JavaScript 백트래킹 알고리즘

백트래킹 문제 풀이

백트래킹 문제 풀이 | 코딩 테스트에서 자주 등장하는 백트래킹 알고리즘 이해하기

강사 나동빈

JavaScript
백트래킹 알고리즘

백트래킹 문제 풀이

N과 M (1): https://www.acmicpc.net/problem/15649

JavaScript 백트래킹
백트래킹 문제 풀이

혼자 힘으로 풀어보기

문제 제목: N과 M (1)

문제 난이도: ★★☆☆☆

문제 유형: 백트래킹, 경우의 수, 완전 탐색

추천 풀이 시간: 40분

JavaScript 백트래킹
백트래킹 문제 풀이

문제 해결 아이디어

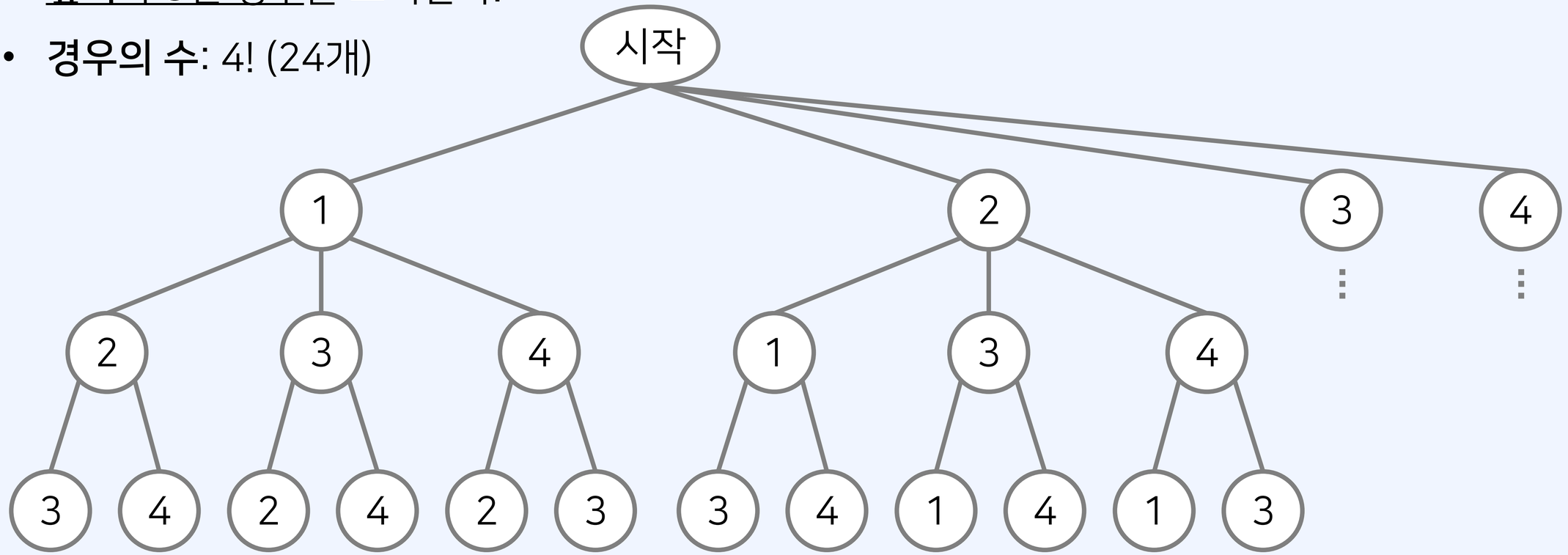
- 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 **모든 수열**을 계산한다.
- 이 문제는 모든 순열을 구하는 문제로 이해할 수 있다.

N과 M (1): https://www.acmicpc.net/problem/15649

JavaScript 백트래킹
백트래킹 문제 풀이

문제 해결 아이디어

- 4개의 수 [1, 2, 3, 4]에서 3개를 고르는 **모든 순열**을 고려해 보자.
- **깊이**가 3인 경우를 고려한다.
- **경우의 수**: 4! (24개)

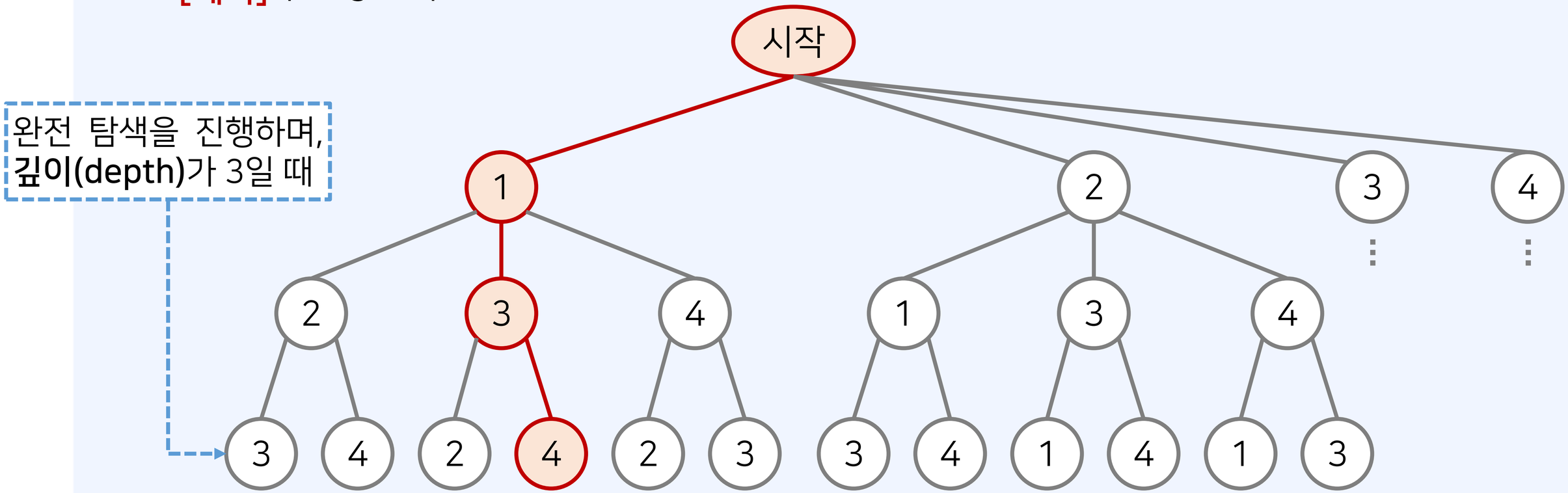


N과 M (1): https://www.acmicpc.net/problem/15649

문제 해결 아이디어

JavaScript 백트래킹
백트래킹 문제 풀이

- 4개의 수 [1, 2, 3, 4]에서 3개를 고르는 **모든 순열**을 고려해 보자.
- **[예시]** 1 → 3 → 4



N과 M (1): https://www.acmicpc.net/problem/15649

문제 해결 아이디어

JavaScript
백트래킹
백트래킹
문제 풀이

- 모든 순열의 수를 고려하기 위해 **재귀 함수(백트래킹)**를 사용할 수 있다.
- 하나의 순열을 트리(tree)에서 리프 노드까지의 경로로 생각할 수 있다.

→ 이때, M개의 원소를 뽑는 순열을 고려하는 것이므로, **깊이(depth)**는 M과 같다.

- 원소를 **중복하여 선택하지 않으므로**, 방문 처리(visited) 배열을 사용한다.

→ 한 번 선택된 원소는 다음 재귀 함수에서 다시 선택되지 않는다.

N과 M (1): https://www.acmicpc.net/problem/15649

정답 코드 예시

JavaScript 백트래킹
백트래킹 문제 풀이

```
let fs = require('fs');
let input = fs.readFileSync('/dev/stdin').toString().split('\n');

let [n, m] = input[0].split(" ").map(Number); // 1부터 N까지 자연수 중에서 중복 없이 M개를 고른 수열
let arr = []; // 순열을 계산하고자 하는 원소(element)가 담긴 배열
for (let i = 1; i <= n; i++) arr.push(i);
let visited = new Array(n).fill(false); // 각 원소 인덱스(index)별 방문 여부
let selected = []; // 현재 순열에 포함된 원소(element)의 인덱스

let answer = "";
function dfs(arr, depth) {
  if (depth == m) { // 모든 순열을 확인하는 부분
    let result = []; // 순열(permutation) 결과 저장 테이블
    for (let i of selected) result.push(arr[i]);
    for (let x of result) answer += x + " "; // 계산된 순열을 실질적으로 처리하는 부분
    answer += "\n";
    return;
  }
  for (let i = 0; i < arr.length; i++) { // 하나씩 원소 인덱스(index)를 확인하며
    if (visited[i]) continue; // [중복을 허용하지 않으므로] 이미 처리 된 원소라면 무시
    selected.push(i); // 현재 원소 선택
    visited[i] = true; // 현재 원소 방문 처리
    dfs(arr, depth + 1); // 재귀 함수 호출
    selected.pop(); // 현재 원소 선택 취소
    visited[i] = false; // 현재 원소 방문 처리 취소
  }
}
dfs(arr, 0);
console.log(answer);
```

모든 순열: https://www.acmicpc.net/problem/10974

혼자 힘으로 풀어보기

JavaScript
백트래킹
백트래킹
문제 풀이

문제 제목: 모든 순열

문제 난이도: ★★☆☆☆

문제 유형: 백트래킹, 경우의 수, 완전 탐색

추천 풀이 시간: 40분

모든 순열: https://www.acmicpc.net/problem/10974

JavaScript 백트래킹
백트래킹 문제 풀이

문제 해결 아이디어

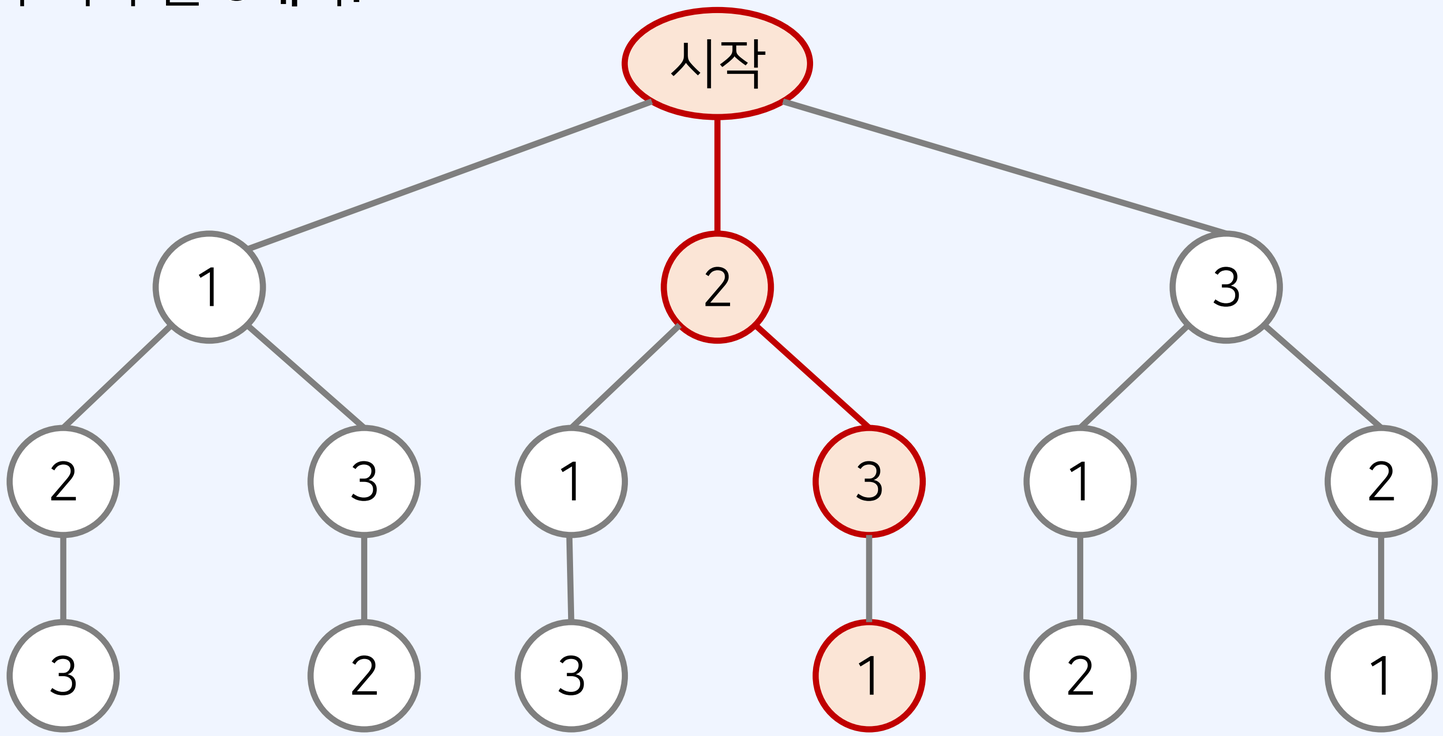
- N이 주어졌을 때, 1부터 N까지의 수로 이루어진 순열을 사전 순으로 출력한다.
- N의 값은 최대 8이다.

→ 이때 **모든 순열**을 출력하므로 최대 경우의 수는 8!개이다.

모든 순열: https://www.acmicpc.net/problem/10974

문제 해결 아이디어

- N의 값이 3일 때의 예시는 다음과 같다.
- 전체 경우의 수는 **6개**다.



모든 순열: https://www.acmicpc.net/problem/10974

JavaScript 백트래킹
백트래킹 문제 풀이

정답 코드 예시

```
let fs = require('fs');
let input = fs.readFileSync('/dev/stdin').toString().split('\n');

let n = Number(input[0]); // 1부터 N까지 자연수 중에서 중복 없이 N개를 고른 수열
let arr = []; // 순열을 계산하고자 하는 원소(element)가 담긴 배열
for (let i = 1; i <= n; i++) arr.push(i);
let visited = new Array(n).fill(false); // 각 원소 인덱스(index)별 방문 여부
let selected = []; // 현재 순열에 포함된 원소(element)의 인덱스

let answer = "";
function dfs(arr, depth) {
  if (depth == n) { // 모든 순열을 확인하는 부분
    let result = []; // 순열(permutation) 결과 저장 테이블
    for (let i of selected) result.push(arr[i]);
    for (let x of result) answer += x + " "; // 계산된 순열을 실질적으로 처리하는 부분
    answer += "\n";
    return;
  }
  for (let i = 0; i < arr.length; i++) { // 하나씩 원소 인덱스(index)를 확인하며
    if (visited[i]) continue; // [중복을 허용하지 않으므로] 이미 처리 된 원소라면 무시
    selected.push(i); // 현재 원소 선택
    visited[i] = true; // 현재 원소 방문 처리
    dfs(arr, depth + 1); // 재귀 함수 호출
    selected.pop(); // 현재 원소 선택 취소
    visited[i] = false; // 현재 원소 방문 처리 취소
  }
}
dfs(arr, 0);
console.log(answer);
```

혼자 힘으로 풀어보기

JavaScript
백트래킹
백트래킹
문제 풀이

문제 제목: 0 만들기

문제 난이도: ★★★☆☆

문제 유형: 백트래킹, 경우의 수, 완전 탐색

추천 풀이 시간: 50분

0 만들기: https://www.acmicpc.net/problem/7490

JavaScript 백트래킹
백트래킹 문제 풀이

문제 해결 아이디어

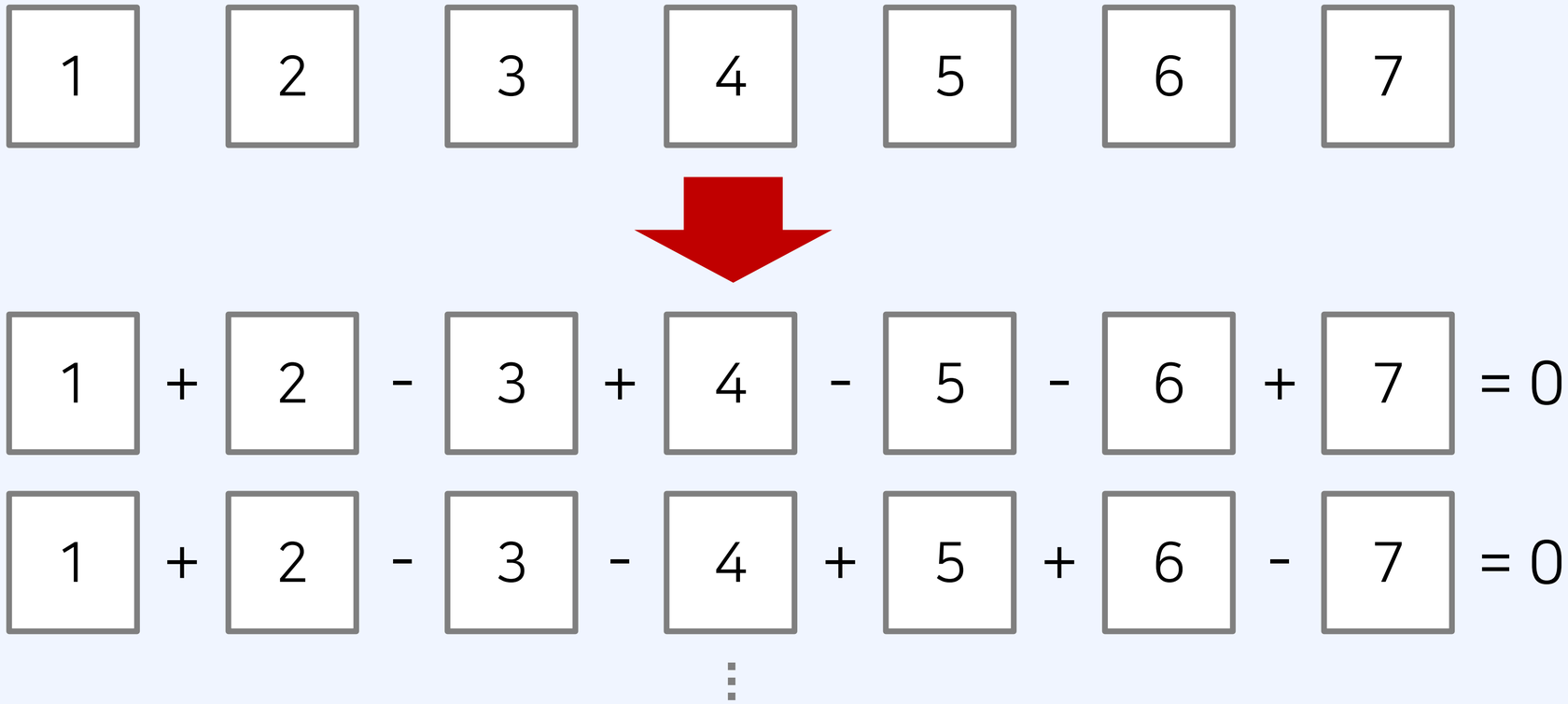
- 1부터 N까지의 수를 오름차순으로 쓴 수열 $[1, 2, 3, \dots\ N]$이 있다고 해보자.
- 이때 각 수 사이에 사용할 수 있는 연산으로는 다음의 **세 가지 연산**이 있다.

1. 더하기(+)
2. 빼기(-)
3. 숫자 이어 붙이기(공백)

- 결과적으로 N이 주어졌을 때, 수식의 결과가 0이 되는 모든 수식을 찾아야 한다.
- 테스트 케이스 및 자연수 N의 최댓값은 9이다.
- 3개의 연산 중에서 연속적으로 N번 선택하는 **중복 순열** 문제로 이해할 수 있다.
- 따라서 가능한 전체 경우의 수는 3^8이다.

문제 해결 아이디어

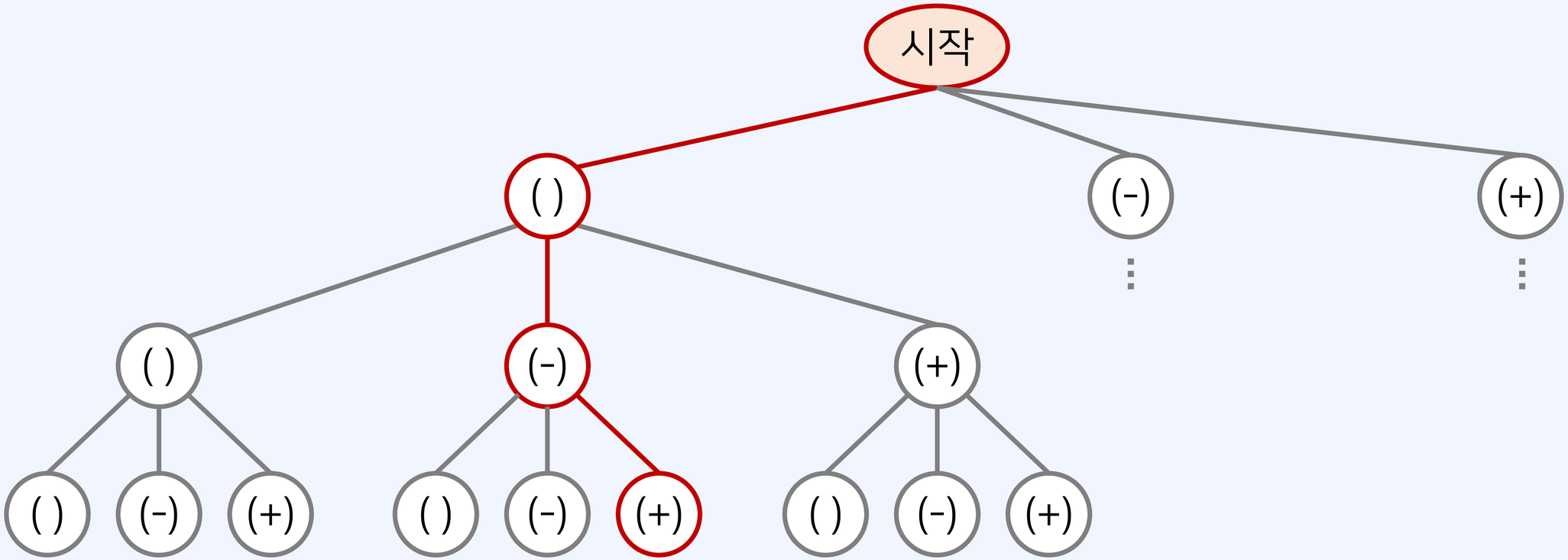
- 1부터 N까지의 수를 오름차순으로 쓴 수열 $[1, 2, 3, \dots\ N]$이 있다고 해보자.
- $N = 7$일 때의 예시는 다음과 같다.



0 만들기: https://www.acmicpc.net/problem/7490

문제 해결 아이디어

- $N = 4$일 때의 예시는 다음과 같다.
- 총 $3^3 = 27$**가지 경우의 수**가 존재한다.



정답 코드 예시

```
function dfs(result, depth) {
  if (depth == n - 1) { // 현재 순열(permutation) 처리(중복 순열)
    let str = ''; // 현재 수식 문자열
    for (let i = 0; i < n - 1; i++) str += arr[i] + result[i];
    str += arr[n - 1] + '';
    current = eval(str.split(' ').join('')); // 공백을 제거한 뒤에 수식 계산
    if (current == 0) console.log(str); // 수식의 결과가 0이 되는 경우 출력
    return;
  }
  for (let x of [' ', '+', '-']) { // 더하기(+), 빼기(-), 혹은 이어 붙이기( )
    result.push(x);
    dfs(result, depth + 1); // 재귀 함수 호출
    result.pop();
  }
}
```

0 만들기: https://www.acmicpc.net/problem/7490

JavaScript 백트래킹
백트래킹 문제 풀이

정답 코드 예시

```
let fs = require('fs');
let input = fs.readFileSync('/dev/stdin').toString().split('\n');

let testCase = Number(input[0]);
let n = 0;
let arr = [];
for (let tc = 1; tc <= testCase; tc++) { // 각 테스트 케이스 처리
  n = Number(input[tc]); // 자연수(N) 입력받기
  arr = [];
  for (let i = 1; i <= n; i++) arr.push(i); // 1부터 N까지의 수 삽입
  dfs([], 0);
  console.log();
}
```