**Báo Cáo Thực Hành**

**Bùi Thị Thanh Phương**

**20110280**

I.Nội Dung Thực Hành

# 1. Bài toán tìm cây khung nhỏ nhất trong đồ thị

Cho n thành phố và các khoảng cách 𝑑𝑖𝑗 giữa mỗi cặp thành phố, tìm tour ngắn nhất sao cho mỗi thành phố được viếng thăm chỉ một lần.

\* Dữ liệu đầu vào cần có:

- Trạng thái ban đầu: Agent ở thành phố bắt đầu và không viếng thăm bất kỳ thành phố nào khác.

- Trạng thái kết thúc: Agent đã viếng thăm tất cả các thành phố và đến thành phố bắt đầu 1 lần nữa.

- Hàm successor: khởi tạo tất cả các thành phố chưa viếng thăm.

- Chi phí cạnh: khoảng cách giữa các thành phố được biểu diễn bởi các nút, sử dụng chi phí này để tính g(n).

- h(n): khoảng cách tới thành phố chưa viếng thăm gần nhất \_ ước lượng khoảng cách đi từ tất cả thành phố bắt đầu.

### **a.**  Thuật toán Kruskal: là một thuật toán trong lý thuyết đồ thị để tìm cây bao trùm nhỏ nhất của một đồ thị liên thông có trọng số. Nói cách khác, nó tìm một tập hợp các cạnh tạo thành một cây chứa tất cả các đỉnh của đồ thị và có tổng trọng số các cạnh là nhỏ nhất.

Các bước thực hiện bài toán:

Bước 1: Khởi tạo T:= Ø sau đó sắp xếp các cạnh của đồ thị theo thứ tự trọng số tang dần

Bước 2: Lần lược lấy từng cạnh thuộc danh sách đaz sắp xếp. nếu T+{e} không chứa chu trình thì gán T:=T+{e}

Nếu T+{e} chứa chu trình thì ta bỏ qua

Bươc 3: Nếu T đủ n-1 phần tử thì dừng, ngược lại quay lại bước 2.

Ví dụ: Cho bài toán sau

Shape

Description automatically generated with low confidence

Bước 1: Khởi tạo cây T:= Ø có 7 đỉnh

Sắp xếp theo chiều tăng dần trọng số ta được:

(A,D): 5 (A,B): 7 (E,F): 8 (F,G): 11

(C,E): 5 (B,E): 7 (D,B): 9 (D,E): 15

(D,F): 6 (B,C): 8 (E,G): 9

Bước 2:

Thêm cạnh (A,D) vào T, không tạo chu trình nên T:={(A,D)}. Do số cạnh T = 1 <(7-1) nên

Ta them cạnh (C,E) vào T, Không tạo chu trình nên T:={(A,D), (C,E)}, do số cạnh T = 2 <6 nên

Ta them cạnh (D,F) vào T, Không tạo chu trình nên T:={(A,D), (C,E), (D,F)}, do số cạnh T=3 < 6 nên

Ta thêm cạnh (A,B) vào T, Không tạo chu trình nên T:={(A,D), (C,E), (D,F), (A,B)}, do số cạnh T=4<6 nên

Ta thêm cạnh (B,E) vào T, Không tạo chu trình nên T:={(A,D), (C,E), (D,F), (A,B), (B,E)}, do số cạnh T=5<6 nên

Ta thêm cạnh (B,C) vào T, Tạo chu trình (B,E,F) nên bỏ qua

Ta thêm cạnh (E,F) vào T, thấy thấy tạo chu trình (A,B,E,F,D)

Ta thêm cạnh (D,B) vào T, thấy thấy tạo chu trình (A,B,D)

Ta thêm cạnh (E,G) vào T, Không tạo chu trình nên T:={(A,D), (C,E), (D,F), (A,B), (B,E), (E,G}, do số cạnh T=6 =(7-1) nên ta suy ra đường đi:

C

A 7 B

7 E 5

5

9

D 6

F G

Tổng đường đi là:39

B. **Thuật toán Prim:**

Thuật toán xuất phát từ một cây chỉ chứa đúng một đỉnh và mở rộng từng bước một, mỗi bước thêm một cạnh mới vào cây, cho tới khi bao trùm được tất cả các đỉnh của đồ thị.

* Dữ liệu vào: Một đồ thị có trọng số liên thông với tập hợp đỉnh *V* và tập hợp cạnh *E* (trọng số có thể âm). Đồng thời cũng dùng *V* và *E* để ký hiệu số đỉnh và số cạnh của đồ thị.
* Khởi tạo: *V*mới = {*x*}, trong đó *x* là một đỉnh bất kì (đỉnh bắt đầu) trong *V*, *E*mới = {}
* Lặp lại cho tới khi *V*mới = *V*:
  + Chọn cạnh (*u*, *v*) có trọng số nhỏ nhất thỏa mãn *u* thuộc *V*mới và *v* không thuộc *V*mới (nếu có nhiều cạnh như vậy thì chọn một cạnh bất kì trong chúng)
  + Thêm *v* vào *V*mới, và thêm cạnh (*u*, *v*) vào *E*mới
* Dữ liệu ra: *V*mới và *E*mới là tập hợp đỉnh và tập hợp cạnh của một cây bao trùm nhỏ nhất

Ví dụ: Cho lại bài toán sau:

Shape

Description automatically generated with low confidence

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Đỉnh  Bước | A | B | C | D | E | F | G |
| KHỞI TẠO | - | [A,7] | [A,∞] | [A,5] | [A,∞] | [A,∞] | [A,∞] |
| 1 | - | [A,7] | [A,∞] | - | [D,15] | [D,6] | [A,∞] |
| 2 | - | [A,7] | [A,∞] | - | [F,8] | - | [F,11] |
| 3 | - | - | [B,8] | - | [B,7] | - | [F,11] |
| 4 | - | - | [E,5] | - | - | - | [E,9] |
| 5 | - | - | - | - | - | - | [E,9] |
| 6 | - | - | - | - | - | - | - |

\*”- “ Khởi tạo giá trị nhỏ nhất

“-“: các giá trị đã khởi tạo

\*Lưu ý : trong quá trình khởi tạo nếu giá trị mới nhỏ hơn giá trị cũ thì giữ nguyên giá trị cũ

Kết luận : tổng đường đi là: 39

Từ 2 thuật toán trên ta có thể kết luận rằng chúng cho ra cùng một kết quả

II. Cài đặt chương trình:

Theo như quá tình nhập code thì em nhận thấy không có lỗi gì

Nhiệm vụ chức năng các hàm :

+ class TreeNode : định nghĩa class để biểu diễn các node tree trong khi mở rộng thuật toán A\*

Graphical user interface, application, Word

Description automatically generated

+ class FringeNode: định nghĩa class để biểu diễn các node fringe có trong danh sách của thuật toán A\*

Text

Description automatically generated

+ Xây dựng class Graph:

* printMST: Hàm hiển thị cấu trúc MST đã được lưu trong node Parent , kết quả trả về sẽ là phần trăm tổng của sum\_weight, min1 và min2. Về mặt lý thuyết, ta sẽ có các bước như sau:

Bước 1: tìm cạnh ngắn nhất trong đồ thị. Nếu có nhiều hơn 1 cạnh như vậy thì chọn 1 ngẫu nhiên 1 cạnh. Đánh dấu cạnh này và các đỉnh được kết nối.

Bước 2: Chọn cạnh ngắn nhất tiếp theo, trừ khi nó tạo thành 1 chu trình với các cạnh đã được đánh dấu trước đó. Đánh dấu cạnh đó và các đỉnh được kết nối.

Bước 3: Nếu tất cả các cạnh được kết nối thì khi đó ta đã hoàn thành. Ngược lại, lặp lại Bước 2.

Text

Description automatically generated

* Hàm minKey : Hàm tìm đỉnh với giá trị khoảng cách ( chi phí) nhỏ nhất từ tập hợp các đỉnh.

Text

Description automatically generated

* Hàm primMST: dùng để xây dựng và hiện thị đồ thị MST và được thể hiện bằng ma trận kề

Text

Description automatically generated

-Hàm heuristic: Xây dựng heuristic và trả về mst\_weight hoặc đồ thị tùy vào số thành phố đã thăm.

Bước 1: Xây dựng một cây khung nhỏ nhất.

Bước 2: Chọn nút gốc là nút bất kỳ.

Bước 3: Duyệt qua tất cả các đỉnh bằng tìm kiếm theo chiều sâu, ghi lại tất các đỉnh (đỉnh đã viếng thăm và đỉnh chưa viếng thăm).

Bước 4: Sử dụng chiến lược nhanh chóng trực tiếp hơn để khởi tạo một tour.

Text

Description automatically generated with medium confidence

* Hàm checkPath : hàm kiểm tra đường đi chiều sâu và chiều rộng cũng như số lượng node còn trong list. Sẽ in về “path complete” nếu chiều sâu, chiều rộng bằng với số đỉnh và không còn node để chờ mở rộng.

Text

Description automatically generated

* Hàm startTSP :Hàm áp dụng thuật toán A\* để giải bài toán TSP. Khi trả về sẽ trả về chi phí của bài toán. Trong hàm này, chúng ta gọi lại các hàm đã xây dựng ở trên.

Text

Description automatically generated

Sau khi chạy tất cả các hàm trên, code không có lỗi và sẽ in ra kết quả với dữ liệu đầu vào ở hàm main: ( V = 4, graph=[[0,5,2,3],[5,0,6,3],[2,6,0,4],[3,3,4,0]]).

Kết quả :

Text

Description automatically generated with medium confidence

Tuy nhiên với V=4 , graph = [[0,300,200],[300,0,500],[200,500,0]] sẽ có lỗi

Khi đó ta cần thay V=3 vào , Với V là số đỉnh tương ứng

Kết quả

Text

Description automatically generated

Tương tự : #V=2

#graph = [[0,300],[300,0]]

Em xin chân thành cảm ơn thầy cô đã xem !