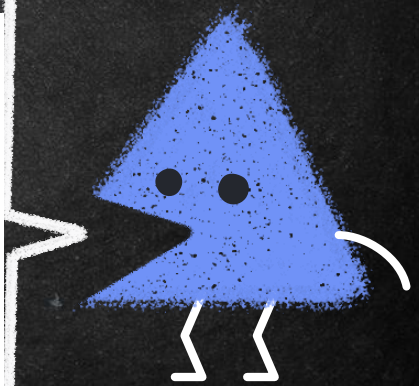
목표: 아래 그림과 같이 A의 원반을 C로 옮기기!!!

<제약조건>

1) 한 번에 움직일 수 있는 원반은 원반 하나 뿐!

2) 어떤 원반 위에 그보다 더 큰 원반 쌓지 못함!



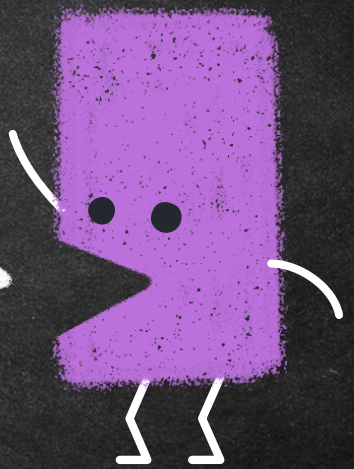


“

최소 몇번을 옮겨야할까요.!?

스스로 생각해보고 다음

슬라이드로 넘어가주세요~



하노이의 탑!!!

쉽지 않은 문제입니다...

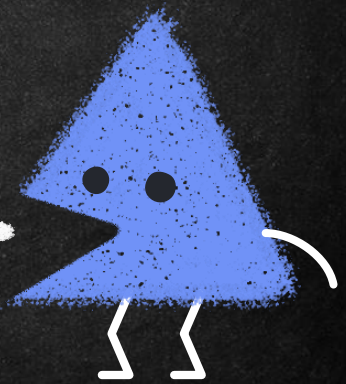
사실 5개라서 매우 러프하죠...

이렇게 상황이 복잡할 때에는,,,

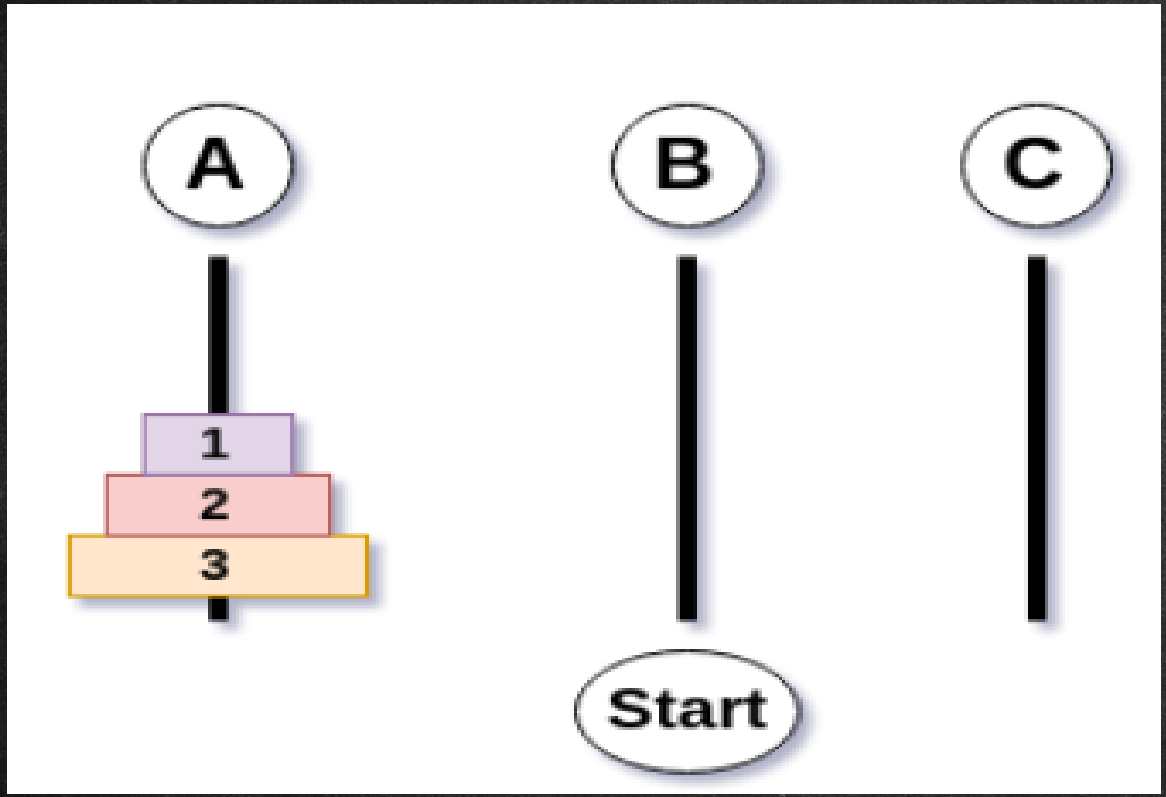
1. 직접 해보면서 상황을 이해하기!

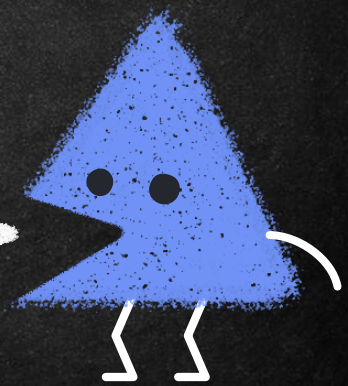
2. 복잡한 상황을 작게 만들어 해결하고 확대하기!

그런 의미에서 5개 -> 3개로 생각을 해보겠습니다~

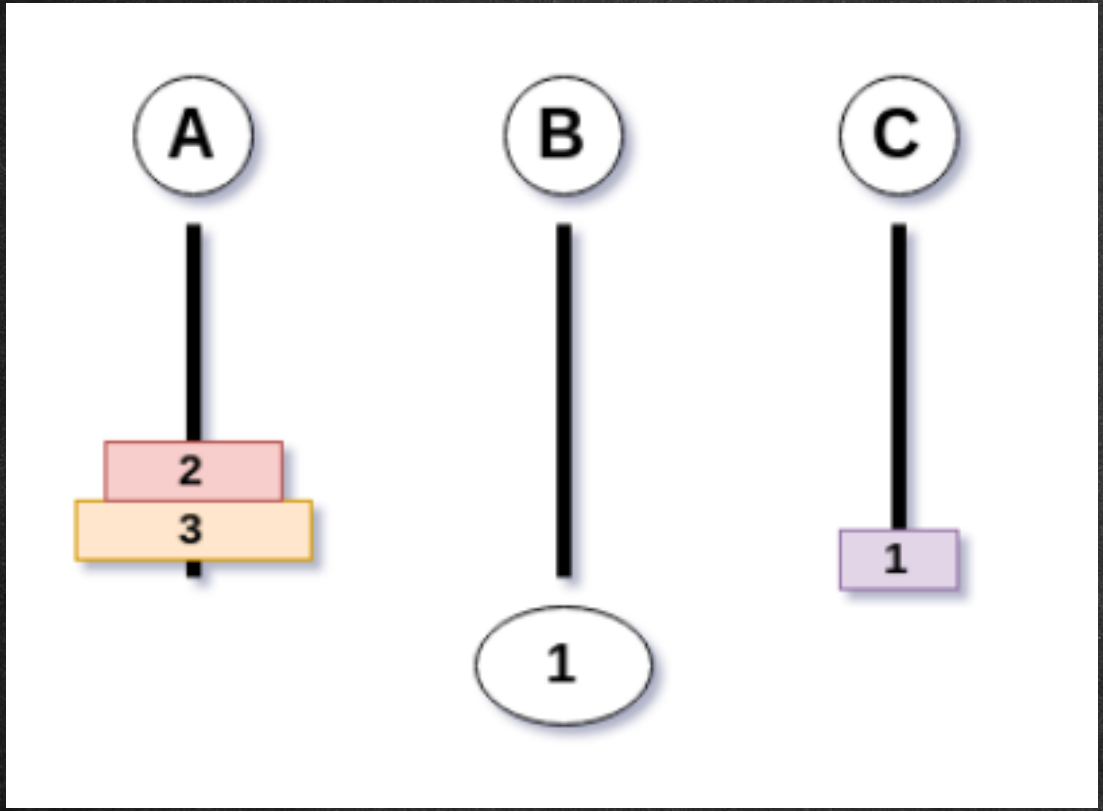


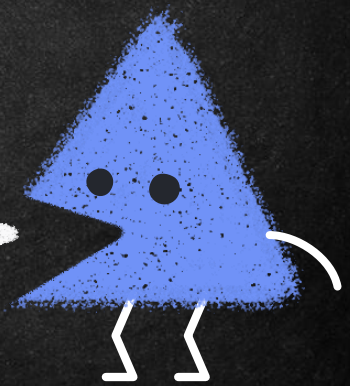
하노이의 탑 해보기!!!



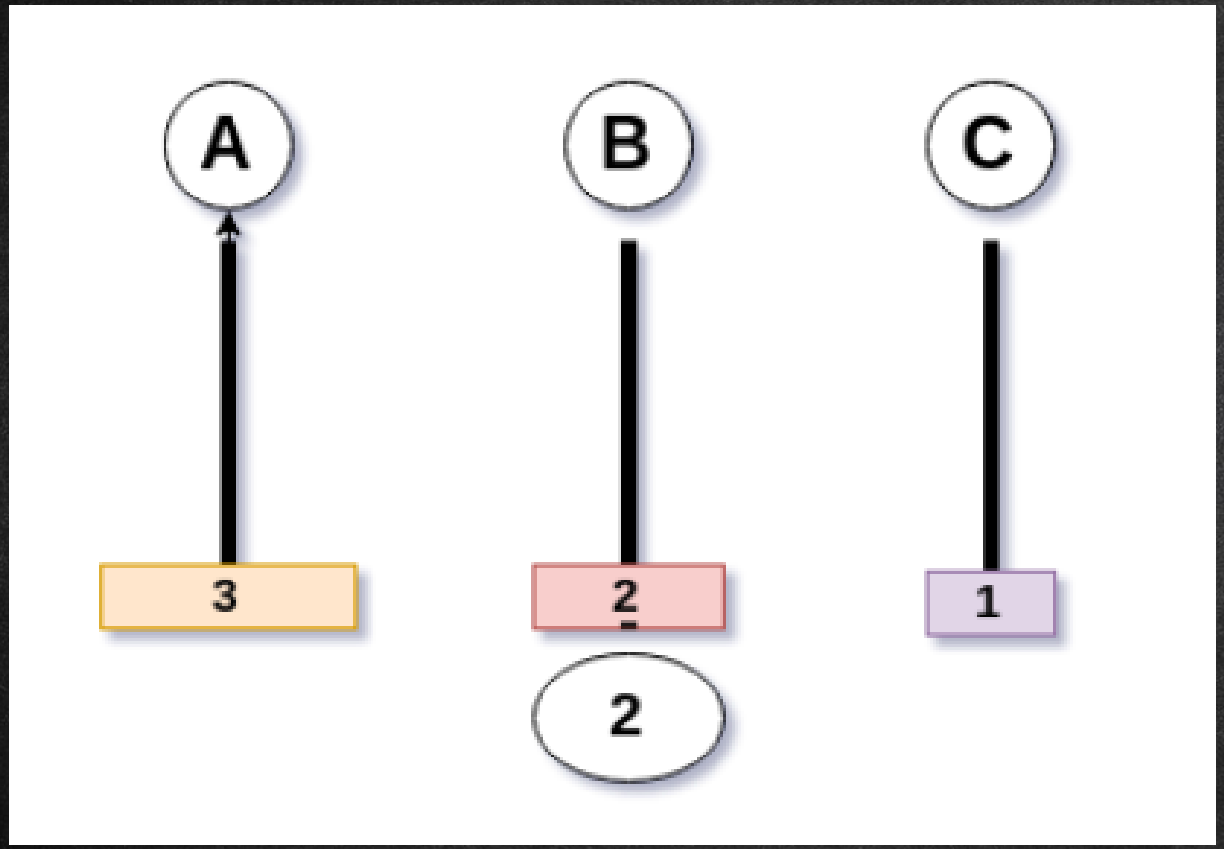


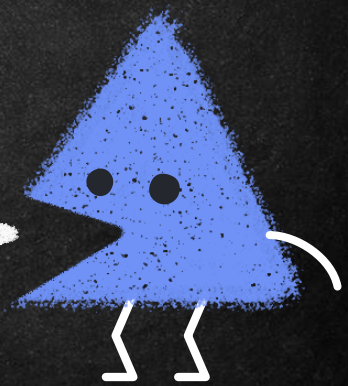
하노이의 탑 해보기!!!



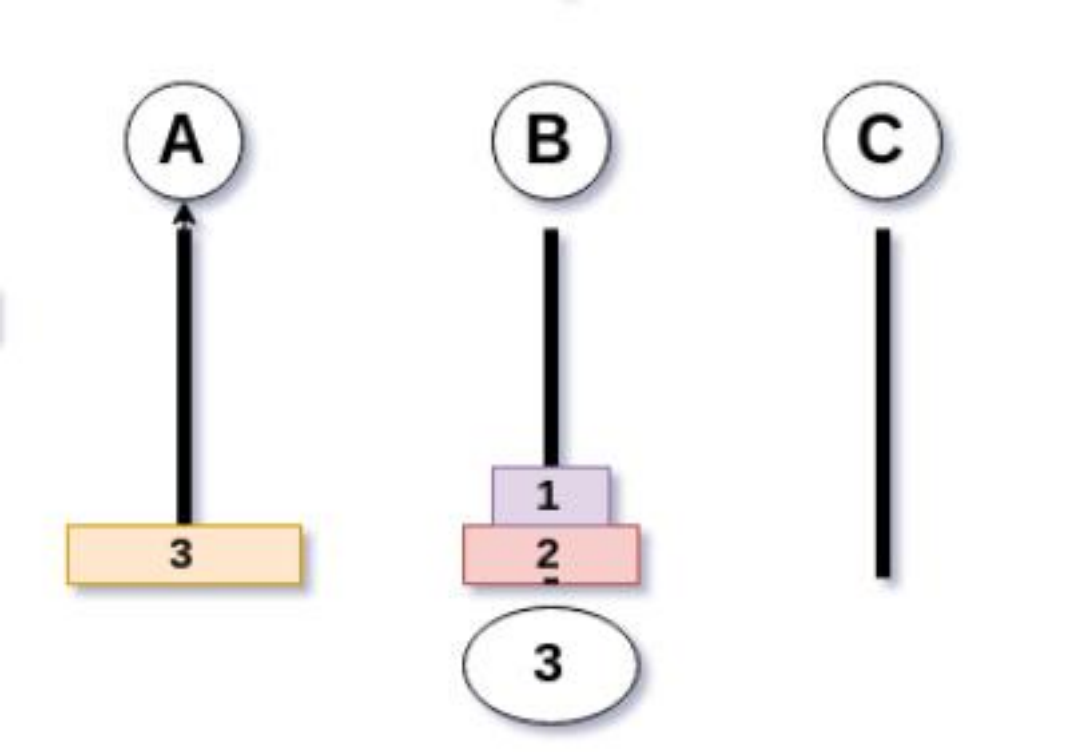


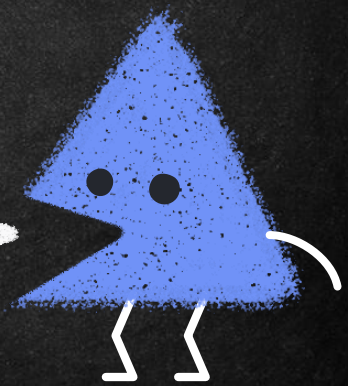
하노이의 탑 해보기!!!



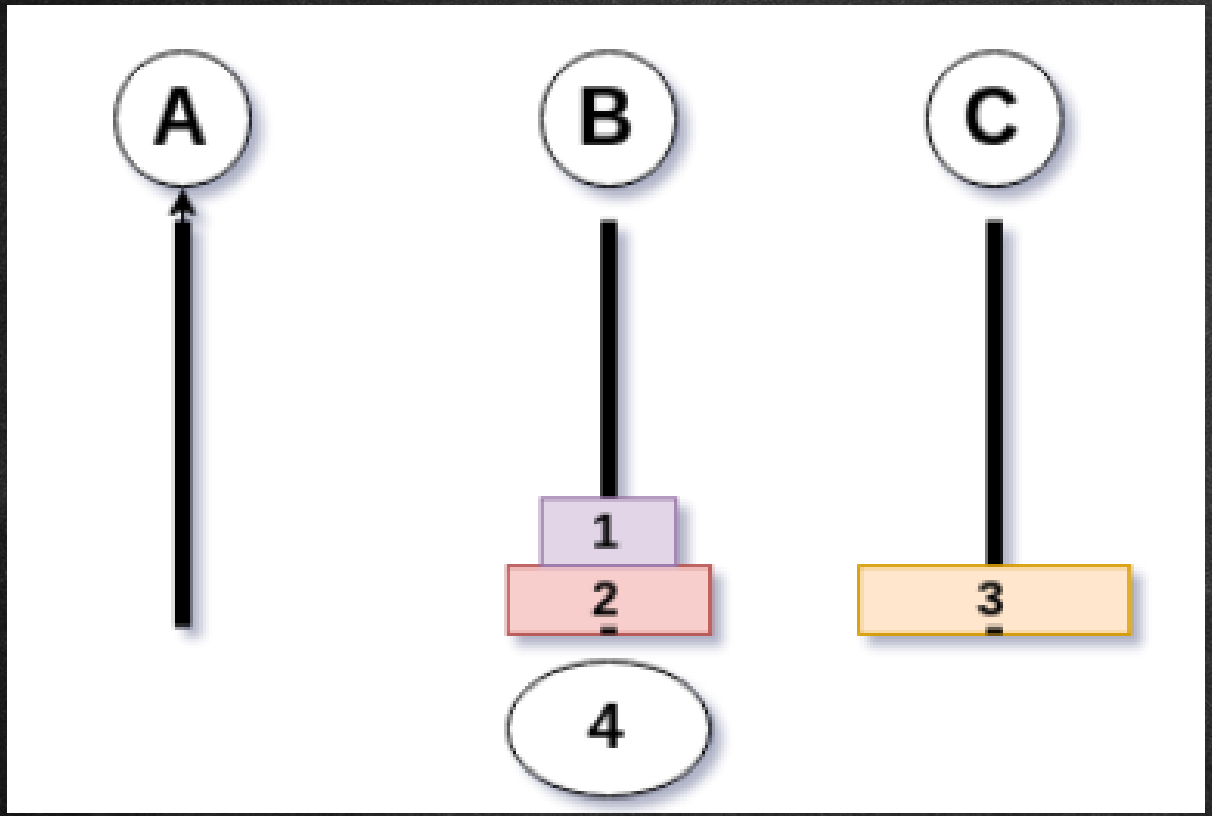


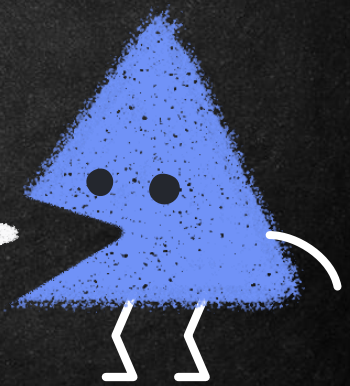
하노이의 탑 해보기!!!



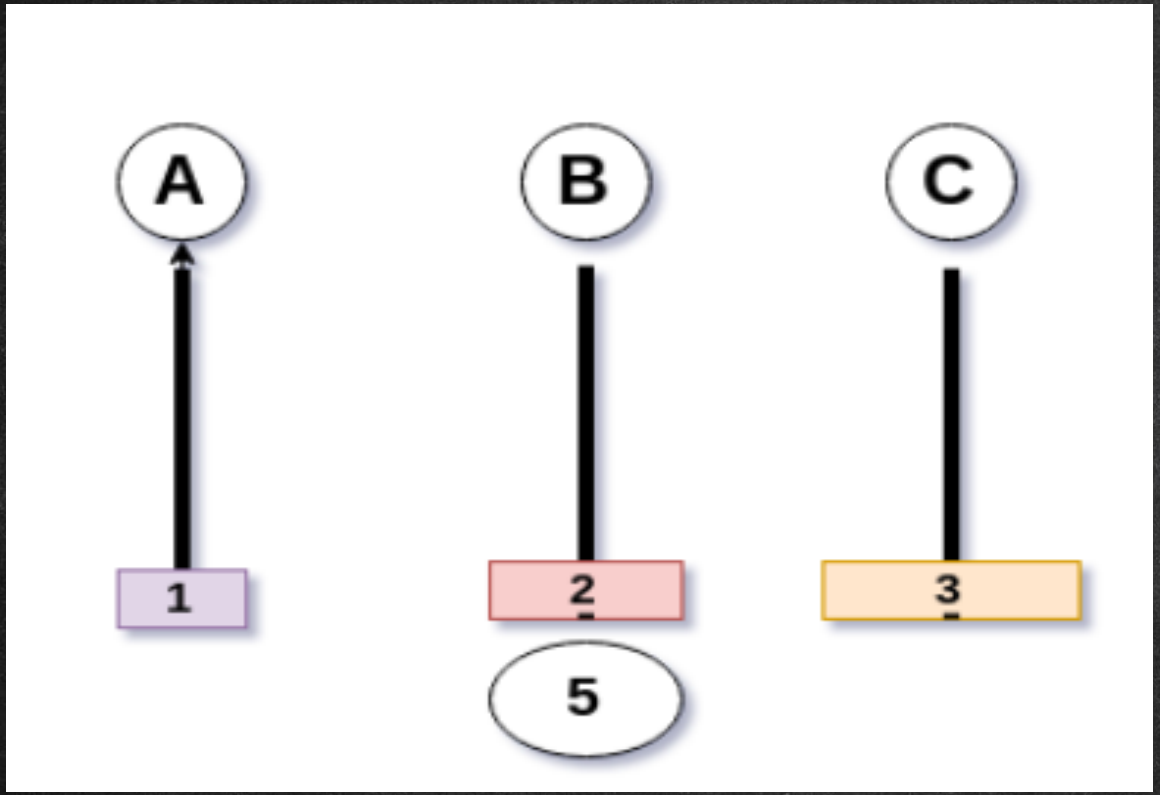


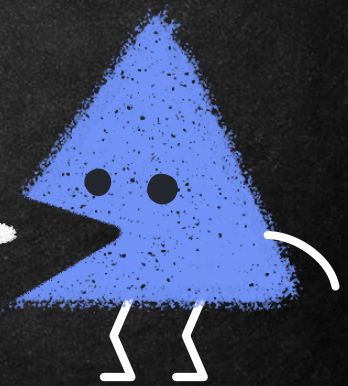
하노이의 탑 해보기!!!



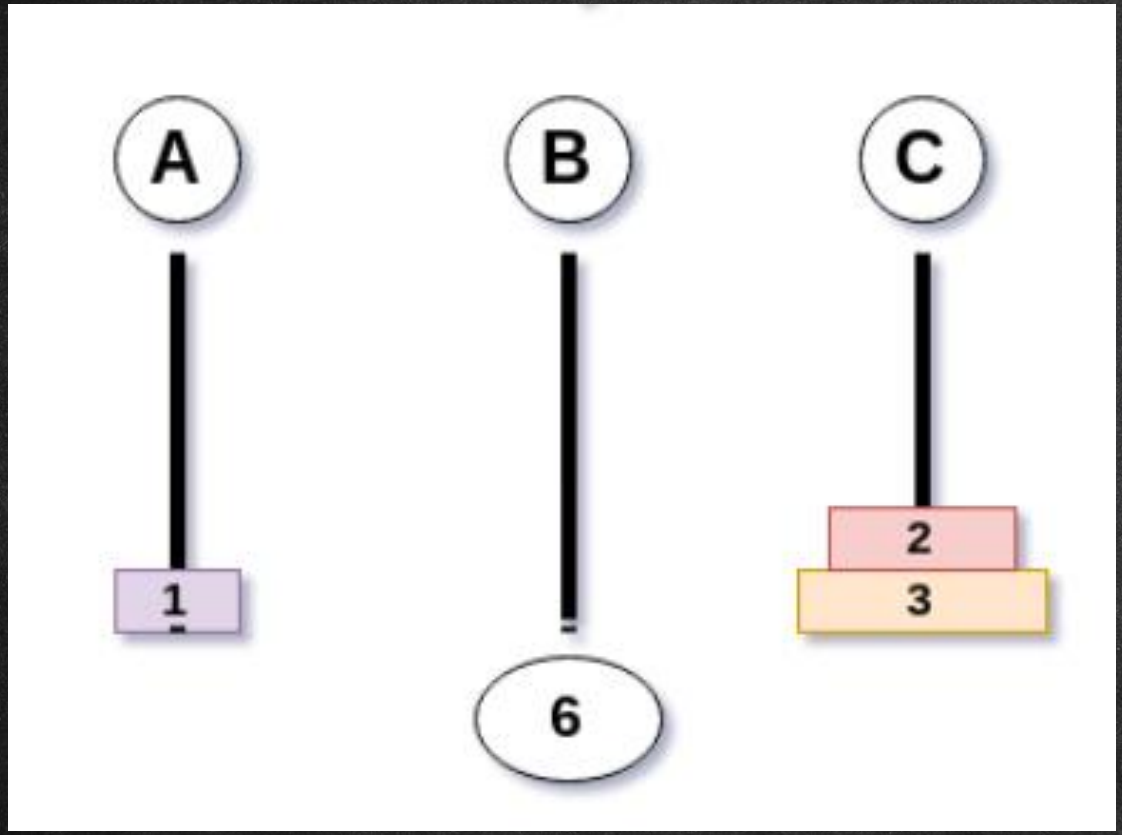


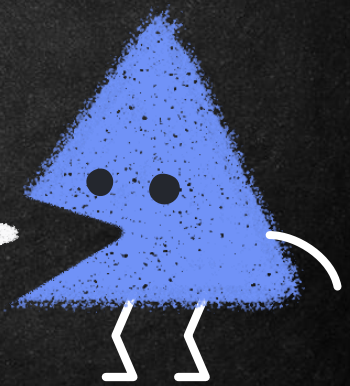
하노이의 탑 해보기!!!



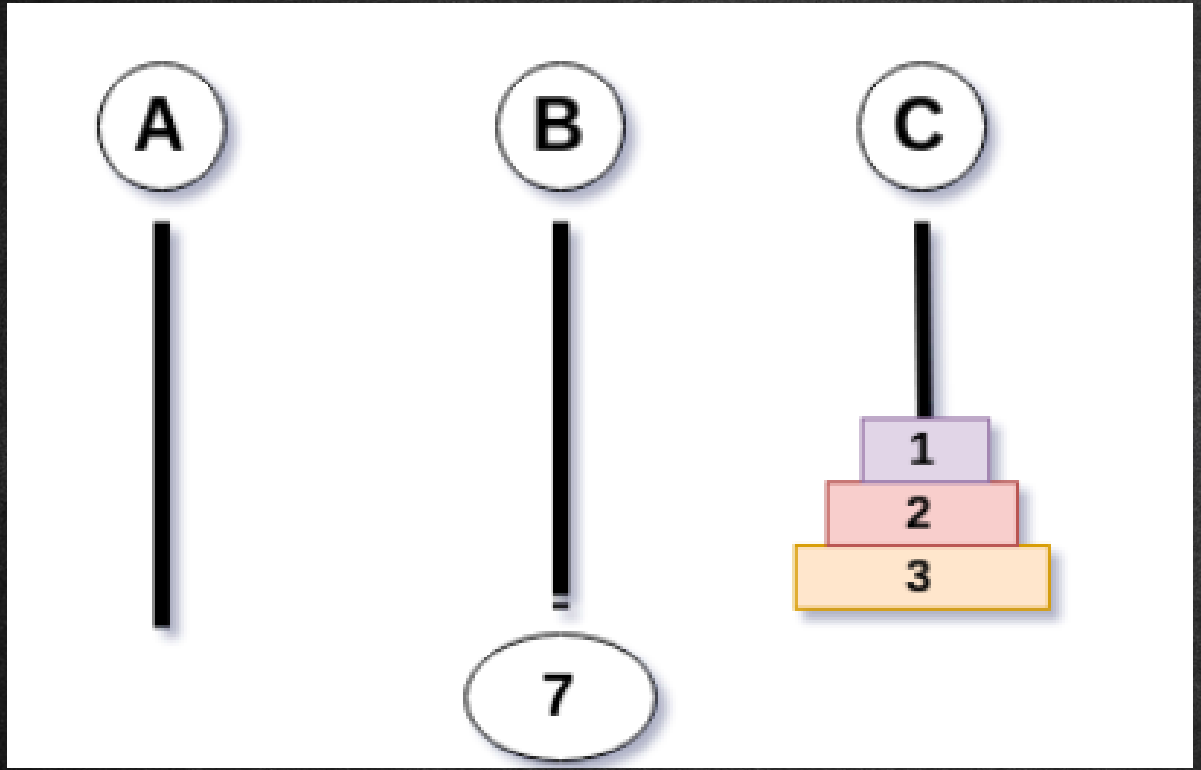


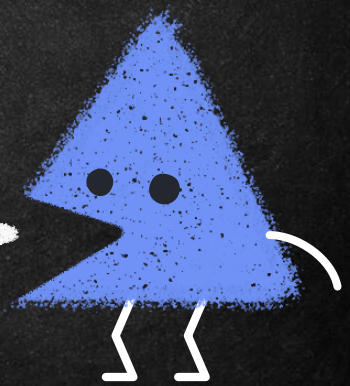
하노이의 탑 해보기!!!





하노이의 탑 해보기!!!





하노이의 탑!!!

<아이디어>

1) 1-2 원반을 A에서 B로 옮겨놓습니다.

-1번 원반을 A에서 C로 옮겨놓습니다.

-2번 원반을 A에서 B로 옮겨놓습니다.

-1번 원반을 C에서 B로 옮겨놓습니다.

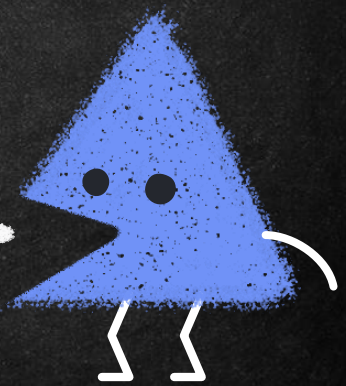
1) 3원반을 C로 옮겨놓습니다.

2) 1-2 원반을 B에서 C로 옮깁니다.

-1번 원반을 B에서 A로 옮겨놓습니다.

-2번 원반을 B에서 C로 옮깁니다.

-1번 원반을 A에서 C로 옮깁니다.



하노이의 탑!!!

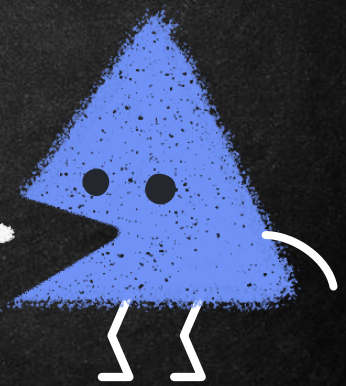
<일반화>

1) 1~n-1개의 원반을 A에서 B로 옮겨놓습니다.

2) n원반을 C로 옮겨놓습니다.

3) 1~n-1 원반을 B에서 C로 옮깁니다.

=> 재귀호출이 2번 들어가게 됩니다. 1) 3)이 해당함!



문제!!!

문제는 다음과 같습니다.

Hanoi(n,start,to,via): start에서 to로 via를 거쳐 총 n개의 원반을 운반할 때 각 이동 과정을 출력해주세요~

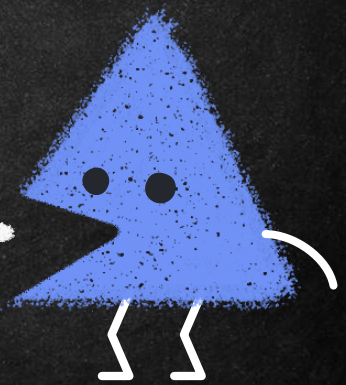
<힌트>

예시) 하노이(3,a,c,b) => 앞의 그림을 통해서 확인하세요!

1) 하노이(2,a,b,c)

2) 이후 3번 원반을 c로 옮긴다.

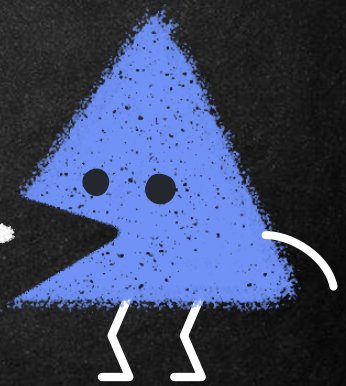
3) 하노이(2,b,c,a)



CODE

```
원반 1을 A에서 C로 이동!
원반 2을 A에서 B로 이동!
원반 1을 C에서 B로 이동!
원반 3을 A에서 C로 이동!
원반 1을 B에서 A로 이동!
원반 2을 B에서 C로 이동!
원반 1을 A에서 C로 이동!
```

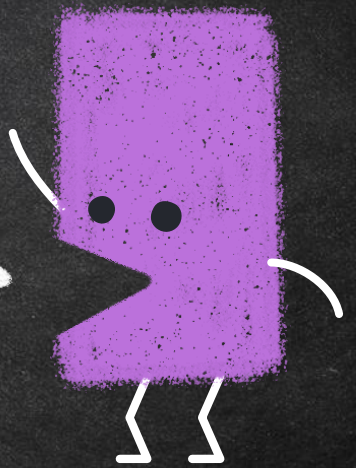
```cpp
#include <iostream>
using namespace std;

void HanoiTower(int n, char from, char via, char to)
{
    if (n == 1)
    {
        cout << "원반 " << n << "을 " << from << "에서 " << to << "로 이동!" << endl;
        return;
    }
    HanoiTower(n - 1, from, to, via);
    cout << "원반 " << n << "을 " << from << "에서 " << to << "로 이동!" << endl;
    HanoiTower(n - 1, via, from, to);
}
int main(void)
{
    HanoiTower(3, 'A', 'B', 'C');
    return 0;
}

```



"

지난 주에 학습한 트리 순회
관련 문제를 하나 풀어볼게요!!

재귀를 학습했으니 더 잘
이해가 됩니다~



문제!!! 백준-1991번 트리의 순회

입력

첫째 줄에는 이진 트리의 노드의 개수 N(1≤N≤26)이 주어진다. 둘째 줄부터 N개의 줄에 걸쳐 각 노드와 그의 왼쪽 자식 노드, 오른쪽 자식 노드가 주어진다. 노드의 이름은 A부터 차례대로 영문자 대문자로 매겨지며, 항상 A가 루트 노드가 된다. 자식 노드가 없는 경우에는 .으로 표현된다.

출력

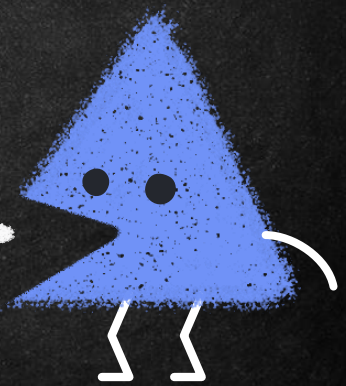
첫째 줄에 전위 순회, 둘째 줄에 중위 순회, 셋째 줄에 후위 순회한 결과를 출력한다. 각 줄에 N개의 알파벳을 공백 없이 출력하면 된다.

예제 입력 1 복사

```
7
A B C
B D .
C E F
E . .
F . G
D . .
G . .
```

예제 출력 1 복사

```
ABDCEFG
DBAECFG
DBEGFCA
```



트리의 순회~

```cpp
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

struct Tree {
    char c;
    Tree* left = NULL;
    Tree* right = NULL;
};
```

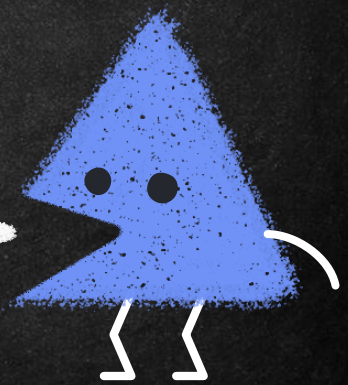
```cpp
vector<Tree> node(26);
Tree* root;

for (int i = 0; i < n; i++) {
    char parent, left, right;

    cin >> parent >> left >> right;
    node[parent - 'A'].c = parent;

    if (left != '.') {
        node[parent - 'A'].left = &node[left - 'A'];
    }
    if (right != '.') {
        node[parent - 'A'].right = &node[right - 'A'];
    }
}

root = &node[0];
```

일단 트리를 만들어볼게요!!!

구조체와 벡터를
활용해서
구현해보겠습니다~

https://xtar.tistory.com/39

트리 구현이 익숙하지
않으시면 읽어보세요!



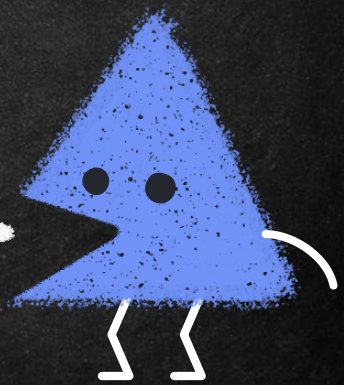
트리의 순회~

재귀를 공부했으니 곰곰이 생각해봅시다!!!

```
//전위 순회
void preorder(Tree* node) {
    cout << node->c;

    if (node->left != NULL) {
        preorder(node->left);
    }

    if (node->right != NULL) {
        preorder(node->right);
    }
}
```

```
//중위 순회
void inorder(Tree* node) {
    if (node->left != NULL) {
        inorder(node->left);
    }

    cout << node->c;

    if (node->right != NULL) {
        inorder(node->right);
    }
}
```

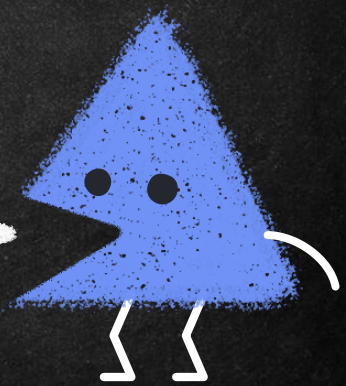
```
//후위 순회
void postorder(Tree* node) {
    if (node->left != NULL) {
        postorder(node->left);
    }

    if (node->right != NULL) {
        postorder(node->right);
    }

    cout << node->c;
}
```



최종 출력!!!

```
preorder(root);
cout << endl;
inorder(root);
cout << endl;
postorder(root);
```

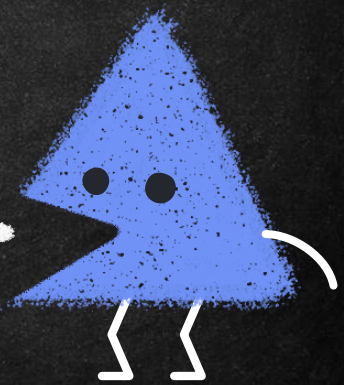
```
7
ABC
BD.
CEF
E..
F.G
D..
G..
ABDCEFG
DBAECFG
DBEGFCA
```



함께 풀어보면 좋은 문제들!!!

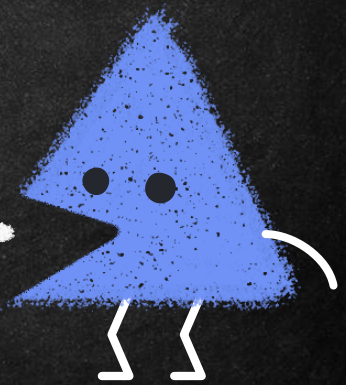
1. 별찍기 10
2. 에너지 모으기
3. 트리
4. 이진수 변환
5. 하노이의 탑(강의에서 다룬 내용)
6. 트리 순회(강의에서 다룬 내용)

➔ 백준 그룹 문제집에 포함되어 있으니 확인해서 풀면 됩니다!!!



NOTICE!!!

- 이번주부터는 5개의 주제에 대해서 문제풀이가 진행됩니다.
- 스터디 참여 인증을 위해서 코드 제출을 받습니다.
- 이름.cpp파일을 카카오톡으로 보내주시면 됩니다.
- 적어도 하나의 문제를 풀어서(그 이상이면 주석 처리, 합쳐서) 보내주시면 확인후에 점검합니다.
- 앞으로의 5주 중에서 2회 초과로 미제출시 스터디 미참여로 간주하고 불이익이 있습니다. (2번까지는 참여인정)
- 문제는 PDF 후반에 있는 <함께 풀어보면 좋은 문제들>중에서 풀어주시면 됩니다.



THANK YOU