

# 재귀와 백트래킹

# 수학적 귀납법

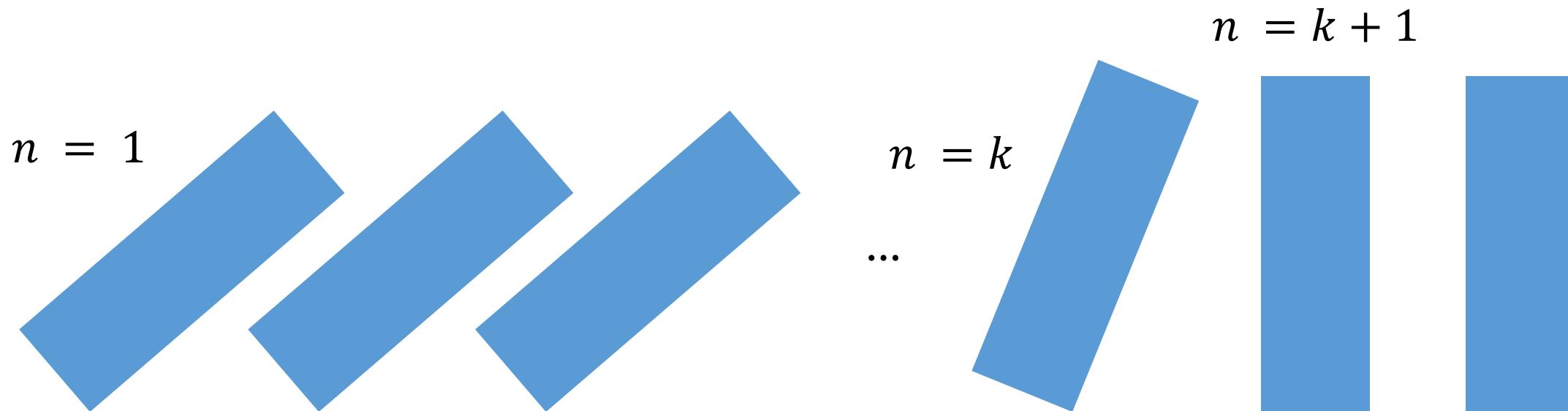
- 모든 자연수에 대해 주어진 명제가 성립함을 증명하는 방법
- 연달아 쓰러지는 도미노처럼, **연속적으로 논리를 전개**

# 수학적 귀납법

- 아래의 두 성질을 만족시키는 명제  $P$ 는 모든 자연수  $n$ 에 대해  $P(n)$ 이 성립한다.

I.  $P(1)$ 이 성립한다. (**Base case**)

II.  $P(k)$ 가 성립한다면,  $P(k + 1)$ 이 성립한다. (**step case**)



# 수학적 귀납법 - 예시

- 홀수의 합 공식

$1 + 3 + 5 + \dots + (2n - 1) = n^2$ 이 모든 자연수  $n$ 에 대해 성립함을 증명하자.

# 수학적 귀납법 - 예시

$1 + 3 + 5 + \dots + (2n - 1) = n^2$ 이 모든 자연수  $n$ 에 대해 성립함을 증명하자.

$n = 1$ 일 때,  $1 = 1^2$ 이 성립한다. (**base case**)

$n = k$ 일 때,  $1 + 3 + 5 + \dots + (2k - 1) = k^2$ 이 성립한다고 가정하면,

$n = k + 1$ 일 때,

$$1 + 3 + 5 + \dots + (2k - 1) + (2(k + 1) - 1)$$

$$= k^2 + 2k + 1$$

$= (k + 1)^2$ 이 성립한다. (**step case**)

따라서 수학적 귀납법에 의해 모든 자연수  $n$ 에 대하여 홀수의 합 공식이 성립한다.

# 재귀함수

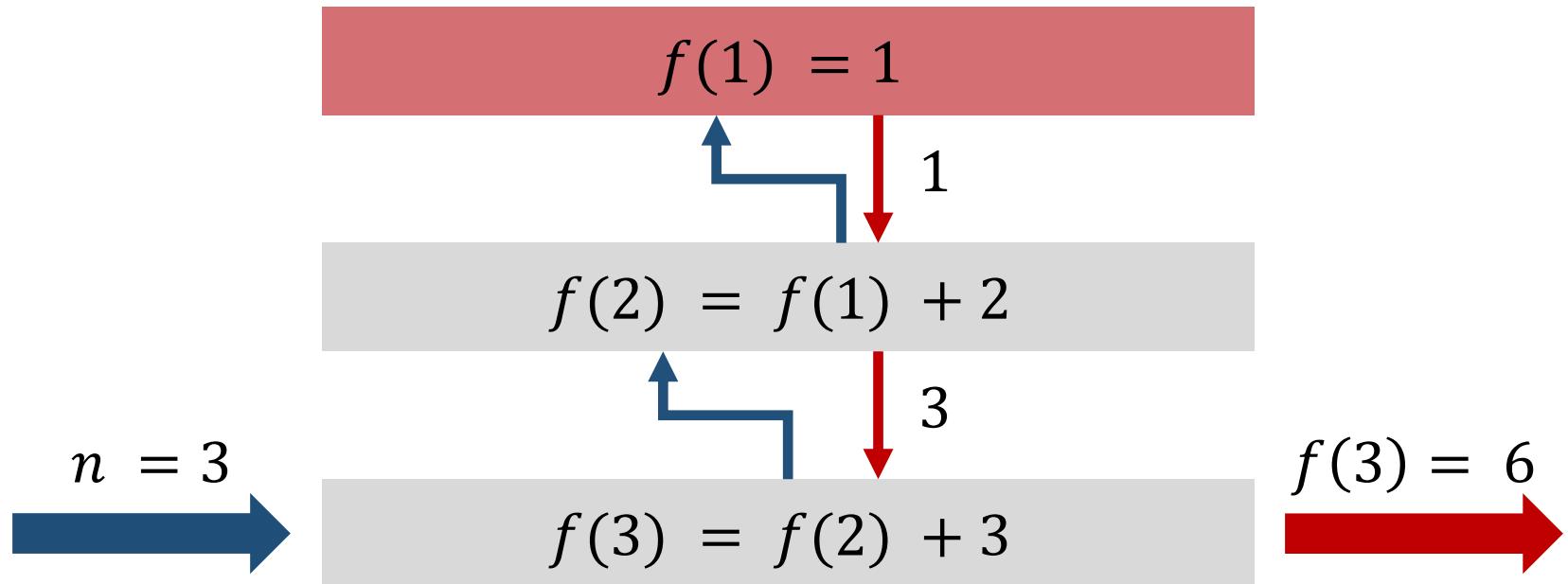
- 자기 자신을 호출할 수 있는 함수
- **Base case** (재귀 호출을 하지 않을 조건) 존재



```
// Add from 1 to n
int n;

int f(int x) {
    if (x == 1) // Base case
        return 1;
    return f(x - 1) + x;
}

cout << f(n);
```



# 재귀함수 – 어디에 쓰는가?

- 트리 등의 그래프 순회
- 백트래킹, 분할 정복 알고리즘
- 세그먼트 트리

# 수학적 귀납법과 재귀함수

- 재귀함수의 작동 원리는 수학적 귀납법과 유사
- 재귀함수의 Base case는 수학적 귀납법에서  $N = 1$ 인 경우 (base case)
- 재귀함수에서 재귀적으로 가져온 값은 수학적 귀납법에서의 step case

# 질문

# 팩토리얼 2 BOJ 27433

- $0 \leq N \leq 20$ 인 정수  $N$ 이 주어진다.
- $N!$ 을 출력하자.

(참고)  $N! = 1 \times 2 \times 3 \times \cdots \times N$

(주의)  $0! = 1$

# 팩토리얼 2 BOJ 27433

- $N! = (N - 1)! \times N$
- Base case:  $N = 0$  또는  $N = 1$ 일 때,  $N! = 1$



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

int factorial(int x) {
    if (x == 0 || x == 1) // Base case
        return 1;
    return factorial(x - 1) * x;
}

int main() {
    int n;
    cin >> n;
    cout << factorial(n);
}
```

# 팩토리얼 2 BOJ 27433

- 맞왜틀???

제출 번호	아이디	문제	문제 제목	결과
74803635	motsuni04	27433	팩토리얼 2	틀렸습니다

- $20! = ?$

# 팩토리얼 2 BOJ 27433

- $20! \approx 2.4 \times 10^{18}$
- int:  $\sim 2,147,483,647$  ( $\approx 21$ 억)
- long long:  $\sim 9,223,372,036,854,775,807$   
 $(\approx 9 \times 10^{18})$



```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

ll factorial(ll x) {
    if (x == 0 || x == 1) // Base case
        return 1;
    return factorial(x - 1) * x;
}

int main() {
    ll n;
    cin >> n;
    cout << factorial(n);
}
```

# 피보나치 수 5 BOJ 10870

- $0 \leq n \leq 20$ 인 정수  $n$ 이 주어진다.
- $n$ 번째 피보나치 수를 구하자.

(예시)  $n = 8$

0, 1, 1, 2, 3, 5, 8, 13, **21**

# 피보나치 수 5 BOJ 10870

- Base case: 0번째, 1번째 피보나치 수는 0, 1이다.
- Step case:  $n$ 번째 피보나치 수는  $(n - 1)$ 번째와  $(n - 2)$ 번째 피보나치 수의 합이다.

# 피보나치 수 5 BOJ 10870



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

int fibonacci(int x) {
    if (x == 0) return 0;
    if (x == 1) return 1;
    return fibonacci(x - 2) + fibonacci(x - 1);
}

int main() {
    int n;
    cin >> n;
    cout << fibonacci(n);
}
```

# 질문

# 재귀함수 – 왜 쓰는가?

- 구현 방식의 편리함, 코드의 간결성
- 재귀함수가 아닌 방법으로 해결하기 어려운 경우 존재

## 예제 출력 1 [복사](#)

어느 한 컴퓨터공학과 학생이 유명한 교수님을 찾아가 물었다.

"재귀함수가 뭔가요?"

"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.

마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.

그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."

\_\_\_\_ "재귀함수가 뭔가요?"

\_\_\_\_ "잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.

\_\_\_\_ 마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.

\_\_\_\_ 그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."

----- "재귀함수가 뭔가요?"

----- "재귀함수는 자기 자신을 호출하는 함수라네"

----- 라고 답변하였지.

----- 라고 답변하였지.

라고 답변하였지.

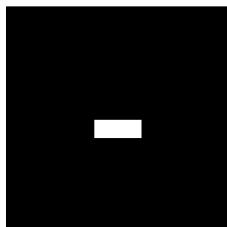
→ 재귀함수가 뭔가요? BOJ 17478

# 칸토어 집합 BOJ 4779

- 정수  $N$  ( $0 \leq N \leq 12$ )이 주어진다.
- ' $'-$ '이  $3^N$ 개 있는 문자열을 만든다.
- 문자열을 3등분하고, 가운데 부분을 전부 공백으로 바꾼다.
- 남아있는 선들을 다시 3등분하고, 가운데를 공백으로 바꾸는 것을 반복한다.
- 모든 남아있는 선들의 길이가 1일 때까지 반복한다.

# 칸토어 집합 BOJ 4779

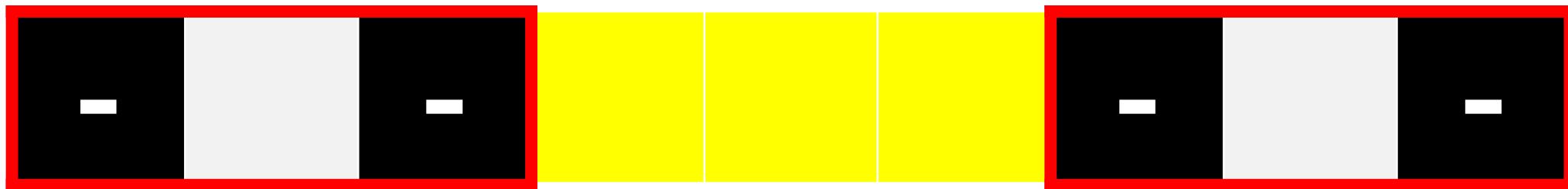
$N = 0$



$N = 1$



$N = 2$



# 칸토어 집합 BOJ 4779

- Base case:  $N = 0$ 일 때,  $f(0) = '-'$
- $f(N + 1) = f(N) + \textcolor{red}{' '} \times 3^{N-1} + f(N)$



```
string(pow(3, x - 1), ' ')
```

# 칸토어 집합 BOJ 4779



- 분할 정복 알고리즘

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

string f(int x) {
    if (x == 0) return "-";
    return f(x - 1) + string(pow(3, x - 1), ' ') + f(x - 1);
}

int main() {
    int n;
    while (cin >> n)
        cout << f(n) << '\n';
}
```

# 특별상이라도 받고 싶어\* BOJ 24460

- 도전문제! 세미나 후 풀어봅시다.
- 정수  $N = 2^m$  ( $0 \leq m \leq 10$ ,  $m$ 은 정수)이 주어진다.
- 이후  $N$ 개의 줄에  $N \times N$ 개의 의자에 적힌 서로 다른 추첨번호가 각각 주어진다.
- 특별상을 받을 수 있는 사람이 한 명이라면, 그 사람이 뽑힌다.
- 그렇지 않다면, 구역을 4개의 정사각형으로 나누어 각 구역에서 이 규칙을 재귀적으로 적용한다. 이렇게 뽑힌 4명 중 추첨번호가 두 번째로 작은 사람이 뽑힌다.

# 질문

# 재귀함수 – 사용 시 주의점

- Base case를 통한 종료 조건을 반드시 선언  
→ stack overflow!
- 재귀 최대 깊이에 주의  
→ 파이썬 사용자의 경우, 명시적 설정 필요
- 재귀는 반복문보다 느리다!  
→ 파이썬 사용자의 경우 특히 주의



```
import sys
sys.setrecursionlimit(10 ** 6)
```

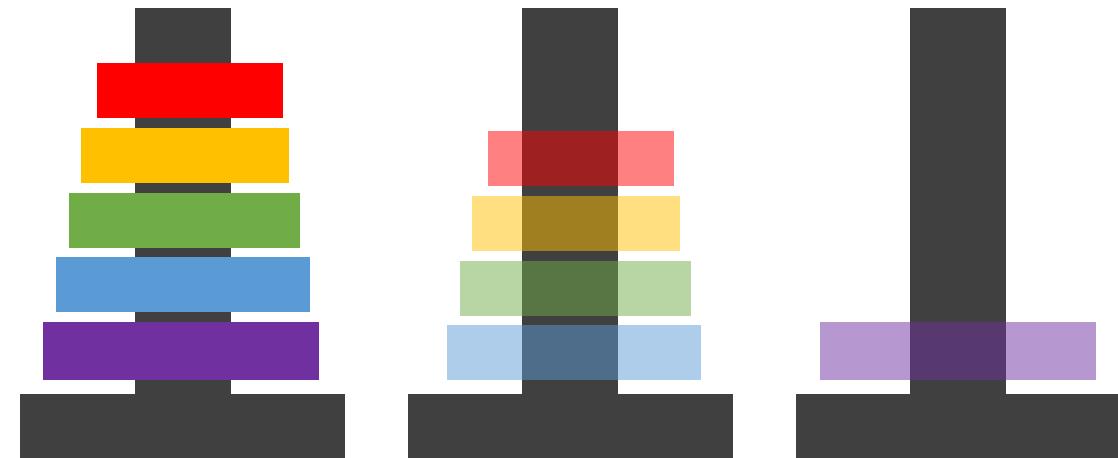
# 하노이 탑 이동 순서 BOJ 11729

- $1 \leq N \leq 20$ 인 원판의 개수  $N$ 이 주어진다.
- 장대 3개가 존재한다. 원판은 1번째 장대에 놓여 있고, 이들을 3번째 장대로 옮겨야 한다.
- 원판은 항상 크기가 증가하는 순서대로 놓여야 한다.
- 옮긴 횟수를 최소화하는 방법을 찾아보자!



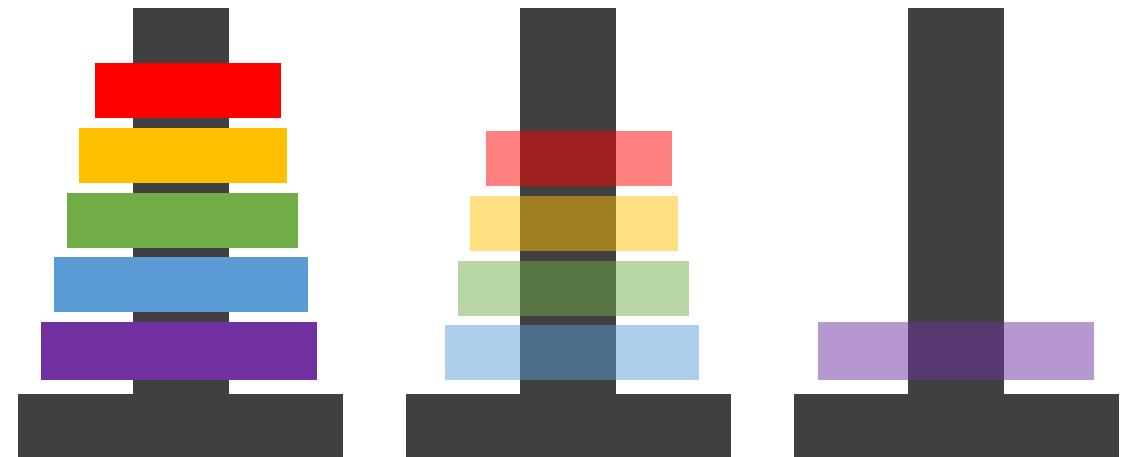
# 하노이 탑 이동 순서 BOJ 11729

- 항상 더 큰 원판이 아래에 오기에, 가장 큰 원판을 우선해서 옮겨야 한다.
- 가장 큰 원판을 옮기려면? 다른 모든 원판을 다른 빈 장대로 옮겨야 한다.
- 다른 모든 원판을 옮기려면? 우선 그 중 가장 큰 원판을 우선해서 옮겨야 한다.
- ... 끝이 없다!



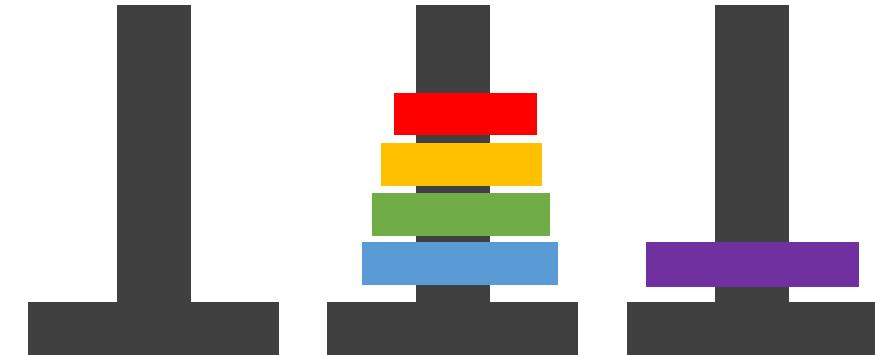
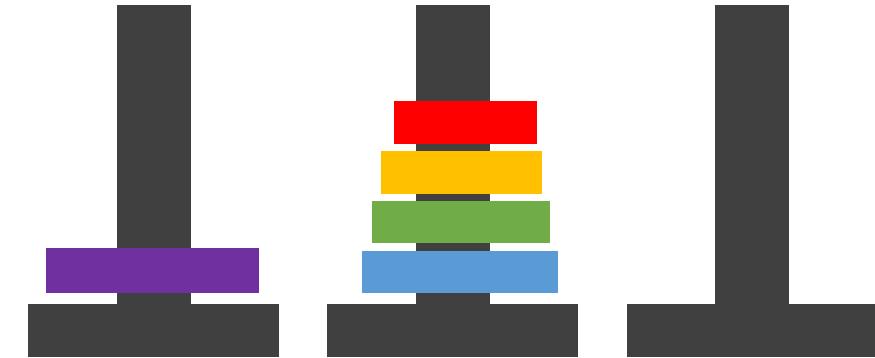
# 하노이 탑 이동 순서 BOJ 11729

- 재귀적 사고를 해보자
- 가장 작은 원판을 옮기는 것은 간단하다 (base case)
- $N$ 번째로 큰 원판을 옮기는 방법은?
- $N - 1$ 번째로 큰 원판까지 옮기는 것이 가능하다고 가정한다
- 이후  $N$ 번째 원판을 옮기는 것이 가능하다  
(step case)



# 하노이 탑 이동 순서 BOJ 11729

- $N$ 번째 원판을 기둥  $A$ 에서  $B$ 로 옮기려면?
- $N = 1$ 이라면, 바로 옮기면 된다.
- $N > 1$ 이라면,
  - $A, B$ 가 아닌 남은 하나의 기둥을  $X$ 라고 하자.
  - $N - 1$ 개의 원판을 먼저  $A$ 에서  $X$ 로 옮긴다.
  - $N$ 번째 원판을  $A$ 에서  $B$ 로 옮긴다.
  - $N - 1$ 개의 원판을  $X$ 에서  $B$ 로 옮긴다.



# 하노이 탑 이동 순서 BOJ 11729

- 실행 횟수는?
- $f(1) = 1$
- $f(n) = 2f(n - 1) + 1$
- 1, 3, 7, 15, 31, ...
- $f(n) = 2^n - 1$

# 하노이 탑 이동 순서 BOJ 11729

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

void f(int n, int a, int b) {
    if (n == 1) {
        cout << a << ' ' << b << '\n';
    } else {
        int x = (a != 1 && b != 1) ? 1 : (a != 2 && b != 2) ? 2 : 3;
        f(n - 1, a, x);
        cout << a << ' ' << b << '\n';
        f(n - 1, x, b);
    }
}

int main() {
    int n; cin >> n;
    int k = 1;
    for (int i = 0; i < n; i++) k *= 2;
    cout << k - 1 << '\n';
    f(n, 1, 3);
}
```

# 질문

# 별 찍기 - 10\* BOJ 2447

$N = 27$

- 도전문제!
- $N = 3^k (0 \leq k \leq 7)$ 이 주어진다.  
 $N \times N$  크기의 패턴을 출력해 보자.
- 힌트: 각 줄을 바로 출력하지 말고, 배열을 사용해 보자

$N = 9$

```
*****  
* * * * *  
*****  
*** ***  
* * * *  
*** ***  
*****  
* * * * *  
*****
```

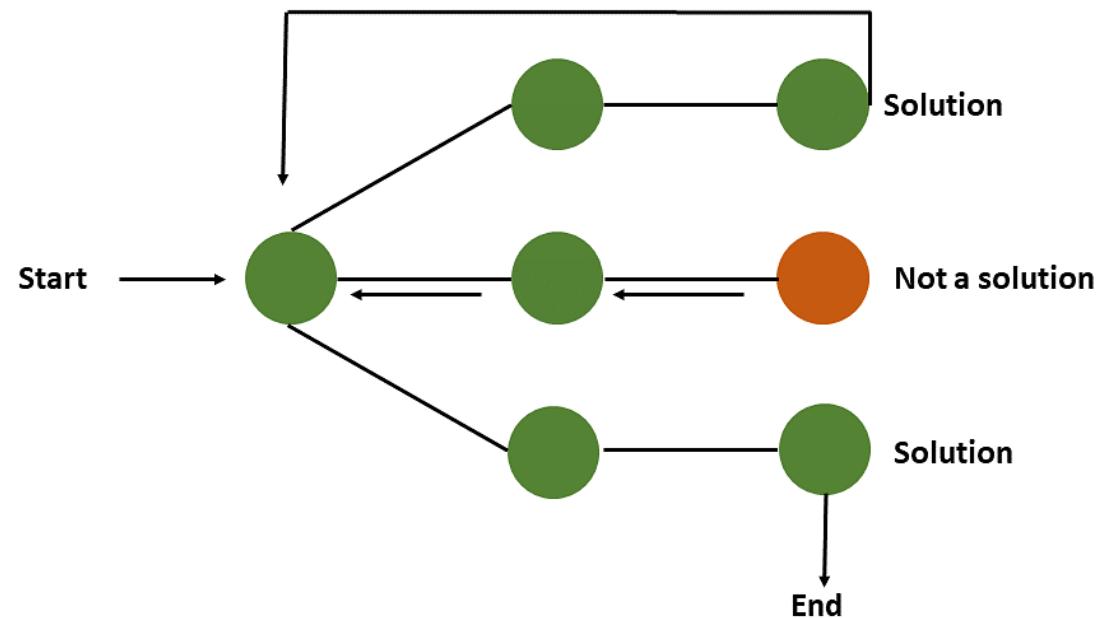
```
*****  
* * * * *  
*****  
*** ***  
* * * * *  
*** ***  
*****  
* * * * *  
*****  
*****  
* * * * *  
*****  
*****  
* * * * *  
*****  
*****  
* * * * *  
*****  
*****  
* * * * *  
*****  
*****  
* * * * *  
*****  
*****  
* * * * *  
*****  
*****
```

# Moo 게임\* BOJ 5904

- 도전문제!
- $S(0) = "m\ o\ o"$
- $S(k) = S(k - 1) + "m\ o\ ... \underset{k+2\text{개}}{o}" + S(k - 1)$
- 이 수열을 무한대로 확장했을 때,  $N$ 번째 글자는? ( $1 \leq N \leq 10^9$ )

# 백트래킹

- 모든 경우의 수를 전부 고려하는 알고리즘
- 시간 복잡도가 보통 지수적으로 증가하기에 제한이 작을 때 사용 가능
- 보통 재귀함수를 사용하여 구현
- 구현이 길다 🤯



# N과 M (3) BOJ 15651

- $1 \leq M \leq N \leq 7$ 인 자연수  $N, M$ 이 주어진다.
- 1부터  $N$ 까지의 자연수 중에서  $M$ 개를 고른 수열을 모두 출력하자.
- 같은 수를 여러 번 골라도 된다.
- 수열들은 사전 순으로 출력한다.
- 사전 순:  $(1, 1, 1), (1, 1, 2), (1, 1, 3), (1, 2, 1), \dots, (3, 3, 2), (3, 3, 3)$

# N과 M (3) BOJ 15651

- for문 M번 돌리기???



```
int main() {
    for (int a = 1; a <= n; a++)
        for (int b = 1; b <= n; b++)
            for (int c = 1; c <= n; c++)
                for (int d = 1; d <= n; d++)
                    ...
}
```

# N과 M (3) BOJ 15651

- $M = N = 7$ 인 경우, 가능한 수열의 경우의 수는  $7^7 = 823,543$ 가지이다.
- 따라서, 백트래킹 기법으로 문제를 해결할 수 있다.
- 가능한 모든 경우를 탐색하는 재귀함수를 작성하자!

# N과 M (3) BOJ 15651

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

int n, m;

void f(vector<int> a) {
    if (static_cast<int>(a.size()) == m) {
        for (int i : a) cout << i << ' ';
        cout << '\n';
    } else {
        for (int i = 1; i <= n; i++) {
            a.push_back(i);
            f(a);
            a.pop_back();
        }
    }
}

int main() {
    cin >> n >> m;
    f(vector<int>());
}
```

# 질문

# 차이를 최대로 BOJ 10819

- $N$  ( $N \leq 8$ )과  $N$ 개의 정수로 이루어진 배열  $A$ 가 주어진다.
- 배열에 들어있는 수의 순서를 원하는 대로 바꾼다.
- 이때, 얻을 수 있는 아래 식의 최댓값은?
  - $|A[0] - A[1]| + |A[1] - A[2]| + \dots + |A[N - 2] - A[N - 1]|$
- 이 문제를 빠르게 해결하는 방법을 떠올릴 수 있는가?

# 차이를 최대로 BOJ 10819

- $N \leq 8$
- $A$ 를 배열할 수 있는 경우의 수는?
- $A$ 를 배열한 각 경우에 대해 점수를 계산하는 연산의 횟수는?

# 차이를 최대로 BOJ 10819

- $N \leq 8$
- $A$ 를 배열할 수 있는 경우의 수는?  $8! = 40,320$
- 점수를 계산하는 연산의 시간 복잡도는?  $\approx O(1)$
- 전부 해보자, 백트래킹!

# 차이를 최대로 BOJ 10819

- `next_permutation`
- “정렬된” 수열을 받아와서 해당 수열의 순열을 생성
- 중복 원소는 제외
- do-while문과 사용



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

int main() {
    vector<int> v = {1, 2, 3};
    sort(v.begin(), v.end());

    do {
        for (int i : v) cout << i << ' ';
        cout << endl;
    } while (next_permutation(v.begin(), v.end()));
}
```

# 차이를 최대로 BOJ 10819

- 이제 단순한 구현!

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

int main() {
    int n; cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    sort(a.begin(), a.end());

    int result = 0;
    do {
        int score = 0;
        for (int i = 0; i < n - 1; i++)
            score += abs(a[i] - a[i + 1]);
        result = max(result, score);
    } while (next_permutation(a.begin(), a.end()));
    cout << result;
}
```

# 질문

# 연산자 끼워넣기 BOJ 14888

- 도전문제!
- $N$ 개의 숫자와  $N - 1$ 개의 연산자가 주어진다.
- 이들을 배치해서 얻을 수 있는 식의 최대 / 최소값은?
- 단, 식은 앞에서부터 우선순위를 무시하고 계산한다.
- $N \leq 11$

1 3 5 7 8 9 11

+ - \* /  
+  
+

# 월드컵\* BOJ 6987

- 도전문제!
- 6개의 국가가 각 쌍마다 1번씩, 총 30번의 경기를 치른다.
- 각 국가의 승, 무, 패 횟수가 주어질 때, 이 경기 결과가 가능한지 구해보자!

국가	승	무	패
A	5	0	0
B	2	2	1
C	2	0	3
D	0	0	5
E	4	0	1
F	1	0	4

국가	승	무	패
A	5	0	0
B	3	0	2
C	2	0	3
D	0	0	5
E	4	0	1
F	1	0	4

# 연습 문제

17478 : 재귀함수가 뭔가요?

24460 : 특별상이라도 받고 싶어

2447\* : 별 찍기 - 10

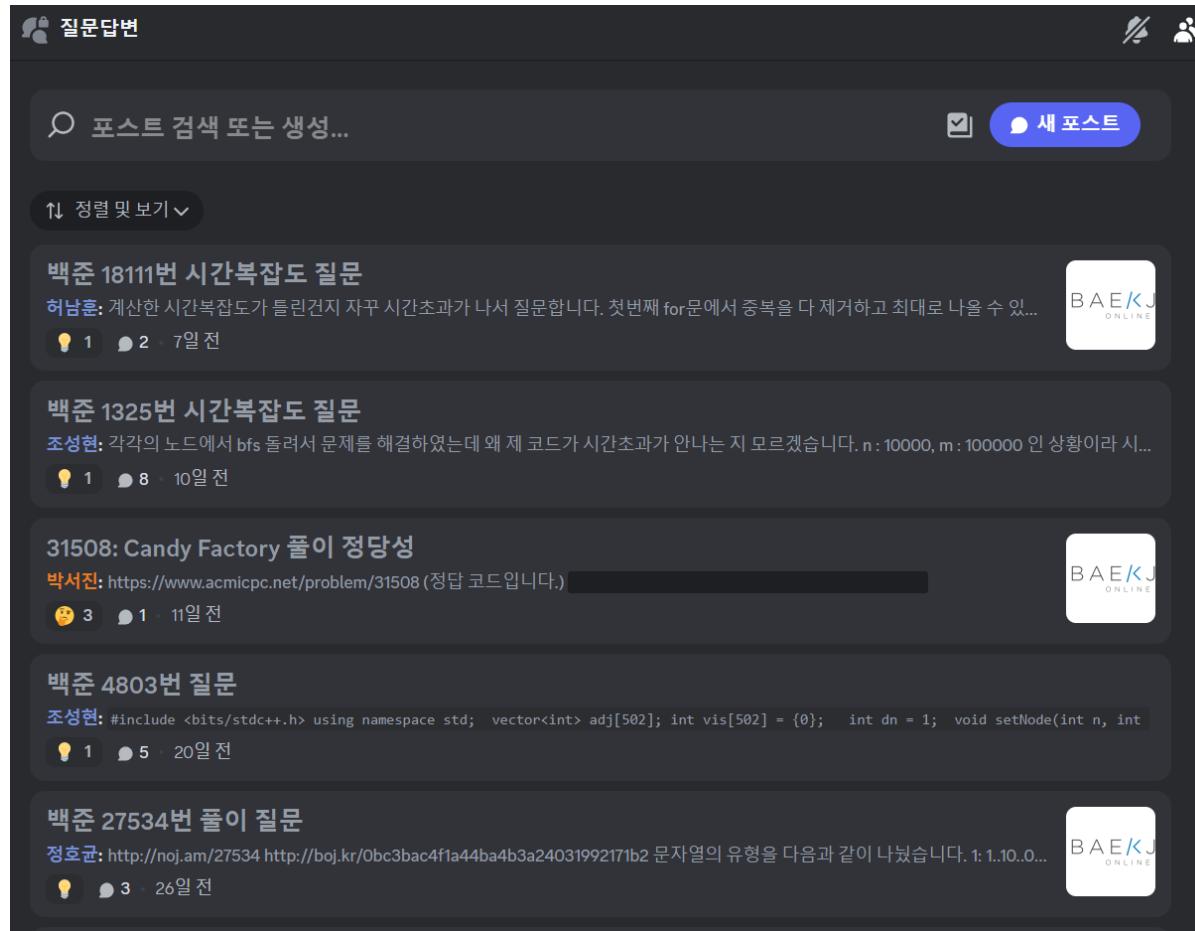
5904\* : Moo 게임

14888 : 연산자 끼워넣기

6987\* : 월드컵

9663\* : N-Queen

# 풀이 중 모르는 점은 질문답변 채널로!



# References

- [github.com/justiceHui/SSU-SCCC-Study](https://github.com/justiceHui/SSU-SCCC-Study)
- [github.com/KU-Alkon/study](https://github.com/KU-Alkon/study)
- [ko.wikipedia.org](https://ko.wikipedia.org)

# 재귀와 백트래킹

발표, 자료 제작: 박서진

감사합니다.