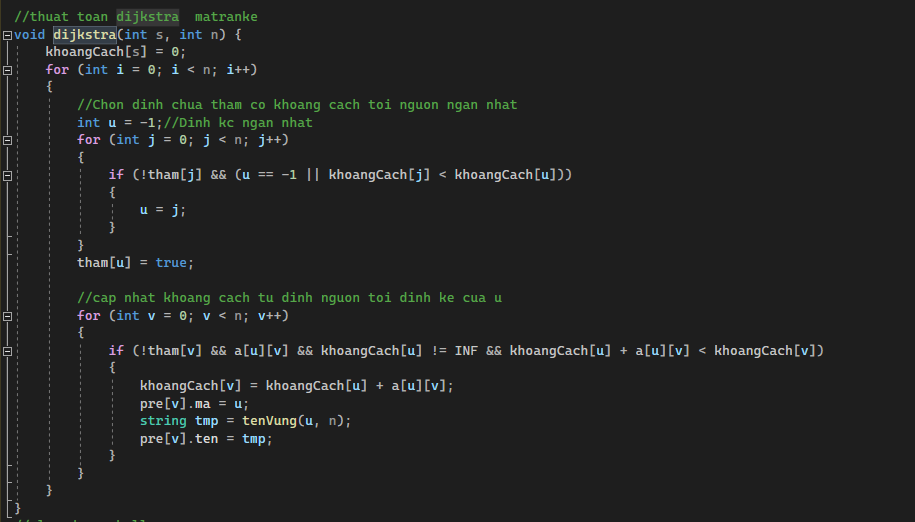
Thuật toán Dijkstra:

Thuật toán Dijkstra là một thuật toán quyết định ngắn nhất (shortest path) được sử dụng để tìm đường đi ngắn nhất giữa hai đỉnh trên một đồ thị có trọng số không âm. Được đặt tên theo nhà toán học và nhà máy tính người Hà Lan Edsger W. Dijkstra, thuật toán này thường được áp dụng trong các ứng dụng mạng và địa lý.

Mục tiêu của thuật toán Dijkstra là tính toán và duy trì một bảng điều khiển (distance table) chứa các giá trị ngắn nhất từ đỉnh nguồn đến tất cả các đỉnh khác trên đồ thị. Khi tất cả các đỉnh đã được xét, bảng này cung cấp thông tin về đường đi ngắn nhất từ đỉnh nguồn đến mọi đỉnh khác trên đồ thị.



1 Khởi tạo Khoảng cách và Đỉnh đã thăm:

khoangCach[s] = 0;

Khởi tạo khoảng cách từ đỉnh nguồn s đến chính nó là 0.

for (int i = 0; i < n; i++) { tham[i] = false; }

Khởi tạo mảng tham để theo dõi việc thăm các đỉnh. Ban đầu, không có đỉnh nào được thăm.

2 Lặp qua tất cả các đỉnh:

for (int i = 0; i < n; i++)

Mỗi vòng lặp của i chọn một đỉnh để thăm tiếp theo.

3 Chọn đỉnh chưa thăm có khoảng cách ngắn nhất:

int u = -1; for (int j = 0; j < n; j++) { if (!tham[j] && (u == -1 || khoangCach[j] < khoangCach[u])) { u = j; } }

Trong mỗi vòng lặp j, tìm một đỉnh chưa thăm (!tham[j]) mà có khoảng cách ngắn nhất từ đỉnh nguồn. Đỉnh này được chọn và đánh dấu là đã thăm (tham[u] = true).

4 Cập nhật khoảng cách từ đỉnh nguồn đến các đỉnh kề của u:

for (int v = 0; v < n; v++) { if (!tham[v] && a[u][v] && khoangCach[u] != INF && khoangCach[u] + a[u][v] < khoangCach[v]) { khoangCach[v] = khoangCach[u] + a[u][v]; pre[v].ma = u; string tmp = tenVung(u, n); pre[v].ten = tmp; } }

Với mỗi đỉnh kề của u (a[u][v] != 0), nếu đỉnh v chưa thăm và có một đường đi ngắn hơn từ đỉnh nguồn đến v thông qua u, thì cập nhật khoảng cách và thông tin liên quan.

5 Lặp lại bước 3 và 4 cho đến khi tất cả các đỉnh đều đã thăm:

Khi tất cả các đỉnh đều đã thăm, thuật toán kết thúc.