JavaScript 최단 경로

최단 경로 문제 풀이

최단 경로 문제 풀이 | 코딩 테스트에서 자주 등장하는 최단 경로 이해하기

강사 나동빈

JavaScript 최단 경로

최단 경로 문제 풀이

플로이드: https://www.acmicpc.net/problem/11404

혼자 힘으로 풀어보기

JavaScript
최단 경로
최단 경로
문제 풀이

문제 제목: 플로이드

문제 난이도: ★★★☆☆

문제 유형: 최단 경로, 플로이드 워셜

추천 풀이 시간: 40분

플로이드: https://www.acmicpc.net/problem/11404

문제 풀이 핵심 아이디어

JavaScript
최단 경로
최단 경로
문제 풀이

[문제 설명]

- 도시의 개수(N)가 최대 100개다.
- 간선의 개수(M)가 최대 100,000개다.
- 모든 도시의 쌍(A, B)에 대해 도시 A에서 B로 가는데 필요한 **최소 비용**을 모두 구해라.

플로이드: https://www.acmicpc.net/problem/11404

JavaScript 최단 경로
최단 경로 문제 풀이

문제 풀이 핵심 아이디어

- 모든 도시의 쌍(A, B)에 **대하여 최단 거리를 모두 구해야 한다.**
- **플로이드 워셜(Floyd-Warshall)** 알고리즘을 사용할 수 있다.

  노드의 개수(N)가 최대 100개이므로, 시간 복잡도 $O(N^3)$으로 충분히 해결 가능하다.
- 두 노드 사이의 간선이 여러 개일 수 있으므로, 가장 비용이 적은 간선만 고려한다. **(중복 간선)**

플로이드: https://www.acmicpc.net/problem/11404

JavaScript 최단 경로
최단 경로 문제 풀이

정답 코드 예시

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

let INF = 1e9; // 무한을 의미하는 값으로 10억을 설정
let n = Number(input[0]); // 노드의 개수(N)
let m = Number(input[1]); // 간선의 개수(M)

// graph[i][j]는 i에서 j로 가기 위한 초기 비용(간선 비용)
let graph = [new Array(n + 1).fill(INF)];
for (let i = 1; i <= n; i++) {
  graph.push(new Array(n + 1).fill(INF));
  graph[i][i] = 0; // 자기 자신으로 가는 비용은 0원
}
for (let i = 2; i <= m + 1; i++) {
  let [a, b, c] = input[i].split(' ').map(Number);
  graph[a][b] = Math.min(graph[a][b], c);
}
```

플로이드: https://www.acmicpc.net/problem/11404

정답 코드 예시

```
// 점화식에 따라 플로이드 워셜 알고리즘을 수행
for (let k = 1; k <= n; k++) { // K는 거쳐가는 노드
  for (let a = 1; a <= n; a++) {
    for (let b = 1; b <= n; b++) {
      let cost = graph[a][k] + graph[k][b];
      if (graph[a][b] > cost) { // K를 거쳐갈 때 비용이 더 저렴하다면 테이블 갱신
        graph[a][b] = cost;
      }
    }
  }
}

for (let a = 1; a <= n; a++) { // 수행된 결과를 출력
  let line = '';
  for (let b = 1; b <= n; b++) {
    if (graph[a][b] == INF) line += '0 '; // 도달할 수 없는 경우, "0"이라고 출력
    else line += graph[a][b] + ' '; // 도달할 수 있는 경우 거리를 출력
  }
  console.log(line);
}
```

혼자 힘으로 풀어보기

문제 제목: 친구

문제 난이도: ★★★☆☆

문제 유형: 최단 경로, 플로이드 워셜

추천 풀이 시간: 50분

친구: https://www.acmicpc.net/problem/1058

문제 풀이 핵심 아이디어

JavaScript
최단 경로
최단 경로
문제 풀이

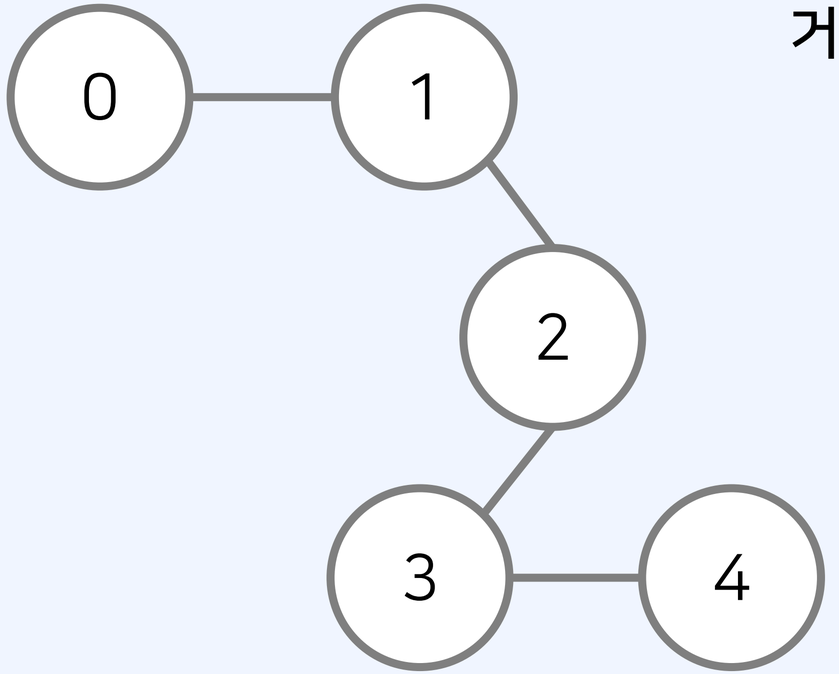
- 거리가 2 이하인 친구의 수를 계산한다.
- 예를 들어 A에서 거리가 2 이하인 예시는 다음과 같다.
- **예시 1)** A - B
- **예시 2)** A - C - B

친구: https://www.acmicpc.net/problem/1058

문제 풀이 핵심 아이디어

- 다음과 같은 입력 예시가 들어온 경우를 생각해 보자.

```
5
NYNNN
YNYNN
NYNYN
NNYNY
NNNYN
```



거리가 2 이하인 친구의 수

- 0번 노드: 2명
- 1번 노드: 3명
- 2번 노드: 4명 → **정답**
- 3번 노드: 3명
- 4번 노드: 2명

친구: https://www.acmicpc.net/problem/1058

정답 코드 예시

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

let INF = 1e9; // 무한을 의미하는 값으로 10억을 설정
let n = Number(input[0]); // 노드의 개수(N)

// graph[i][j]는 i에서 j로 가기 위한 초기 비용(간선 비용)
let graph = [new Array(n + 1).fill(INF)];
for (let i = 1; i <= n; i++) {
  graph.push(new Array(n + 1).fill(INF));
  let line = input[i].split('');
  for (let j = 0; j < n; j++) {
    if (line[j] == 'Y') graph[i][j + 1] = 1;
  }
}
for (let i = 1; i <= n; i++) graph[i][i] = 0; // 자기 자신으로 가는 비용은 0원
```

친구: https://www.acmicpc.net/problem/1058

JavaScript 최단 경로
최단 경로 문제 풀이

정답 코드 예시

```
// 점화식에 따라 플로이드 워셜 알고리즘을 수행
for (let k = 1; k <= n; k++) { // K는 거쳐가는 노드
  for (let a = 1; a <= n; a++) {
    for (let b = 1; b <= n; b++) {
      let cost = graph[a][k] + graph[k][b];
      if (graph[a][b] > cost) { // K를 거쳐갈 때 비용이 더 저렴하다면 테이블 갱신
        graph[a][b] = cost;
      }
    }
  }
}

// 모든 A에서 B로 가는 최단 경로 확인
let twoFriends = new Array(n + 1).fill(0);
for (let a = 1; a <= n; a++) {
  for (let b = 1; b <= n; b++) {
    // 거리가 2 이하인 노드의 수 세기
    if (a != b && graph[a][b] <= 2) twoFriends[a]++;
  }
}
// 거리가 2 이하인 노드의 수 중에서 최댓값을 출력
console.log(twoFriends.reduce((a, b) => Math.max(a, b)));
```

최단 경로: https://www.acmicpc.net/problem/1753

혼자 힘으로 풀어보기

문제 제목: 최단 경로

문제 난이도: ★★★☆☆

문제 유형: 최단 경로, 다익스트라

추천 풀이 시간: 50분

최단 경로: https://www.acmicpc.net/problem/1753

문제 풀이 핵심 아이디어

- 전형적인 방향 그래프에서의 최단 경로 탐색 문제다.
- 하나의 시작점에서 다른 모든 정점까지의 최단 경로를 구해야 한다.

  이때 **노드의 개수(N)**가 최대 20,000개이므로, 다익스트라 최단 경로 알고리즘을 사용한다.
- 우선순위 큐를 활용한 다익스트라 알고리즘으로, $O(ElogV)$의 복잡도를 보장한다.

최단 경로: https://www.acmicpc.net/problem/1753

JavaScript 최단 경로
최단 경로 문제 풀이

정답 코드 예시

```
function dijkstra() { // 다익스트라(Dijkstra) 알고리즘 수행
  let pq = new PriorityQueue((a, b) => b[0] - a[0]); // 최소힙(Min Heap)
  // 시작 노드로 가기 위한 최단 거리는 0으로 우선순위 큐에 삽입
  pq.enq([0, start]);
  distance[start] = 0;
  while (pq.size() != 0) { // 우선순위 큐가 비어있지 않다면
    // 가장 최단 거리가 짧은 노드에 대한 정보 꺼내기
    let [dist, now] = pq.deq();
    // 현재 노드가 이미 처리된 적이 있는 노드라면 무시
    if (distance[now] < dist) continue;
    // 현재 노드와 연결된 다른 인접한 노드들을 확인
    for (let i of graph[now]) {
      let cost = dist + i[1];
      // 현재 노드를 거쳐서, 다른 노드로 이동하는 거리가 더 짧은 경우
      if (cost < distance[i[0]]) {
        distance[i[0]] = cost;
        pq.enq([cost, i[0]]);
      }
    }
  }
}
```

최단 경로: https://www.acmicpc.net/problem/1753

정답 코드 예시

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

let INF = 1e9; // 무한을 의미하는 값으로 10억을 설정
// 노드의 개수, 간선의 개수를 입력받기
let [n, m] = input[0].split(' ').map(Number);
let start = Number(input[1]); // 시작 노드 번호
// 각 노드에 연결되어 있는 노드에 대한 정보를 담는 배열을 만들기
let graph = [];
for (let i = 0; i <= n + 1; i++) graph.push([]);
// 모든 간선 정보를 입력받기
for (let i = 2; i <= m + 1; i++) {
  let [a, b, c] = input[i].split(' ').map(Number);
  // a번 노드에서 b번 노드로 가는 비용이 c라는 의미
  graph[a].push([b, c]);
}
// 최단 거리 테이블을 모두 무한으로 초기화
let distance = new Array(n + 1).fill(INF);
```

최단 경로: https://www.acmicpc.net/problem/1753

JavaScript 최단 경로
최단 경로 문제 풀이

정답 코드 예시

```
// 다익스트라 알고리즘을 수행
dijkstra();
// 모든 노드로 가기 위한 최단 거리를 출력
for (let i = 1; i <= n; i++) {
  // 도달할 수 없는 경우 무한(INF)이라고 출력
  if (distance[i] == INF) console.log('INF');
  // 도달할 수 있는 경우 거리를 출력
  else console.log(distance[i]);
}
```