Shortest Precedent Chain in Indian Case Law

1 Problem Statement

You are building a legal research tool for Indian courts. When a lawyer cites a Supreme Court or High Court judgment, they often need to trace how one case references another through a chain of legal precedents.

Given a citation graph where each edge represents "Case X cites Case Y", find the shortest chain of citations from a starting case to a target case. If no citation path exists between the two cases, return an empty list.

Context: In Indian legal practice, lawyers often need to trace precedent chains. For example, if a recent judgment cites an older landmark case indirectly through intermediate cases, you need to find the shortest such chain to understand the legal lineage.

1.1 Input Format

```
{
  "citations": [
    {"from": "Kesavananda_Bharati_1973", "to": "Minerva_Mills_1980"},
    {"from": "Minerva_Mills_1980", "to": "Waman_Rao_1981"},
    {"from": "Kesavananda_Bharati_1973", "to": "IC_Golaknath_1967"}
  ],
  "start": "Kesavananda_Bharati_1973",
  "target": "Waman_Rao_1981"
}
```

Each citation object represents "Case X cites Case Y". Case identifiers are strings (could be case names, citations like "AIR 1973 SC 1461", or case IDs).

1.2 Output Format

```
["Kesavananda_Bharati_1973", "Minerva_Mills_1980", "Waman_Rao_1981"]
```

Return an array of case identifiers representing the shortest citation path from start to target. If no path exists, return an empty array `[]`.

2 Example Test Cases

2.1 Test Case 1: Direct Citation

- **Input:**
  - citations: `["from": "Maneka_Gandhi_1978", "to": "Gopalan_1950"]`

  - start: `"Maneka_Gandhi_1978"`
  - target: `"Gopalan_1950"`
- **Output:** `["Maneka_Gandhi_1978", "Gopalan_1950"]`
- **Explanation:** Maneka Gandhi case directly cites Gopalan case

2.2 Test Case 2: Multi-hop Citation Chain

- **Input:**
  - citations: `["from": "State_of_Maharashtra_1962", "to": "Bombay_1951", {"from": "Bombay_1951", "to": "Privy_Council_1947"}]`
  - start: `"State_of_Maharashtra_1962"`
  - target: `"Privy_Council_1947"`
- **Output:** `["State_of_Maharashtra_1962", "Bombay_1951", "Privy_Council_1947"]`
- **Explanation:** Maharashtra case cites Bombay case, which cites Privy Council case

2.3 Test Case 3: No Citation Path

- **Input:**
  - citations: `["from": "A_K_Gopalan_1950", "to": "Bank_Nationalization_1970"]`
  - start: `"A_K_Gopalan_1950"`
  - target: `"Unknown_Case_1999"`
- **Output:** `[]`
- **Explanation:** No citation chain connects these cases

2.4 Test Case 4: Same Case

- **Input:**
  - citations: `[]`
  - start: `"Case_ABC"`
  - target: `"Case_ABC"`
- **Output:** `["Case_ABC"]`
- **Explanation:** Start and target are the same case

2.5 Test Case 5: Multiple Possible Paths

- **Input:**
  - citations: `["from": "A", "to": "B", {"from": "A", "to": "C"}, {"from": "B", "to": "D"}, {"from": "C", "to": "D"}]`
  - start: `"A"`
  - target: `"D"`
- **Output:** `["A", "B", "D"]` or `["A", "C", "D"]` (both are shortest, pick one)
- **Explanation:** Two paths of equal length exist

3 Edge Cases

- Start case equals target case → return `[start]`
- No citation path exists → return `[]`
- Disconnected citation graph → return `[]`
- Empty citations list → return `[]` (unless start == target)
- Cases form citation cycles → should still find shortest path if one exists

4 Requirements

- Find the shortest citation chain (minimum number of intermediate cases)
- Handle cycles in the citation graph (cases may cite each other)
- Return exact path (list of case identifiers), not just length
- If multiple shortest paths exist, return any one of them