Họ tên: Đinh Xuân Khang

MSSV: 22280042

BÁO CÁO THỰC HÀNH TUẦN 5

Chương trình:

Chương trình đã cho không bị báo lỗi.

Bài toán người bán hàng- Travelling Salesperson Problem (TSP):

Về bài toán: Giả sử một nhân viên bán hàng muốn đến thăm một số thành phố nhất định được giao cho anh ta. Anh ta biết khoảng cách của cuộc hành trình giữa mọi cặp thành phố. Vấn đề của anh ta là chọn một con đường bắt đầu từ thành phố bắt đầu của anh ta, đi qua mỗi thành phố chính xác một lần và trở về thành phố bắt đầu của anh ta một khoảng cách ngắn nhất có thể. Bài toán này liên quan mật thiết đến việc tìm một đoạn mạch Hamilton có độ dài nhỏ nhất. Nếu chúng ta biểu diễn các thành phố bằng các đỉnh và đường nối hai cạnh của các thành phố, chúng ta sẽ có được một đồ thị có trọng số.

Yêu cầu bài toán: Cho trước n thành phố và các khoảng cách d_{ij} giữa mỗi cặp thành phố, tìm tour ngắn nhất sao cho mỗi thành phố được viếng thăm chỉ một lần.

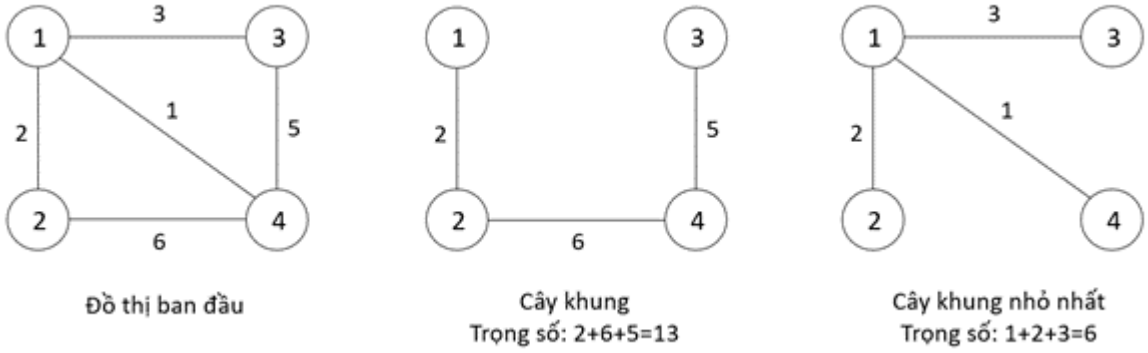
Cách giải bài toán: ta sử dụng thuật toán $A*$ để giải quyết bài toán TSP và heuristic được sử dụng là ***cây khung nhỏ nhất***.

- 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.

Cây khung nhỏ nhất (Minimum Spanning Tree – MST):

- Cây khung là một tập các cạnh sao cho tập cạnh này không tồn tại chu trình và liên thông.
- Cây khung nhỏ nhất là cây khung mà tổng trọng số các cạnh thuộc cây khung là nhỏ nhất

Ví dụ:



Mô tả cây khung nhỏ nhất. (Nguồn: howkteam.vn)

Thuật toán cho MST – Thuật toán Prim:

Thuật toán trong bài được sử dụng để tìm cây khung nhỏ nhất là **thuật toán Prim. Thuật toán Prim** là một thuật toán tham lam để tìm cây khung nhỏ nhất của một đồ thị vô hướng có trọng số liên thông. Ý tưởng xây dựng thuật toán Prim khá giống với ý tưởng của thuật toán Dijkstra: Tại mỗi bước, ta sẽ thêm một đỉnh vào cây khung thoả mãn đỉnh đó chưa được chọn vào cây khung và khoảng cách từ nó đến một đỉnh đã được chọn là nhỏ nhất.

Mô tả các bước:

- 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.

Thuật toán Heuristics chèn gần nhất (Nearest Insertion):

Là thuật toán đơn giản và dễ hiểu để giải quyết bài toán TSP. **Nearest Insertion** sử dụng kỹ thuật mở rộng đường đi (hay còn được gọi là tour), bằng cách chèn những điểm mới vào những điểm trong đường đi trước đó.

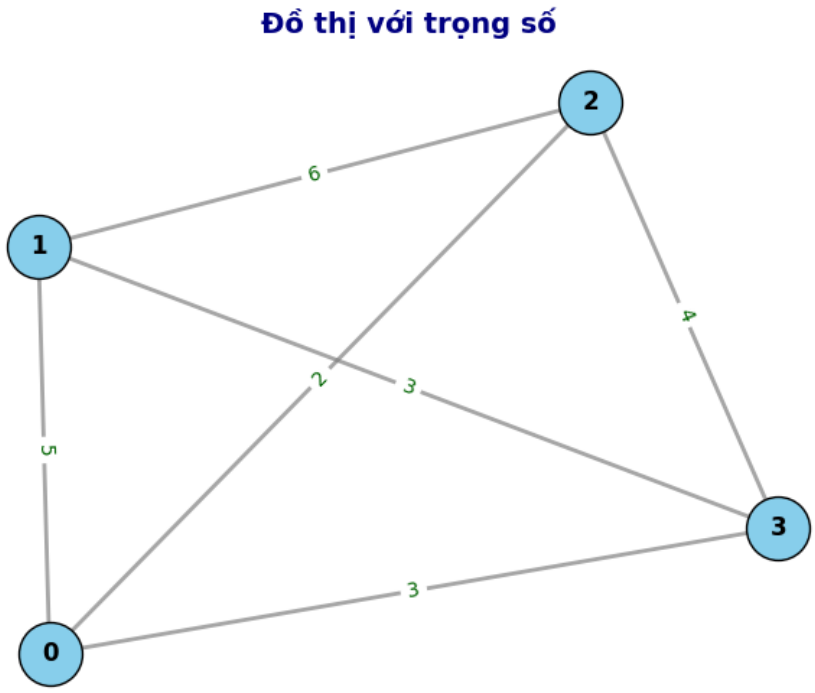
Mô tả các bước:

- Bước 1: Chọn 1 nút *v* bất kỳ và cho chu trình *C* chỉ chứa *v*.
- Bước 2: Tìm một nút bên ngoài *C* gần nhất với 1 nút trong *C*, gọi là *k*.
- Bước 3: Tìm 1 cạnh *{ij}* trong *C* sao cho $d_{ik} + d_{kj} - d_{ij}$ là tối tiểu.
- Bước 4: Xây dựng một chu trình *C* mới bằng việc thay thế *{ij}* với *{ik}* và *{k,j}*.
- Bước 5: Nếu chu trình *C* hiện tại chứa tất cả các đỉnh thì dừng. Ngược lại, quay lại Bước 2.

Kết quả từ Chương trình Python:

Input: Ma trận kề của đồ thị có 4 đỉnh:

```
[[0, 5, 2, 3], [5, 0, 6, 3], [2, 6, 0, 4], [3, 3, 4, 0]]
```



Hình biểu diễn ma trận kề từ đồ thị.

Output: Chu trình Hamilton ngắn nhất cùng với trọng số

```
Path complete

[0, 2, 3, 1, 0]

Ans is 14
```

Nhận xét: thuật toán được sử dụng đưa ra kết quả đúng và chính xác với bài toán TSP.