Bài 10.5: Thuật toán Bellman-Ford

- ✓ Khái niệm và đặc điểm
- ✓ Mã giả và triển khai
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Khái niệm và đặc điểm

- Bellman-Ford là thuật toán tìm đường đi ngắn nhất giữa đỉnh bắt đầu đến tất cả các đỉnh còn lại của đồ thị có cạnh chứa trọng số không có chu trình âm.
- Trong trường hợp đồ thị có chu trình âm, bài toán sẽ không có lời giải.
- Thuật toán này chậm hơn thuật toán Dijkstra khi giải quyết cùng vấn đề.
- Thuật toán này có thể tìm được lời giải ngay cả khi đồ thị có cạnh mang trọng số âm.
- Thuật toán Dijkstra chỉ xét các đỉnh kề trong mỗi lần lặp còn thuật toán Bellman-Ford xét từng đỉnh thông qua từng cạnh trong mọi lần lặp.
- Thuật toán có độ phức tạp O(|V| * |E|). Với V, E lần lượt là số đỉnh và số cạnh tương ứng.

Khái niệm và đặc điểm

- Thuật toán này sử dụng chủ yếu trong định tuyến cổng internet.
- Sử dụng để định tuyến khoảng cách giúp định tuyến các gói tin trên mạng.
- Sử dụng trong giao thức định tuyến thông tin.

Mã giả

➢ Sau đây là mã giả thuật toán Bellman-Ford:

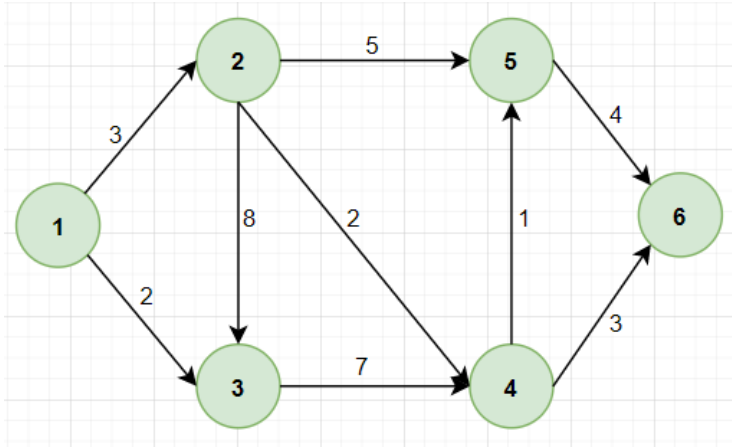
```
// thuật toán Bellman-Ford
// G: đồ thị đang xét
// source: đỉnh bắt đầu duyệt
// distance: danh sách chứa khoảng cách
// prev: danh sách lưu vết đường đi đến đỉnh đích
function BellmanFord(G, source):
    // B1: khởi tạo
    for(từng đỉnh v trong tập đỉnh của đồ thị):
        distance[v] = vô cực // khởi tạo khoảng cách
        prev[v] = null // khởi tạo đỉnh trước v là null
    distance[source] = 0 // gán khoảng cách của đỉnh source
    // B2: lặp |V| - 1 lần
    k = 0
    while(k nhỏ hơn V - 1): // V là số đỉnh của đồ thị
        for(từng cạnh (u, v) với trọng số w trong tập cạnh):
            if(distance[u] + w < distance[v]):
                distance[v] = distance[u] + w
                prev[v] = u
        k++
    // B3: kiểm chu trình âm
    for(từng cạnh (u, v) với trọng số w trong tập cạnh):
        if(distance[u] + w < distance[v]):
            thông báo đồ thị có chu trình âm
            return null
    return distance[], prev[]
```

Triển khai

➢ Triển khai thuật toán Bellman-Ford:

```
public static Vertex[] bellmanFord(Vertex[] vertices, List<Edge> edges, int source) {
    // bước 1: khởi tạo
    Vertex[] prev = new Vertex[vertices.length]; // mảng chứa lưu vết đường đi
    for (Vertex vertex : vertices) {
        vertex.weight = Integer.MAX_VALUE;
    }
    vertices[source].weight = 0;
    // bước 2: lặp |V| - 1 lần
    var k = 0;
    while (k < vertices.length - 1) {
        for (Edge e : edges) {
            var alt = vertices[e.start].weight + e.weight;
            if (alt < vertices[e.end].weight) {
                vertices[e.end].weight = alt;
                prev[e.end] = vertices[e.start];
            }
        }
        k++;
    }
    // bước 3: kiểm tra chu trình âm
    for (Edge e : edges) {
        var alt = vertices[e.start].weight + e.weight;
        if (alt < vertices[e.end].weight) {
            System.out.println("Đồ thị có chu trình âm");
            return null;
        }
    }
    return prev; // trả về danh sách chứa đường đi của thuật toán
}
```

Ví dụ



- Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 6: 1 -> 2 -> 4 -> 6.
- Độ dài hành trình: 8.

Nội dung tiếp theo

Thuật toán Floyd-Warshall