|  |  |
| --- | --- |
| Trường Đại học Khoa học Tự Nhiên  **ĐHQG.TPHCM** | **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**  **Độc lập - Tự do - Hạnh phúc**  *TPHCM,ngày 2 tháng 12 năm 2024* |

**BÁO CÁO THỰC HÀNH – NHẬP MÔN TRÍ TUỆ NHÂN TẠO – LẦN 5**

**Họ và tên: Nguyễn Thanh Kiên.**

**MSSV: 22110092.**

**Nhắc lại kiến thức**

1. **Bài toán Travelling salesman problem (TSP).**

Cho trước 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.

Bài toán người du hành (TSP) là một trong những vấn đề nổi bật và khó khăn nhất về lý thuyết đồ thị, được giới thiệu từ thế kỷ XVII bởi Hamilton và Kirkman. Đây là bài toán NP-hard với độ phức tạp tăng theo hàm số mũ, lần đầu được giải trên máy tính vào năm 1954 (49 đỉnh) và đến năm 2004 đạt tới 24.978 đỉnh, dự kiến sẽ còn tăng nữa.

TSP yêu cầu tìm hành trình đi qua tất cả các thành phố đúng 1 lần và quay lại điểm xuất phát sao cho tổng quãng đường là nhỏ nhất. Bài toán được biểu diễn dưới dạng đồ thị có trọng số, trong đó mỗi cặp đỉnh được nối với nhau bằng một cạnh.

Ngày nay, nhiều phương pháp đã được phát triển để giải quyết TSP. Trong bài báo cáo này, chúng ta sẽ tìm hiểu cách giải sử dụng thuật toán A\*, kết hợp với cây khung nhỏ nhất (MST) và thuật toán heuristic chèn gắn gần nhất (Nearest Insertion).

A map with red pins and a couple of red pins on it

Description automatically generated

Một lời giải trực quan cho bài toán TSP

1. **Cây khung nhỏ nhất – Minimum Spanning Tree – MST**

Trong lý thuyết về đồ thị, cây khung nhỏ nhất của đồ thị có trọng số G = [V, E] là một tập hợp các cạnh của đồ thị sao cho:

* Tập hợp các cạnh này không chứa chu trình và liên thông, hay nói cách khác là từ một đỉnh bất kỳ có thể đi tới các đỉnh khác mà chỉ dùng các cạnh trên tập hợp đỉnh đó.
* Tổng trọng số của các cạnh trong tập hợp này là nhỏ nhất.

Minh họa cây khung và MST của nó với đồ thị vô hướng có trọng số

A diagram of a tree

Description automatically generated

1. **Thuật toán Prim**

Trong khoa học máy tính, **thuật toán Prim** là một thuật toán tham lam để tìm cây bao trùm nhỏ nhất của một đồ thị vô hướng có trọng số liên thông. Nó khá giống với thuật toán Dijikstra (thuật toán tìm 1 đường đi ngắn nhất trên đồ thị), thuật toán Prim kết nạp từng đỉnh của đồ thị theo điều kiện độc nhất: đỉnh được kết nạp vào tiếp theo phải chưa được nạp và gần nhất với các đỉnh đã được nạp và đồ thị. Ta sẽ sử dụng một hàng đợi ưu tiên PriorQueue để lưu lại các cạnh có trọng số nhỏ nhất. Nhận xét một chút về độ phức tạp lúc này, là O((V + E)logV), vì mỗi đỉnh được thêm vào PO chỉ một lần và thao tác thêm phần tử vào hàng đợi có chi phí thời gian là hàm Logarit.

A white page with black text

Description automatically generated

Minh họa thuật toán Prim

A group of lines with dots and circles

Description automatically generated with medium confidence

Minh họa Trực quan thuật toán Prim bằng Graph

1. **Giải thuật Heuristic Nearest Insertion**

Hiện tại, lời giải tối ưu cho bài toán TSP vẫn chưa được tìm ra, nhưng tuy vậy, có một vài lời giải mà chúng ta đều có thể tạm chấp nhận. Một trong số đó là sử dụng giải thuật heuristic, cụ thể là phương pháp Chèn gắn gần nhất (Nearest Insertion).

Nearest Insertion mở rộng hành trình (tour) bằng cách chèn điểm mới vào giữa các điểm hiện có, đảm bảo tổng trọng số là nhỏ nhất. Phương pháp này có độ phức tạp O(n2) vì các bước tìm đỉnh và cạnh để chèn có độ phức tạp là O(n).

A white paper with black text

Description automatically generated

Minh họa cho giải thuật.

A group of triangles with lines and letters

Description automatically generated with medium confidence

Minh họa bằng trực quan cho giải thuật.

**Thực hành và Coding:**

1. **Nhận xét về Code:**

* **Graph.printMST(self, parent, g, d\_temp, t)**: In ra cây khung và trả về trọng số của cây khung nhỏ nhất.
* **Graph.minKey(self, key, mstSet):** Tìm giá trị nhỏ nhất trong tập đỉnh thuộc cây khung.
* **Graph.primMST(self, g, d\_temp, t):** Thực hiện thuật toán Prim tìm cây khung nhỏ nhất.
* **heuristic(tree, p\_id, t, V, graph):** Thực hiện giải thuật heuristic Nearest Insertion.
* **checkPath(tree, toExpand, V):** Kiểm tra và in ra đường đi ngắn nhất.
* **tartTSP(graph, tree, V):** Giải bài toán TSP bằng thuật toán A\*.

1. **Sửa lỗi – Debug**

Đoạn code cho trong File thực hành xuất hiện một số lỗi sau

1. Phải import thư viện TreeLib

Ta sử dụng: !pip install treelib trên colab hoặc pip install treelib trong cmd để sử dụng trong các IDE.

1. def minKey

Ta phải khai báo biến min\_index = một giá trị cụ thể nào đó, ở đây em khai báo = -1

min\_index = -1

1. def Heuristic

* Với l = len(visited), ta phải khai báo num = V – l, không phải V – 1.

Sau khi debug 3 vấn đề trên, đoạn code chạy không bị lỗi.

1. Chạy và kết quả

Xây dựng hàm run\_test\_cases để chạy 6 Test Case:

A screenshot of a computer program

Description automatically generated

Kết quả:

A white background with black text

Description automatically generatedA white background with black text

Description automatically generated

A white paper with black text

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generatedA white background with black lines and numbers

Description automatically generated

**Nhận xét:**

* Chương trình cho ở trên chỉ giải được các bài toán khi và chỉ khi từng cặp đỉnh trong đồ thị được nối với nhau bởi 1 cạnh.
* Thuật toán trên sẽ chạy rất chậm nếu số lượng đỉnh lớn (5, 6, 7,…)
* Ngoài HNI (heuristic nearest insertion), ta có thể sử dụng các giải thuật khác như Lin-Kernighan hay 3-opt,…

Người báo cáo

Nguyễn Thanh Kiên