JavaScript BFS 알고리즘
BFS 문제 풀이

BFS 문제 풀이 | 코딩 테스트에서 자주 등장하는 BFS 알고리즘 이해하기

강사 나동빈

JavaScript
BFS 알고리즘

BFS 문제 풀이

치즈: https://www.acmicpc.net/problem/2638

혼자 힘으로 풀어보기

문제 제목: 치즈

문제 난이도: ★★★☆☆

문제 유형: BFS

추천 풀이 시간: 60분

치즈: https://www.acmicpc.net/problem/2638

문제 풀이 핵심 아이디어

- 전체 맵은 $N \times M$ 형태를 가진다. (N과 M은 최대 100)
- 한 시간마다 어떤 위치의 치즈를 녹이게 될 지 일일이 계산하면 어떨까?
- BFS로 녹일 위치를 선택하고, **녹이는 과정을 반복 수행**하여 본 문제를 해결할 수 있다.

치즈: https://www.acmicpc.net/problem/2638

문제 풀이 핵심 아이디어

- 어떤 위치의 치즈를 녹여야 할까?
- 치즈 내부의 공간은 치즈 외부 공기와 접촉하지 않는 것으로 간주한다.

  따라서, 단순히 주변이 0인지를 확인하는 방식으로는 해결할 수 없다.
- 즉, 외부 공기를 정확히 파악하기 위해 **(0, 0)의 위치에서 출발**하여 BFS를 진행한다.

  가장자리에는 치즈가 없기 때문이다.

치즈: https://www.acmicpc.net/problem/2638

문제 풀이 핵심 아이디어

[BFS 파트]

- (0, 0)의 위치에서 출발하여 BFS를 진행한다.
- 큐에서 하나의 원소를 꺼낸 뒤에는 상, 하, 좌, 우 위치를 확인한다.

  인접한 위치에 치즈가 있다면, **치즈가 있는 위치에 대하여 카운트(count)**한다.

[녹이기 파트]

- 카운트가 2 이상인 치즈를 없앤다.

치즈: https://www.acmicpc.net/problem/2638

정답 코드 예시


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

// 맵의 크기(N과 M) 정보 입력
let [n, m] = input[0].split(' ').map(Number);
let graph = []; // 2차원 맵 입력받기
for (let i = 1; i <= n; i++) {
  let row = input[i].split(' ').map(Number);
  graph.push(row);
}

// 상, 하, 좌, 우 방향 정보
let dx = [-1, 1, 0, 0];
let dy = [0, 0, -1, 1];
```

치즈: https://www.acmicpc.net/problem/2638

정답 코드 예시

JavaScript
BFS
BFS
문제 풀이

```
function bfs() {
  let visited = []; // 방문 처리 배열
  for (let i = 0; i < n; i++) visited.push(new Array(m).fill(false));
  visited[0][0] = true; // 제일 왼쪽 위에서 출발
  let queue = new Queue(); // 너비 우선 탐색(BFS) 수행
  queue.enqueue([0, 0]);
  while (queue.getLength() != 0) { // 큐가 빌 때까지 반복하기
    let [x, y] = queue.dequeue();
    for (let i = 0; i < 4; i++) {
      let nx = x + dx[i];
      let ny = y + dy[i];
      // 맵을 벗어나는 경우 무시
      if (nx < 0 || nx >= n || ny < 0 || ny >= m) continue;
      if (!visited[nx][ny]) {
        if (graph[nx][ny] >= 1) graph[nx][ny] += 1; // 카운트 증가
        else {
          queue.enqueue([nx, ny]);
          visited[nx][ny] = true;
        }
      }
    }
  }
}
```

치즈: https://www.acmicpc.net/problem/2638

정답 코드 예시


```
function melt() {
  let finish = true; // 더 녹일 치즈가 없는지 여부
  for (let i = 0; i < n; i++) {
    for (let j = 0; j < m; j++) {
      if (graph[i][j] >= 3) { // 녹일 치즈라면
        graph[i][j] = 0; //녹이기
        finish = false;
      }
      else if (graph[i][j] == 2) graph[i][j] = 1; // 한 면만 닿은 경우 무시
    }
  }
  return finish;
}

let result = 0;
while (true) {
  bfs();
  if (melt()) {
    console.log(result); // 전부 녹았다면
    break;
  }
  else result += 1;
}
```

A → B: https://www.acmicpc.net/problem/16953

혼자 힘으로 풀어보기

문제 제목: A → B

문제 난이도: ★★☆☆☆

문제 유형: 너비 우선 탐색, 그래프 순회, 최단 거리

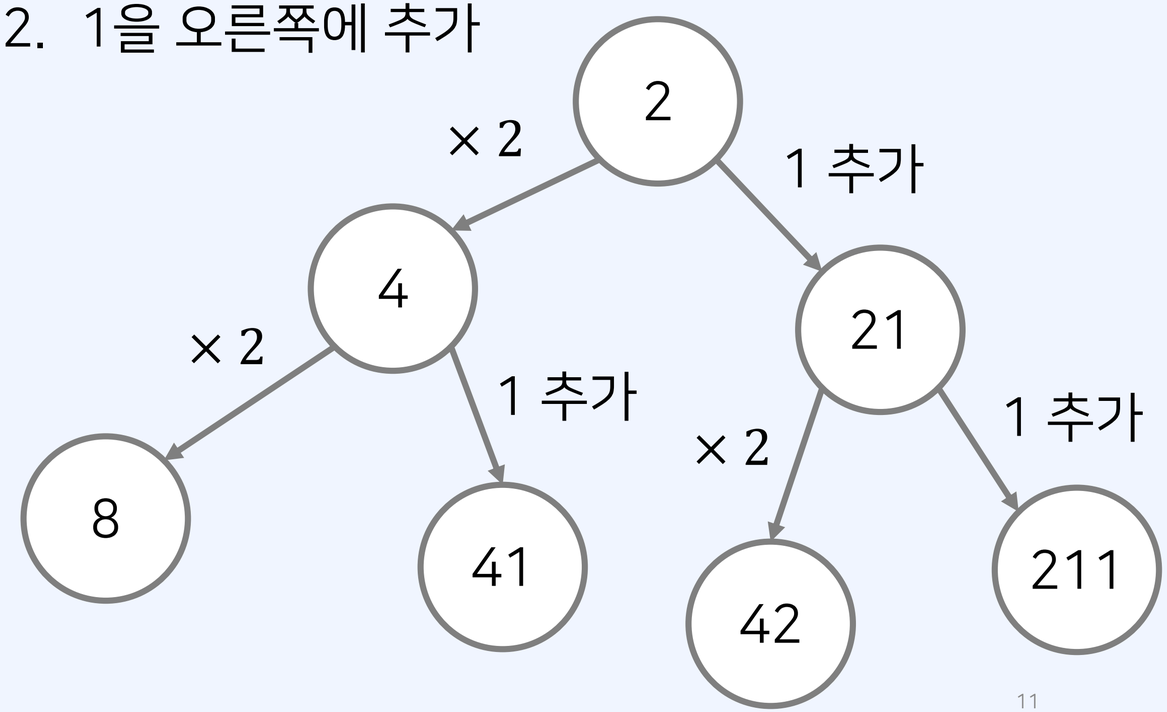
추천 풀이 시간: 40분

A → B: https://www.acmicpc.net/problem/16953

문제 해결 아이디어

- 정수 A를 B로 바꾸려고 한다.
- 사용할 수 있는 연산은 **두 가지 종류**가 있으며, "**최소 연산 횟수**"를 구해야 한다.

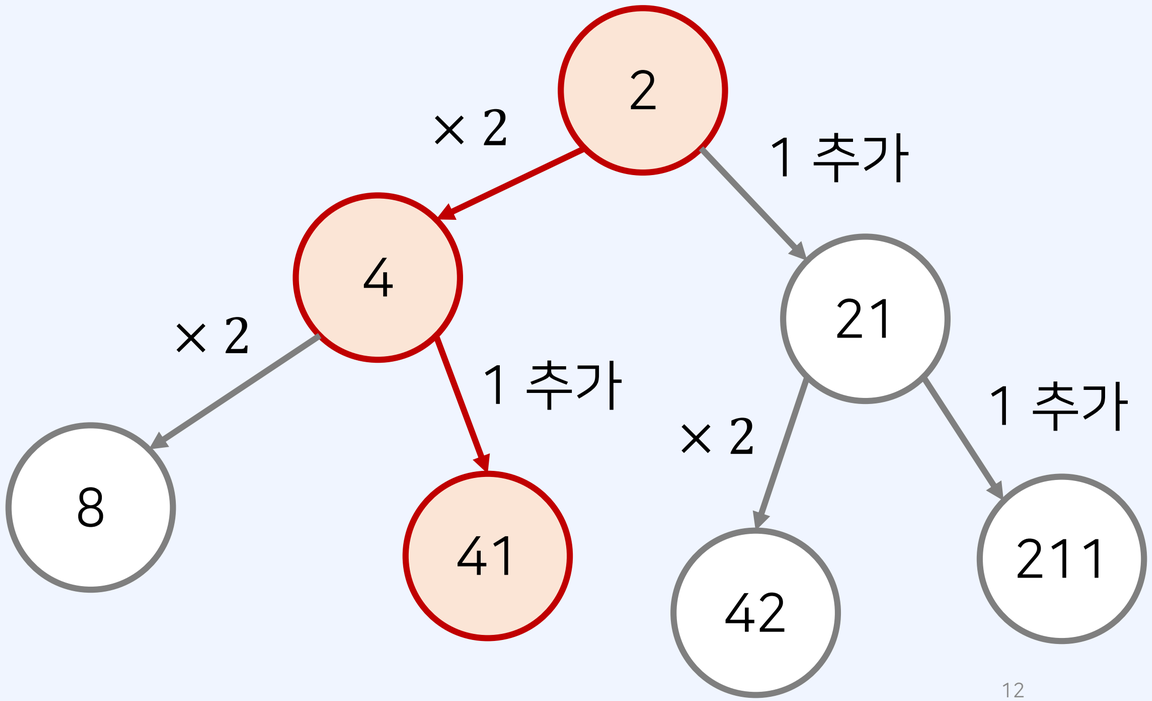
1. 2를 곱하기
2. 1을 오른쪽에 추가



A → B: https://www.acmicpc.net/problem/16953

문제 해결 아이디어

- 본 문제에서는 "필요한 연산의 최솟값"을 구해야 한다.
- 이는 다시 말하면, "최소 거리"를 구하는 문제와 동일하다.
- $A = 2, B = 41$일 때, 최소 거리는 2이다.



A → B: https://www.acmicpc.net/problem/16953

정답 코드 예시


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

// 시작(s)과 목표(t)를 입력받기
let [s, t] = input[0].split(' ').map(Number);
let queue = new Queue(); // 너비 우선 탐색(BFS) 수행
queue.enqueue([s, 0]) // (값, 최소 연산 횟수) 삽입
let visited = new Set();
let found = false;
```

A → B: https://www.acmicpc.net/problem/16953

정답 코드 예시


```
// 큐가 빌 때까지 반복하기
while (queue.getLength() != 0) {
  let [value, dist] = queue.dequeue();
  if (value > 1e9) continue; // 범위를 벗어나는 경우
  if (value == t) { // 목표 값에 도달한 경우
    console.log(dist + 1); // 최소 연산 횟수 + 1 출력
    found = true;
    break;
  }
  for (let oper of ['*', '+']) {
    let nextValue = value;
    if (oper == '*') nextValue *= 2; // 2를 곱하기
    if (oper == '+') { // 1을 오른쪽에 추가
      nextValue *= 10;
      nextValue += 1;
    }
    if (!visited.has(nextValue)) {
      queue.enqueue([nextValue, dist + 1]);
      visited.add(nextValue);
    }
  }
}
if (!found) console.log(-1); // 바꿀 수 없는 경우
```