#1 recursion

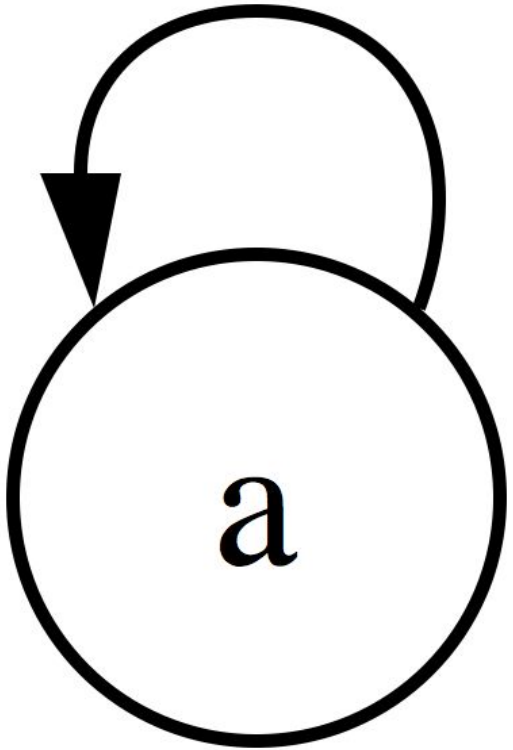
진민성

재귀함수

재귀 함수: 자기 자신을 호출하는 함수

- 예시: f(n) = f(n-1) + f(n-2)

- 용도: 점화식 계산, 완전 탐색,
  반복문으로 반복하기 힘든 작업, ...



재귀함수

주의사항

- 재귀 호출을 하지 않는 조건(종료 조건)이 있어야 함 - 없으면 프로그램이 종료되지 않음

- 재귀 호출 과정에 사이클이 없어야 함
  - f(a) → f(b) → f(c) → ... → f(a) - 프로그램이 종료되지 않음

- 종료 조건에 가까워지는 방향으로 설계

재귀 함수 예시 1

Factorial함수를 재귀를 이용해서 구현해보자

- Factorial함수는 다음과 같은 점화식으로 표현 할 수 있다.
- f(n) = n * f(n-1)
- 종료조건으로 n이 1일때 1을 반환하도록 한다.

```cpp
#include <bits/stdc++.h>
using namespace std;

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

int main() {
  cout << factorial(5) << '\n'; // 120
}
```

재귀함수 예시 2

Fibonacci 수열은 첫째 및 둘째 항이 1이며 그 뒤의 모든 항은 바로 앞 두 항의 합인 수열

n번째 Fibonacci수를 구하는 함수를 재귀로 구현해보자.

- 피보나치 수는 다음과 같은 점화식으로 표현할 수 있다.
- f(n) = f(n-1) + f(n-2)

정의에 의해서 종료조건은 1,2번째 피보나치 수일때 1을 리턴하면 됨

```cpp
#include <bits/stdc++.h>
using namespace std;

int fibonacci(int n) {
  if(n<=2) return 1;
  return fibonacci(n-1) + fibonacci(n-2);
}

int main() {
  // 1 1 2 3 5 8 13 ...
  cout << fibonacci(4) << '\n'; // 3
}
```

BOJ 17829. 종이의 개수

N×N크기의 행렬로 표현되는 종이가 있다. 종이의 각 칸에는 -1, 0, 1 중 하나가 저장되어 있다.

1. 만약 종이가 모두 같은 수로 되어 있다면 이 종이를 그대로 사용한다.
2. 1이 아닌 경우에는 종이를 같은 크기의 종이 9개로 자르고, 각각의 잘린 종이에 대해서 1의 과정을 반복한다.

하나의 수로만 채워진 종이의 개수를 구하는 문제

BOJ 17829. 종이의 개수

크기 NxN의 종이가 있다고 하자.

그러면 n/3의 크기를 가진 종이 9개로 자를 수 있고, 각 종이를 (-1의 개수, 0의 개수, 1의 개수)로 표현하자.

전체 종이의 개수는 9 구역의 종이의 개수를 모두 합친 것과 같다.

하지만 만약에 (0,0,9), (0,9,0), (9,0,0)처럼 표현 되는 종이가 발생하면 하나의 종이로 표현해야하므로 반환할때 (0,0,1), (0,1,0), (1,0,0)으로 반환하면 된다.

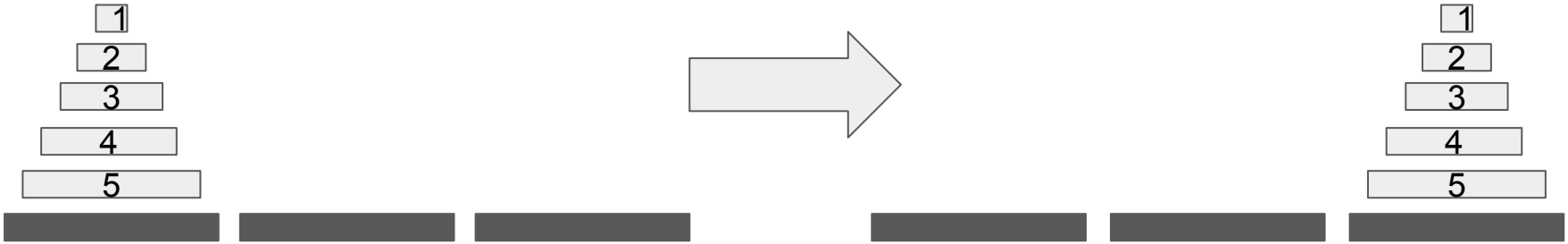
0	0	0	1	1	1	-1	-1	-1
0	0	0	1	1	1	-1	-1	-1
0	0	0	1	1	1	-1	-1	-1
1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0
0	1	-1	0	1	-1	0	1	-1
0	-1	1	0	1	-1	0	1	-1
0	1	-1	1	0	-1	0	1	-1

BOJ 17829. 종이의 개수

```cpp
#include <bits/stdc++.h>
using namespace std;

int A[3030][3030];

tuple<int,int,int> f(int n, int x=0, int y=0) {
    if(n==1) return {A[x][y]==-1, !A[x][y], A[x][y]==1};
    n/=3;
    vector<pair<int,int>> v={{0,0},{0,n},{0,n*2},{n,0},{n,n},{n,2*n},{2*n,0},{2*n,n},{2*n,2*n}};
    int a=0,b=0,c=0;
    for(auto [i,j]:v) {
        auto [d,e,g]=f(n,x+i,y+j);
        a+=d, b+=e, c+=g;
    }
    if((a+b+c) == 9 && max({a,b,c}) == 9) return {a/9, b/9, c/9};
    return {a,b,c};
}

int main() {
    cin.tie(0)->sync_with_stdio(0);
    int n; cin >> n;
    for(int i=0;i<n;i++) for(int j=0;j<n;j++) cin >> A[i][j];
    auto [a,b,c]=f(n);
    cout << a << '\n' << b << '\n' << c;
    return 0;
}
```

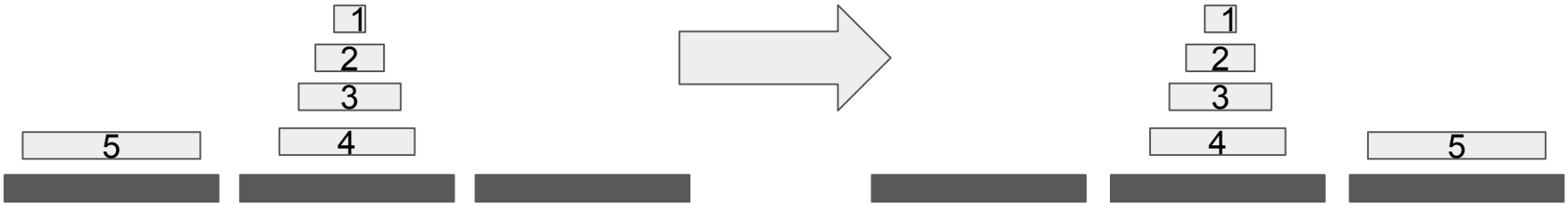
BOJ 11729. 하노이 탑 이동 순서

- 3개의 기둥과 크기가 서로 다른 N개의 원판이 있음
- 원판 위에는 자기 자신보다 작은 원판만 올라갈 수 있음
- 원판을 최소한으로 이동해서 첫 번째 기둥에 있는 모든 원판을 세 번째 기둥으로 옮기는 문제



BOJ 11729. 하노이 탑 이동 순서

- 1번 기둥의 N-1개 원판을 2번 기둥으로 옮기고
- N번 원판을 3번으로 옮긴 후에
- 2번 기둥의 N-1개의 원판을 3번으로 옮기면 됨.



BOJ 11729. 하노이 탑 이동 순서

N개의 원판을 a에서 c로 옮기기 위해서

N-1개의 원판을 a에서 b로 옮기고

N번째 원판을 a에서 c로 옮긴 후에

다시 N-1개의 원판을 b에서 c로 옮기면 끝!

```cpp
#include <bits/stdc++.h>
using namespace std;

void hanoi(int n, int a=1, int b=2, int c=3) {
  if(n == 1) {
    cout << a << ' ' << c << '\n';
    return;
  }
  f(n-1,a,c,b);
  cout << a << ' ' << c << '\n';
  f(n-1,b,a,c);
}

int main() {
  int n;
  cin >> n;
  hanoi(n);
}
```