Bài 11.6: Thuật toán Floyd-Warshall

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

Khái niệm và đặc điểm

- Thuật toán Floyed-Warshall dùng để tìm đường đi ngắn nhất trong đồ thị có hướng với cạnh có trọng số âm hoặc dương. Đồ thị phải đảm bảo không chứa chu trình âm.
- Mỗi lần thực hiện thuật toán sẽ tìm độ dài của các đường đi ngắn nhất giữa tất cả các cặp đỉnh.
- Floyd-Warshall là thuật toán thuộc loại quy hoạch động được công bố vào năm 1962.
- Thuật toán có độ phức tạp O($|V|^3$). Với V là số đỉnh của đồ thị.
- Thuật toán này được sử dụng trong trường hợp tổng quát để tìm đường đi ngắn nhất giữa các cặp đỉnh của đồ thị.

Ứng dụng

- Tìm đường đi ngắn nhất trên đồ thị.
- Tìm sự giống nhau giữa các đồ thị.
- Định tuyến tối ưu.
- Nghịch đảo ma trận thực.

Mã giả

```
// thuật toán Floyd-Warshall
// edges: tập cạnh
// dist: mảng chứa khoảng cách cấp |V| x |V|
// next: mảng chứa đường đi cấp |V| x |V|
// size: số đỉnh của đồ thị
function FloydWarshall(edges[], dist[][], next[][], size):
    // B1: khởi tạo ma trận khoảng cách, ma trận next
    for(i từ 0 đến |V| - 1):
        for(j từ 0 đến |V| - 1):
            if(i != j):
                dist[i][j] = dương vô cực
            else:
                dist[i][j] = 0
                next[i][j] = i
    for(từng cạnh (u, v) trong tập cạnh với trọng số w):
        dist[u][v] = w
        next[u][v] = v
    // B2: lặp
    for(k từ 0 đến |V| - 1):
        for(i từ 0 đến |V| - 1):
            for(j từ 0 đến |V| - 1):
                alt = dist[i][k] + dist[k][j]
                if(dist[i][j] > alt):
                    dist[i][j] = alt
                    next[i][j] = next[i][k]
```

Triển khai

```
// thuật toán floydWarshall
void floydWarshall(vector<Edge> edges, int** dist, int** next, int size) {
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++)
        {
            if (i != j) {
                dist[i][j] = SHRT_MAX; // giả định giá trị vô cực
                next[i][j] = -1;
            }
            else {
                dist[i][j] = 0;
                next[i][j] = i;
            }
        }
    }
    for (Edge e : edges) {
        dist[e.start][e.end] = e.weight;
        next[e.start][e.end] = e.end;
    }
    for (int k = 0; k < size; k++) {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                int alt = dist[i][k] + dist[k][j];
                if (dist[i][j] > alt) {
                    dist[i][j] = alt;
                    next[i][j] = next[i][k];
                }
            }
        }
    }
}
```

Mã giả

- Sau đây là mã giả tìm vết đường đi:

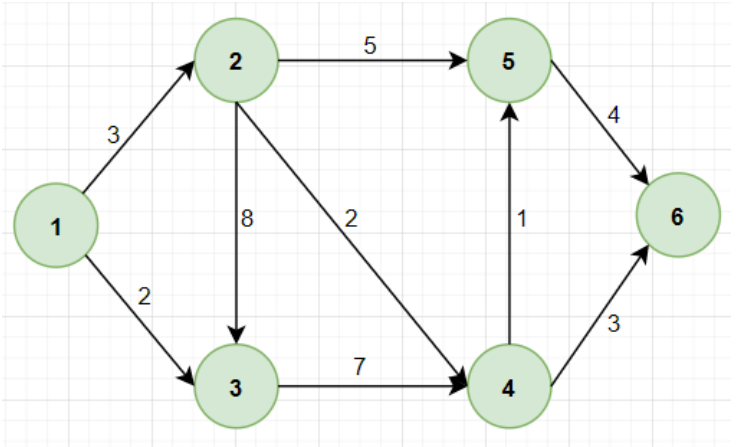
```
// thủ tục lấy đường đi từ đỉnh u tới đỉnh v
// next: mảng chứa đường đi đến tất cả các đỉnh
// path: mảng chứa đường đi từ đỉnh u->v
// u, v: đỉnh đầu, cuối cần tìm đường đi
function minPath(next[][], path[], u, v):
    if(next[u][v] = null):
        return []
    path.append(u)
    while(u != v):
        u = next[u][v]
        path.append(u)
    return path
```

Triển khai

- Triển khai tìm vết đường đi:

```
void minPath(int** next, vector<int>& path, int u, int v) {
    if (next[u][v] == -1) {
        return;
    }
    path.push_back(u);
    while (u != v) {
        u = next[u][v];
        path.push_back(u);
    }
}
```

Ví dụ



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

The End

Cảm ơn các bạn đã đồng hành cùng Branium Academy!