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

도로포장: https://www.acmicpc.net/problem/1162

JavaScript
최단 경로
최단 경로
문제 풀이

혼자 힘으로 풀어보기

문제 제목: 도로포장

문제 난이도: ★★★★☆

문제 유형: 최단 경로, 다익스트라, 다이나믹 프로그래밍

추천 풀이 시간: 60분

도로포장: https://www.acmicpc.net/problem/1162

문제 풀이 핵심 아이디어

[문제 설명]

- 1번 노드에서 N번 노드까지 도달하기 위한 최단 거리를 구하는 문제다.
- 노드의 개수(N)가 최대 10,000개 이므로, 다익스트라 최단 경로 알고리즘을 사용해야 한다.
- 이때 **K개의 간선의 비용을 0으로 만들 수 있다는 점**이 본 문제의 특징이다.

도로포장: https://www.acmicpc.net/problem/1162

문제 풀이 핵심 아이디어

[문제 해결 아이디어]

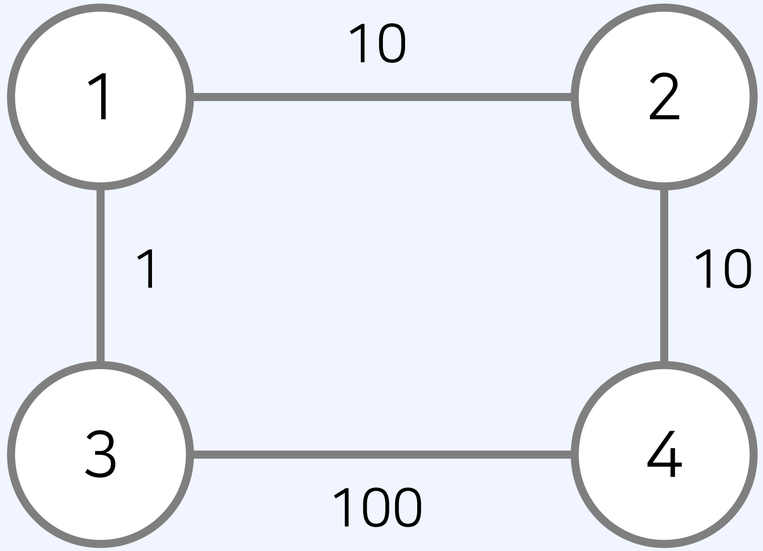
- 본 문제는 **다이나믹 프로그래밍**(dynamic programming)을 사용하여 해결할 수 있다.
- **다익스트라**를 사용하며, 방문하는 각 노드까지의 최단 거리를 DP 테이블에 기록한다.
- 구체적으로 각 노드에 대해 $distance$[노드 번호][현재까지 포장 횟수]를 갱신한다.

  해당 노드를 **포장할지 안 할지 결정**하는 방식이다.

도로포장: https://www.acmicpc.net/problem/1162

문제 풀이 핵심 아이디어

- K개의 간선의 비용을 0으로 만들 수 있다는 점이 본 문제의 특징이다.
- $distance$[노드 번호][현재까지 포장 횟수]를 갱신한다.
- **점화식**: $d[a][k+1] = d[a$의 이전 노드 $p][k]$ ($p \rightarrow$a 비용을 0으로 만든다.)



도로포장: https://www.acmicpc.net/problem/1162

정답 코드 예시

```
function dijkstra(start) { // 다익스트라(Dijkstra) 알고리즘 수행
  let pq = new PriorityQueue((a, b) => b[0] - a[0]); // 최소힙(Min Heap)
  // 시작 노드로 가기 위한 최단 경로는 0으로 설정하여, 큐에 삽입
  pq.enq([0, start, 0]); // (비용, 노드 번호, 포장 횟수)
  distance[start][0] = 0;
  while (pq.size() != 0) { // 우선순위 큐가 비어있지 않다면
    // 가장 최단 거리가 짧은 노드에 대한 정보 꺼내기
    let [dist, now, paved] = pq.deq();
    // 현재 노드가 이미 처리된 적이 있는 노드라면 무시
    if (distance[now][paved] < dist) continue;
    // 현재 노드와 연결된 다른 인접한 노드들을 확인
    for (let i of graph[now]) {
      // 현재 노드를 거쳐서, 다른 노드로 이동하는 거리가 더 짧은 경우
      // 1) 포장하지 않는 경우
      let cost = dist + i[1];
      if (cost < distance[i[0]][paved]) {
        distance[i[0]][paved] = cost;
        pq.enq([cost, i[0], paved]);
      }
      // 2) 포장하는 경우(cost 대신에 dist 사용)
      if (paved < k && dist < distance[i[0]][paved + 1]) {
        distance[i[0]][paved + 1] = dist
        pq.enq([dist, i[0], paved + 1]);
      }
    }
  }
}
```

도로포장: https://www.acmicpc.net/problem/1162

정답 코드 예시

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

let INF = 1e17; // 무한을 의미하는 값으로 10억을 설정
// 노드의 개수(N), 간선의 개수(M), 포장할 간선의 수(K)
let [n, m, k] = input[0].split(' ').map(Number);
// 각 노드에 연결되어 있는 노드에 대한 정보를 담는 배열을 만들기
let graph = [];
for (let i = 0; i <= n + 1; i++) graph.push([]);
for (let i = 1; i <= m; i++) { // 모든 간선 정보를 입력받기
  let [a, b, c] = input[i].split(' ').map(Number);
  graph[a].push([b, c]); // 양방향 간선
  graph[b].push([a, c]);
}
// 최단 거리 테이블을 모두 무한으로 초기화 ([인덱스][포장 횟수])
let distance = [new Array(k + 1).fill(INF)];
for (let i = 1; i <= n; i++) distance[i] = new Array(k + 1).fill(INF);

dijkstra(1); // 다익스트라 알고리즘을 수행
let result = INF; // 노드 N에 도착하기 위한 최소 거리 출력
for (let i = 0; i <= k; i++) {
  result = Math.min(result, distance[n][i]);
}
console.log(result);
```

개코전쟁: https://www.acmicpc.net/problem/2325

JavaScript
최단 경로
최단 경로
문제 풀이

혼자 힘으로 풀어보기

문제 제목: 개코전쟁

문제 난이도: ★★★★☆

문제 유형: 최단 경로, 다익스트라

추천 풀이 시간: 60분

개코전쟁: https://www.acmicpc.net/problem/2325

JavaScript
최단 경로
최단 경로
문제 풀이

문제 풀이 핵심 아이디어

- 본 문제에서는 양방향 그래프가 주어진다.
- **하나의 간선을 제거하여 1번 정점에서 N번 정점으로 가는 최단 거리를 최대화**해야 한다.

  하나의 간선을 제거할 때는 양방향의 길(간선)이 모두 제거되는 것으로 이해할 수 있다.
- 노드의 개수가 1,000개 이상이므로, 본 문제는 다익스트라를 이용해 해결해야 한다.

개코전쟁: https://www.acmicpc.net/problem/2325

문제 풀이 핵심 아이디어

JavaScript
최단 경로
최단 경로
문제 풀이

[문제 해결 아이디어]

- 모든 간선을 하나씩 지우면서 매번 다익스트라를 수행해 보면, 너무 많은 시간이 소요된다.
- **"최단 경로에 포함되지 않은 간선을 제거하면, 최단 거리는 변하지 않는다"**는 사실에 주목하자.
- 따라서 모든 최단 경로를 구한 뒤에, 최단 경로들에 포함된 간선들을 하나씩 제거해 보자.

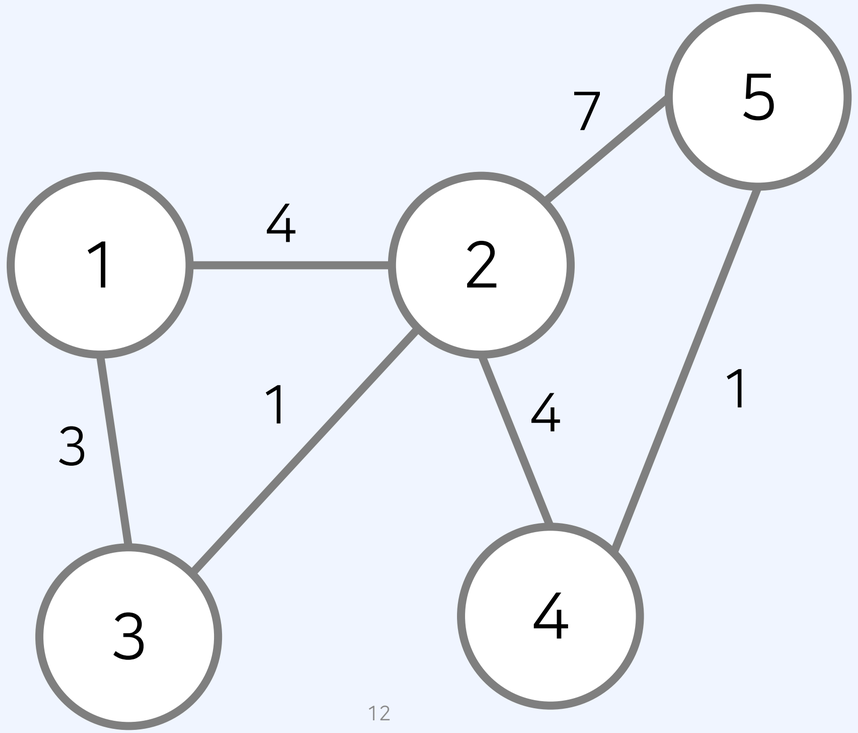
  하나씩 지워보면서, 다시 다익스트라를 호출하여 문제를 해결할 수 있다.

개코전쟁: https://www.acmicpc.net/problem/2325

문제 풀이 핵심 아이디어

- 다음과 같은 입력 예시가 들어온 경우를 생각해 보자.
- **최단 경로**: ① → ② → ④ → ⑤ (**최단 거리**: 9)
- ② → ④ 혹은 ④ → ①을 제거하면, 최단 거리가 11이 된다.

```
5 6
1 2 4
1 3 3
2 3 1
2 4 4
2 5 7
4 5 1
```



개코전쟁: https://www.acmicpc.net/problem/2325

JavaScript 최단 경로
최단 경로 문제 풀이

문제 풀이 핵심 아이디어

[문제 해결 방법]

1. 먼저 다익스트라 알고리즘을 이용해 최단 경로를 한 번 계산한다.
2. 이후에 BFS를 이용해 "최단 경로들"에 포함된 모든 간선을 찾는다.

   참고로 최단 경로는 하나가 아닐 수 있으며, 모든 간선을 다 찾아야 한다.
3. 결과적으로 그러한 간선들을 하나씩 지우면서, 매번 다익스트라를 호출한다.

[참고]

- 기존의 다익스트라 함수를 전혀 변경하지 않고, 그대로 사용할 수 있다.

  다만, 경로 추적을 위한 $BFS()$ 함수를 구현해야 한다.

개코전쟁: https://www.acmicpc.net/problem/2325

정답 코드 예시

```
// 일반적인 다익스트라와 동일하지만, a ↔ b 간선은 무시하는 함수
function dijkstra(a, b) {
  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]) {
      // a ↔ b 간선은 무시
      if (i[0] == a && now == b) continue;
      else if (i[0] == b && now == a) continue;
      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]]);
      }
    }
  }
}
```

개코전쟁: https://www.acmicpc.net/problem/2325

JavaScript 최단 경로
최단 경로 문제 풀이

정답 코드 예시

```
// 최단 경로 역추적 함수
function bfs() {
  let queue = new Queue();
  let visited = new Set(); // 특정한 노드 방문 여부
  queue.enqueue(end); // 도착 지점(end)을 큐에 삽입
  let removes = []; // 삭제할 간선들(결과)
  while (queue.getLength() != 0) { // 큐가 빌 때까지 반복하기
    let now = queue.dequeue();
    if (now == start) { // 시작점에 도착한 경우
      continue; // 모든 최단 경로를 확인하기 위해 break 대신 continue
    }
    for (let i of graph[now]) { // 현재 노드와 연결된 간선들 확인
      let cost = distance[i[0]] + i[1];
      // 최단 경로에 포함된 간선인 경우 삭제 목록에 추가
      if (cost == distance[now]) {
        removes.push([i[0], now]);
        // 각 "직전 노드"는 한 번씩만 방문
        if (!visited.has(i[0])) {
          queue.enqueue(i[0]);
          visited.add(i[0]);
        }
      }
    }
  }
  return removes;
}
```

개코전쟁: https://www.acmicpc.net/problem/2325

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);
// 시작 노드와 도착 노드
let [start, end] = [1, n];
// 각 노드에 연결되어 있는 노드에 대한 정보를 담는 배열을 만들기
graph = [];
for (let i = 0; i <= n + 1; i++) graph.push([]);

// 모든 간선 정보를 입력받기
for (let i = 1; i <= m; i++) {
  let [a, b, c] = input[i].split(' ').map(Number);
  graph[a].push([b, c]);
  graph[b].push([a, c]);
}
// 최단 거리 테이블을 모두 무한으로 초기화
let distance = new Array(n + 1).fill(INF);
// 다익스트라 알고리즘을 수행
dijkstra(-1, -1);
```

개코전쟁: https://www.acmicpc.net/problem/2325

정답 코드 예시

```
// 최단 경로 역추적: 모든 최단 경로에 포함된 간선 쌍 (a, b)들을 계산
let removes = bfs();

let result = 0;
// 모든 최단 경로에 포함된 간선 쌍 (a, b)들을 확인
for ([a, b] of removes) {
  // 최단 거리 테이블을 모두 무한으로 초기화
  distance = new Array(n + 1).fill(INF);
  // a ↔ b 간선은 무시하는 다익스트라 알고리즘을 수행
  dijkstra(a, b);
  result = Math.max(result, distance[end])
}
console.log(result);
```