JavaScript BFS 알고리즘
BFS 문제 풀이

BFS 문제 풀이 | 코딩 테스트에서 자주 등장하는 BFS 알고리즘 이해하기

강사 나동빈

JavaScript
BFS 알고리즘

BFS 문제 풀이

이분 그래프: https://www.acmicpc.net/problem/1707

혼자 힘으로 풀어보기

문제 제목: 이분 그래프

문제 난이도: ★★★☆☆

문제 유형: 너비 우선 탐색, 그래프 순회

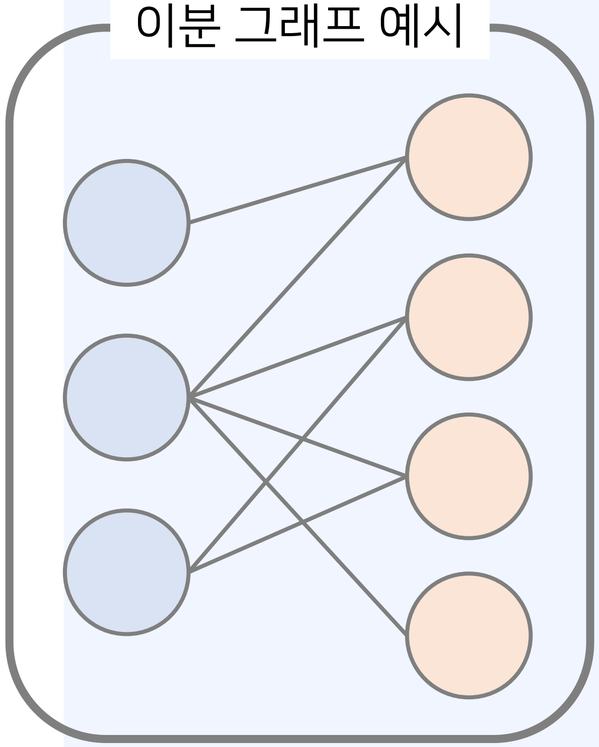
추천 풀이 시간: 60분

이분 그래프: https://www.acmicpc.net/problem/1707

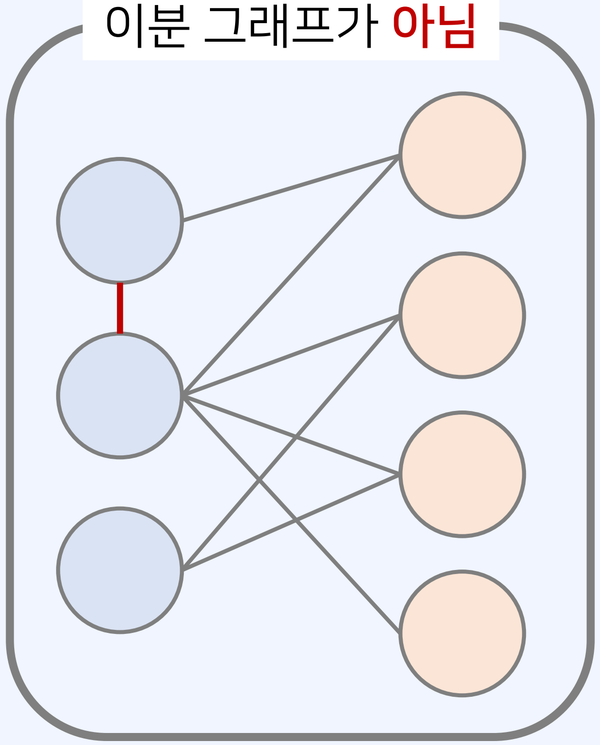
문제 해결 아이디어

- **이분 그래프**(Bipartite Graph)는 노드들을 두 개의 집합으로 분할한 뒤에
- 같은 집합에 속한 정점끼리 서로 인접하지 않는 그래프를 의미한다.

이분 그래프 예시



이분 그래프가 **아님**



1. 아직 방문하지 않은 각 노드에서 시작하여 BFS로 **인접 노드로 이동**하며 **빨** → 파 → **빨** → 파 … 색칠하기
2. 이분 그래프 판별: 모든 노드에 대하여 인접 노드와 색상이 다른지 여부를 판별하기

이분 그래프: https://www.acmicpc.net/problem/1707

정답 코드 예시

JavaScript
BFS
BFS
문제 풀이

```
// Node.js에서는 Stack Size 초과로 DFS로 해결 불가능
// 미방문(color: -1), 빨강(color: 0), 파랑(color: 1)
function bfs(x, graph, visited) {
  queue = new Queue();
  queue.enqueue(x);
  visited[x] = 0; // 처음 노드는 빨간색으로 칠하기
  while (queue.getLength() != 0) {
    x = queue.dequeue();
    for (let y of graph[x]) {
      if (visited[y] == -1) {
        visited[y] = (visited[x] + 1) % 2; // 빨강 ↔ 파랑
        queue.enqueue(y);
      }
    }
  }
}

function isBipartite(graph, visited) {
  for (let x = 1; x < visited.length; x++) {
    for (let y of graph[x]) if (visited[x] == visited[y]) return false;
  }
  return true;
}
```

이분 그래프: https://www.acmicpc.net/problem/1707

정답 코드 예시

JavaScript
BFS
BFS
문제 풀이

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

let testCases = Number(input[0]); // 테스트 케이스의 수
let line = 1;
while (testCases--) {
  // 정점의 개수(V), 간선의 개수(E)
  let [v, e] = input[line].split(' ').map(Number);
  // 그래프 정보 입력받기
  let graph = [];
  for (let i = 1; i <= v; i++) graph[i] = [];
  for (let i = 1; i <= e; i++) {
    let [u, v] = input[line + i].split(' ').map(Number);
    graph[u].push([v]);
    graph[v].push([u]);
  }
  let visited = new Array(v + 1).fill(-1); // 방문 정보
  for (let i = 1; i <= v; i++) { // BFS를 이용해 색칠
    if (visited[i] == -1) bfs(i, graph, visited);
  }
  line += e + 1; // 다음 테스트 케이스로 이동
  if (isBipartite(graph, visited)) console.log("YES");
  else console.log("NO");
}
```

4연산: https://www.acmicpc.net/problem/14395

혼자 힘으로 풀어보기

문제 제목: 4연산

문제 난이도: ★★★☆☆

문제 유형: 너비 우선 탐색, 그래프 순회, 최단 거리

추천 풀이 시간: 60분

4연산: https://www.acmicpc.net/problem/14395

문제 해결 아이디어

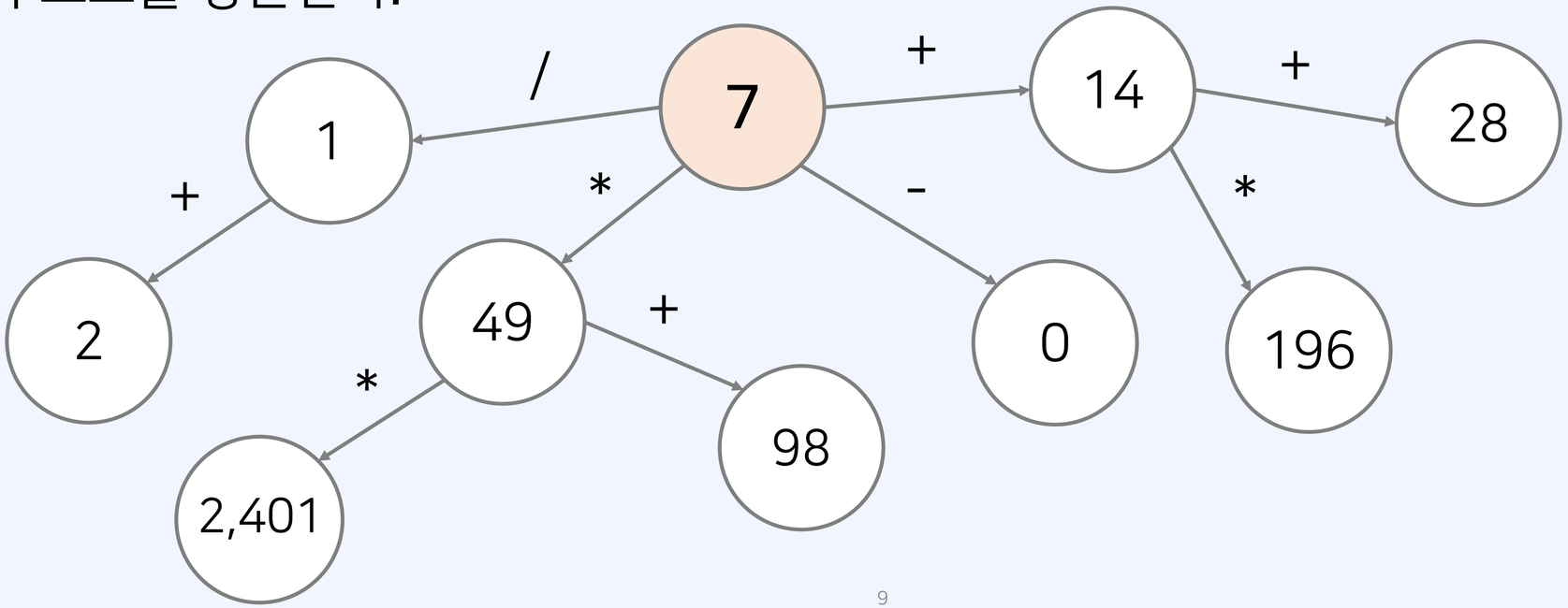
- BFS 문제는 **최단 거리**를 찾거나 **최소 횟수**를 찾는 문제에서 많이 사용된다.
  만약에 그래프 형태로 문제가 표현된다면 **BFS**를 이용해 볼 수 있다.
- 이 문제에서는 주어진 연산이 +, −, *, /이다.
  특정한 정수 s에서 시작해서 "탐색"을 하는 형태로 간주할 수 있다.
- 따라서 값이 t인 노드를 만날 때까지 BFS를 수행한다.

4연산: https://www.acmicpc.net/problem/14395

문제 해결 아이디어

[문제 해결 아이디어]

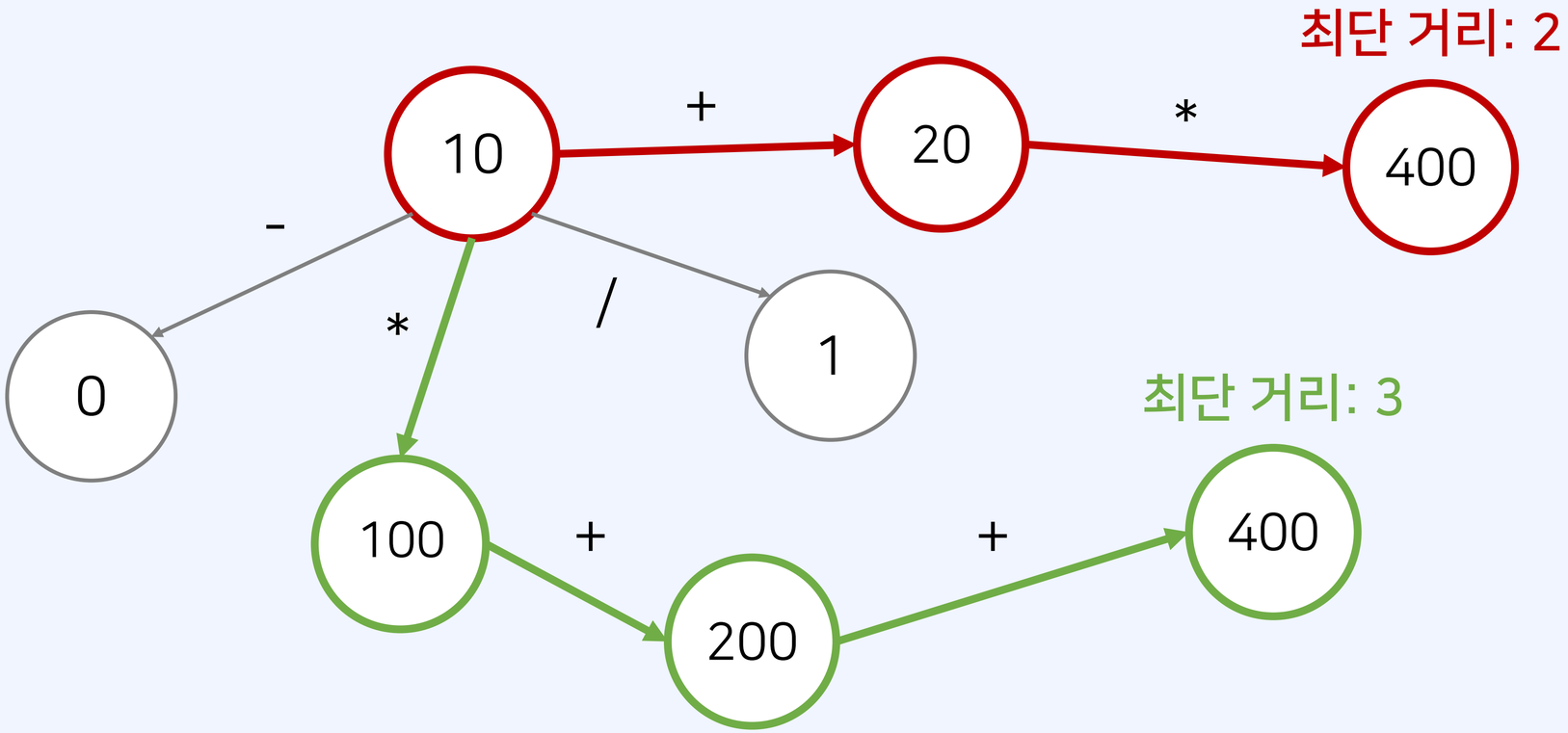
- 최소 연산 횟수를 구하는 문제이므로, **너비 우선 탐색(BFS)**을 사용한다.
- 예를 들어 $s = 7$일 때, **연산 횟수(간선 개수)**를 2까지 고려할 때 다음과 같은 형태로 각 노드를 방문한다.



4연산: https://www.acmicpc.net/problem/14395

문제 해결 아이디어

- $s = 10$에서 $t = 400$으로 가는 예시는 다음과 같다.



4연산: https://www.acmicpc.net/problem/14395

정답 코드 예시

JavaScript
BFS
BFS
문제 풀이

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

// 시작(s)과 목표(t)를 입력받기
let [s, t] = input[0].split(' ').map(Number);

// 너비 우선 탐색(BFS) 수행
let queue = new Queue();
// (값, 해당 값을 만들기 위한 연산자) 삽입
queue.enqueue([s, '']);
let visited = new Set([s]);
let found = false;

if (s == t) { // 시작 값과 목표 값이 같은 경우
  console.log(0);
  process.exit();
}
```

4연산: https://www.acmicpc.net/problem/14395

정답 코드 예시

JavaScript
BFS
BFS
문제 풀이

```
// 큐가 빌 때까지 반복하기
while (queue.getLength() != 0) {
  let [value, opers] = queue.dequeue();
  if (value > 1e9) continue; // 값의 범위를 벗어나는 경우
  if (value == t) { // 목표 값에 도달한 경우
    console.log(opers); // 전체 연산자들을 출력
    found = true;
    break;
  }
  for (let oper of ['*', '+', '-', '/']) { // 각 연산자로 BFS 수행
    let nextValue = value;
    if (oper == '*') nextValue *= value;
    if (oper == '+') nextValue += value;
    if (oper == '-') nextValue -= value;
    if (oper == '/' && value != 0) nextValue = 1;
    if (!visited.has(nextValue)) {
      queue.enqueue([nextValue, opers + oper]);
      visited.add(nextValue);
    }
  }
}
// 바꿀 수 없는 경우
if (!found) console.log(-1);
```