* **Thuật toán dijkstra**

# Lý thuyết

Thuật toán Dijkstra có thể giải quyết bài toán tìm đường đi ngắn nhất trên đồ thị vô hướng lẫn có hướng miễn là 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.

Độ phức tạp: O((E+V) logV)

Ý tưởng cơ bản của thuật toán như sau:

*Bước 1:* Từ đỉnh gốc, khởi tạo khoảng cách tới chính nó là (0,0), khởi tạo khoảng cách nhỏ nhất ban đầu tới các đỉnh khác là (-,∞). Ta được danh sách các khoảng cách tới các đỉnh.

*Bước 2:* Chọn đỉnh a có khoảng cách nhỏ nhất trong danh sách này và ghi nhận. Các lần sau sẽ không xét tới đỉnh này nữa.

*Bước* *3:* Lần lượt xét các đỉnh kề b của đỉnh a. Nếu khoảng cách từ đỉnh gốc tới đỉnh b nhỏ hơn khoảng cách hiện tại đang được ghi nhận thì cập nhật giá trị và đỉnh kề a vào khoảng cách hiện tại của b.

*Bước 4:* Sau khi xét tất cả đỉnh kề b của đỉnh a. Lúc này ta được danh sách khoảng cách tới các điểm đã được cập nhật. Quay lại Bước 2 với danh sách này. Thuật toán kết thúc khi chọn được khoảng cách nhỏ nhất từ tất cả các điểm.

**Code mẫu thuật toán dijkstra**

Sử dụng mảng 2 chiều để lưu đường đi trọng số của bài toán.

1. Đặt khoảng cách nguồn là 0

2. Duyệt qua tất cả các đỉnh của đồ thị

Khởi tạo biến u để lưu đỉnh có khoảng cách ngắn nhất

Vòng lặp chọn đỉnh chưa thăm có khoảng cách tới nguồn ngắn nhất

Điều kiện: Đỉnh thứ j chưa thăm và khoảng cách ngắn hơn đỉnh u hoặc u chưa được gán giá trị

Gán u = j

Đánh dấu đỉnh u đã thăm

Vòng lặp cập nhật khoảng cách từ đỉnh nguồn tới đỉnh kề của u

Điều kiện: “Đỉnh v chưa xét, nối với u, khoảng cách từ s tới u không phải là vô cùng, và khoảng cách mới từ s tới v ngắn hơn khoảng cách hiện tại”

Cập nhật khoảng cách từ nguồn tới đỉnh v bằng cách lấy khoảng cách của u cộng với trọng số của cạnh uv.

void dijkstra(int s, int n) {

khoangCach[s] = 0;

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

{

//Chon dinh chua tham co khoang cach toi nguon ngan nhat

int u = -1;//Dinh kc ngan nhat

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

{

if (!tham[j] && (u == -1 || khoangCach[j] < khoangCach[u]))

{

u = j;

}

}

tham[u] = true;

//cap nhat khoang cach tu dinh nguon toi dinh ke cua 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] = u;

}

}

}

}

# Ứng dụng

Tìm đường đi ngắn nhắt từ …. đến …. (tên tỉnh thành)

Bắt các chuyến xe đi gần nhất

Trọng số 10km = 1

Mã vùng

|  |  |
| --- | --- |
| 0 : Sài Gòn | 1 : Vũng Tàu |
| 2 : Cần Giờ | 3 : Tiền Giang |
| 4 : Bến Tre | 5 : Tây Ninh |
| 6 : Sóc Trăng |  |

Table : Danh sách cạnh

|  |  |  |
| --- | --- | --- |
| Đầu | Cuối | Trọng số |
| 0 | 1 | 10 |
| 0 | 3 | 6 |
| 0 | 2 | 5 |
| 0 | 6 | 24 |
| 1 | 5 | 18 |
| 1 | 2 | 1 |
| 3 | 4 | 2 |
| 3 | 6 | 16 |

Table : Đồ thị của Ứng dụng



Mục tiêu: Cho người dùng nhập điểm đi và điểm đến chạy thuật toán để tìm đường ngắn nhất đến các điểm đó.

void dijkstra(int s, int n) {

khoangCach[s] = 0;

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

{

//Chon dinh chua tham co khoang cach toi nguon ngan nhat

int u = -1;//Dinh kc ngan nhat

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

{

if (!tham[j] && (u == -1 || khoangCach[j] < khoangCach[u]))

{

u = j;

}

}

tham[u] = true;

//cap nhat khoang cach tu dinh nguon toi dinh ke cua 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;

}

}

}

}

//in đường đi

void printPathUngDung(int s, int dest) {

if (khoangCach[dest] == INF) {

cout << "Khong co duong nao tu " << maVungArr[s].ten << " den " << maVungArr[dest].ten << endl;

return;

}

cout << "Duong ngan nhat tu " << maVungArr[s].ten << " den " << maVungArr[dest].ten << ": ";

ma\_Vung current = maVungArr[dest];

string tmp[Nmax]; int i = 0;

string den = current.ten;

while (pre[current.ma].ma != INF) {

tmp[i] = pre[current.ma].ten;

current = pre[current.ma];

i++;

}

while (i != 0)

{

cout << tmp[i - 1] << " -> ";

i--;

}

cout << den;

cout << endl;

}