JavaScript BFS 알고리즘
BFS 문제 풀이

BFS 문제 풀이 | 코딩 테스트에서 자주 등장하는 BFS 알고리즘 이해하기

강사 나동빈

JavaScript
BFS 알고리즘

BFS 문제 풀이

환승 : https://www.acmicpc.net/problem/5214

혼자 힘으로 풀어보기

문제 제목: 환승

문제 난이도: ★★★☆☆

문제 유형: 너비 우선 탐색, 최단 거리

추천 풀이 시간: 60분

환승 : https://www.acmicpc.net/problem/5214

문제 해결 아이디어

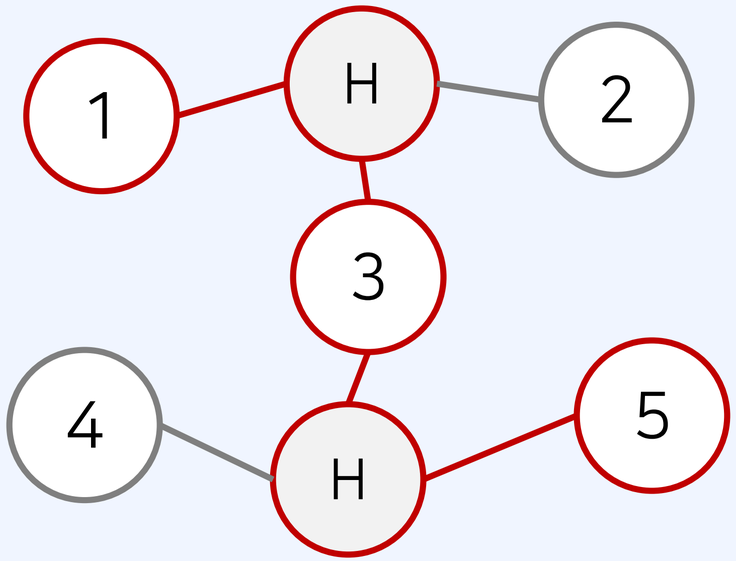
- 하이퍼튜브 하나는 역 K개를 서로 연결한다.
- 1번 역에서 N번 역으로 가는데 방문하는 최소 역의 수를 출력하면 된다.
- 역의 개수 N이 최대 100,000이고, 하이퍼튜브의 개수 M은 최대 1,000이다.
  - 이때, 하이퍼튜브도 역(노드)으로 보고 BFS를 수행할 수 있다.
- 일종의 최단 거리 문제이다.
  - **각 간선의 비용이 모두 동일**하기 때문에, BFS로 최단 거리를 계산할 수 있다.
  - 만약 각 간선의 비용이 다를 수 있다면, 다익스트라와 같은 방법을 이용해야 한다.
- 하이퍼튜브를 통해서만 이동이 가능하므로, 계산된 최단 거리 값을 2로 나누면 그것이 정답이다.
  - 1 → H → 3 → H → 5로 갔다고 하면 거리가 4이므로, 지나야 하는 역의 개수는 2개다.

환승 : https://www.acmicpc.net/problem/5214

문제 해결 아이디어

[문제 해결 아이디어]

- **하이퍼튜브**를 포함하여 전체 노드를 그래프로 구성한다.
- BFS를 이용해서 1번에서 N번까지의 최단 거리를 계산한다.
- $N = 5$일 때의 예시로는 다음과 같은 것이 있다.



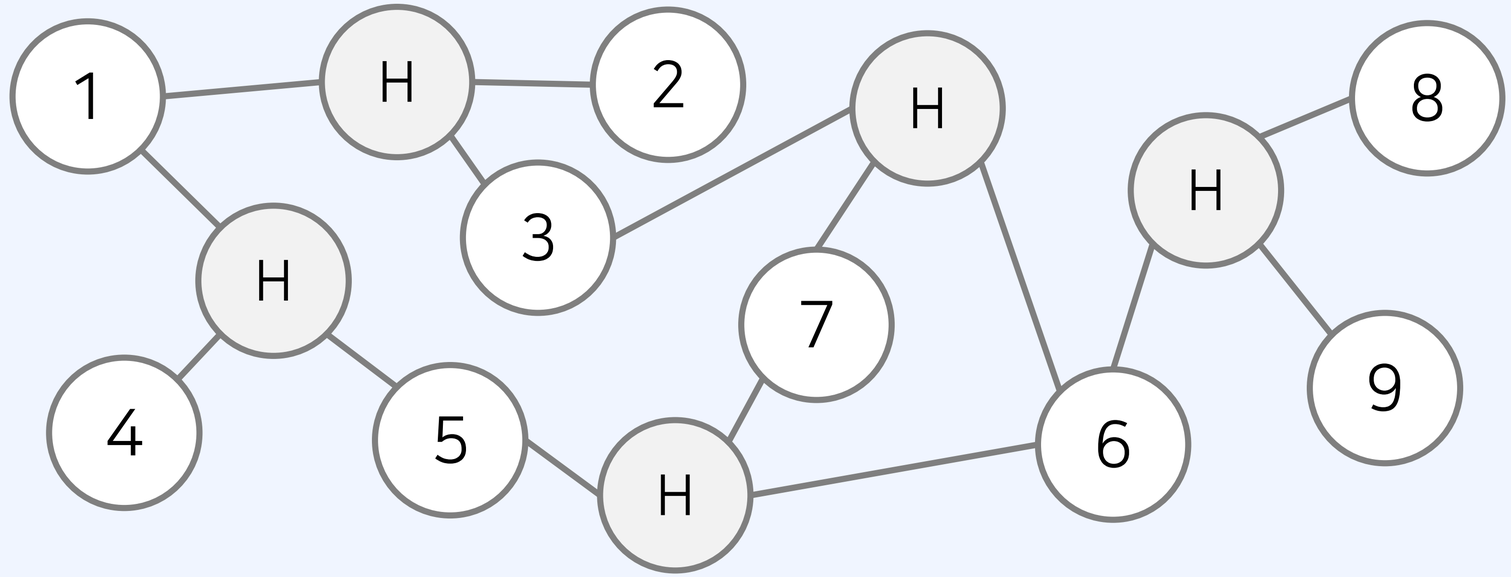
최단 경로: ① → H → ③ → H → ⑤

따라서, **거쳐가는 역의 수**는 3이다.

환승 : https://www.acmicpc.net/problem/5214

문제 해결 아이디어

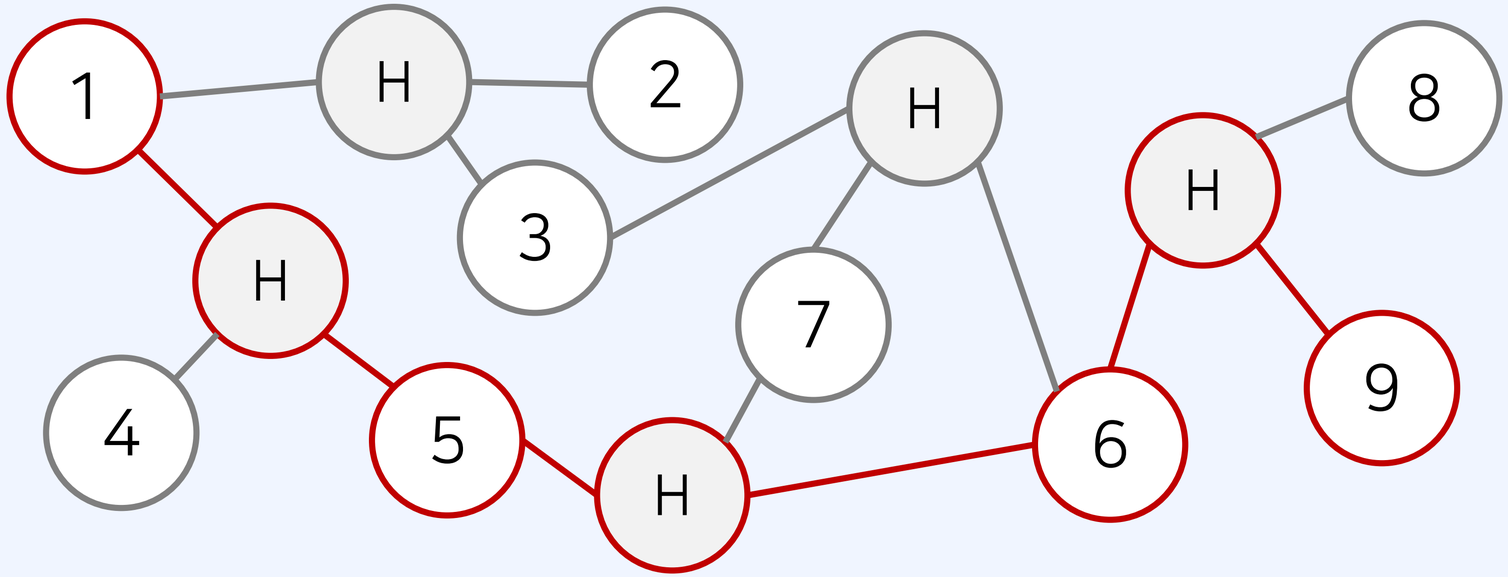
- 1번에서 N번까지 가는데 방문하는 **최소 역의 수**를 구해야 한다.
- [핵심 아이디어] 하이퍼튜브를 노드로 간주하고, **너비 우선 탐색(BFS)**을 사용한다.
- 각 하이퍼튜브와 하이퍼튜브가 연결하는 노드들을 모두 양방향 간선으로 연결한다.



환승 : https://www.acmicpc.net/problem/5214

문제 해결 아이디어

- 1번 노드에서부터 **너비 우선 탐색(BFS)을 수행**한다.
  - N번 노드까지의 거리가 $distance$일 때 정답은 $distance$ // 2 + 1이다.
  - 아래 예시에서 **정답은 4**이다.



환승 : https://www.acmicpc.net/problem/5214

정답 코드 예시


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

// 역의 개수(N), 간선의 개수(K), 하이퍼튜브의 개수(M)
let [n, k, m] = input[0].split(' ').map(Number);
// 그래프 정보(N개의 역과 M개의 하이퍼튜브는 모두 노드)
let graph = [];
for (let i = 1; i <= n + m; i++) graph[i] = [];
for (let i = 1; i <= m; i++) {
  let arr = input[i].split(' ').map(Number);
  for (let x of arr) {
    graph[x].push(n + i); // 노드 → 하이퍼 튜브
    graph[n + i].push(x); // 하이퍼 튜브 → 노드
  }
}
```

환승 : https://www.acmicpc.net/problem/5214

정답 코드 예시

JavaScript
BFS
BFS
문제 풀이

```
let visited = new Set([1]); // 1번 노드에서 출발
let queue = new Queue();
queue.enqueue([1, 1]); // [거리, 노드 번호]
let found = false;

while (queue.getLength() != 0) { // 큐가 빌 때까지 반복하기
  let [dist, now] = queue.dequeue();
  // N번 노드에 도착한 경우
  if (now == n) {
    // 절반은 하이퍼 튜브
    console.log(parseInt(dist / 2) + 1);
    found = true;
    break;
  }
  for (let y of graph[now]) { // 인접 노드를 하나씩 확인
    if (!visited.has(y)) { // 아직 방문하지 않았다면
      queue.enqueue([dist + 1, y]); // 방문 처리
      visited.add(y);
    }
  }
}
if (!found) console.log(-1); // N번 노드에 도달 불가능
```

결혼식: https://www.acmicpc.net/problem/5567

혼자 힘으로 풀어보기

문제 제목: 결혼식

문제 난이도: ★★☆☆☆

문제 유형: 너비 우선 탐색, 그래프 순회, 최단 거리

추천 풀이 시간: 40분

결혼식: https://www.acmicpc.net/problem/5567

문제 해결 아이디어

- 문제의 요구사항은 1번 학생의 "친구"와 "친구의 친구"의 수를 출력하는 것이다.

  [해결 방법] 그래프로 표현한 뒤에 거리가 2 이하인 노드의 수를 출력한다.
- 너비 우선 탐색(BFS)을 사용하면, 각 노드까지의 최단 거리를 구할 수 있다.
- 결과적으로 BFS 호출 이후에 **최단 거리가 2 이하**인 노드의 수를 계산하면 된다.

결혼식: https://www.acmicpc.net/problem/5567

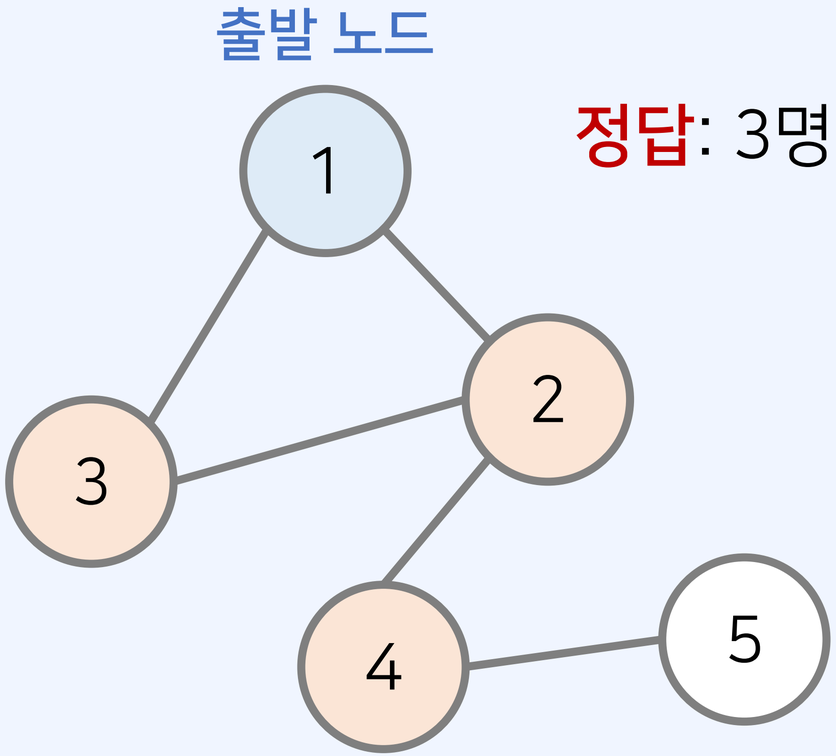
문제 해결 아이디어

- 노드의 개수(N)는 최대 500이다.
- 간선의 개수(M)는 최대 10,000이다.
- 기본적인 BFS를 이용하여 문제를 해결할 수 있다.

결혼식: https://www.acmicpc.net/problem/5567

문제 해결 아이디어

- 문제의 예시를 그래프로 표현하면 다음과 같다.



정답: 3명

결혼식: https://www.acmicpc.net/problem/5567

정답 코드 예시


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

let n = Number(input[0]); // 학생의 수
let m = Number(input[1]); // 친구 관계의 수
// 그래프 정보
let graph = [];
for (let i = 1; i <= n; i++) {
  graph[i] = [];
}
for (let i = 2; i <= m + 1; i++) {
  let [x, y] = input[i].split(' ').map(Number);
  graph[x].push(y);
  graph[y].push(x);
}
// 모든 친구(노드)에 대한 최단 거리 초기화
let distance = new Array(n + 1).fill(-1);
distance[1] = 0; // 시작점까지의 거리는 0으로 설정
```

결혼식: https://www.acmicpc.net/problem/5567

정답 코드 예시


```
let queue = new Queue(); // 너비 우선 탐색(BFS) 수행
queue.enqueue(1);
while (queue.getLength() != 0) { // 큐가 빌 때까지 반복하기
  let now = queue.dequeue();
  for (let nextNode of graph[now]) { // 현재 노드에서 이동할 수 있는 모든 노드를 확인
    if (distance[nextNode] == -1) { // 방문하지 않은 도시라면
      distance[nextNode] = distance[now] + 1;
      queue.enqueue(nextNode);
    }
  }
}
// 최단 거리가 2 이하인 모든 친구(노드)의 수를 계산
let result = 0;
for (let i = 1; i <= n; i++) {
  if (distance[i] != -1 && distance[i] <= 2) {
    result++;
  }
}
console.log(result - 1); // 자기 자신은 제외
```