**YÊU CẦU BÀI TẬP LỚN MÔN TOÁN RỜI RẠC**

--o0o--

**Yêu cầu:**

1. Mỗi nhóm gồm 3 SV / 1 đề tài. Nếu không chọn được nhóm thì chọn ngẫu nhiên 03 SV 01 nhóm.
2. Định dạng báo cáo theo mẫu chung của khoa.
3. Cấu trúc báo cáo bài tập lớn gồm những nội dung sau:
4. Mục lục
5. Lời nói đầu
6. Chương 1: Trình bày những kiến thức cơ sở liên quan như khái niệm, định nghĩa, định lý, hệ quả, tính chất, ứng dụng,…
7. Chương 2: Giải quyết vấn đề của đề tài gồm:

2.1. Mô tả bài toán  
2.2. Xây dựng thuật toán  
2.3. Một vài ví dụ minh hoạ

1. Chương 3: Trình bày cài đặt thuật toán bằng Python/C++
2. Kết luận
3. Tài liệu tham khảo

**Đề tài:**

**Chủ đề 1: Tối ưu hoá mạng lưới giao thông**

Giả sử bạn được giao nhiệm vụ lập kế hoạch tuyến đường giao thông trong một thành phố lớn, nơi có nhiều con đường nối giữa các địa điểm quan trọng. Thành phố này có một hệ thống giao thông phức tạp với nhiều tuyến đường có độ dài khác nhau. Bạn cần phải xây dựng một ứng dụng để tìm ra tuyến đường ngắn nhất từ một điểm xuất phát đến một điểm đến dựa trên khoảng cách của các con đường.

**Yêu cầu:**

* Xây dựng chương trình giải quyết bài toán trên sử dụng **thuật toán Dijkstra** để tìm đường đi ngắn nhất giữa hai điểm trong thành phố.
* Thành phố được mô tả như một đồ thị với các điểm giao thông là các đỉnh và các con đường là các cạnh có trọng số (khoảng cách giữa các điểm).
* Đầu vào là các điểm bắt đầu và kết thúc, cùng với danh sách các con đường (đỉnh và trọng số).
* Đầu ra là tuyến đường đi ngắn nhất và tổng khoảng cách của tuyến đường.

**Nhiệm vụ:**

1. Nhập vào danh sách các địa điểm trong thành phố (được đánh số từ 1 đến N) và danh sách các con đường giữa các địa điểm đó.
2. Áp dụng thuật toán Dijkstra để tìm đường đi ngắn nhất từ một địa điểm cụ thể đến địa điểm khác.
3. In ra tuyến đường ngắn nhất cùng với khoảng cách của tuyến đường đó.

**Dữ liệu thử nghiệm:** [**Dijsktra**](Large_Dijkstra_Test_Data.csv)

**Chủ đề 2: Khám phá một hệ thống mạng máy tính**

Giả sử bạn đang quản lý một mạng máy tính trong một công ty lớn, nơi có rất nhiều máy chủ và các máy tính được kết nối với nhau. Nhiệm vụ của bạn là phát hiện tất cả các máy tính có thể truy cập được từ một máy chủ cụ thể. Tuy nhiên, mạng này rất phức tạp và không phải lúc nào các máy tính cũng có thể được kết nối trực tiếp với nhau.

**Yêu cầu:**

* **Đầu vào**: Một hệ thống mạng gồm các máy tính và các kết nối giữa chúng. Mỗi máy tính được coi là một đỉnh trong đồ thị, và mỗi kết nối mạng giữa hai máy tính được coi là một cạnh của đồ thị.
* **Đầu ra**: Bạn cần phải sử dụng thuật toán **Duyệt theo chiều sâu (DFS)** để khám phá và liệt kê tất cả các máy tính có thể truy cập từ một máy chủ ban đầu, trong hệ thống mạng này.

**Nhiệm vụ:**

* + Mô tả các thuật toán duyệt đồ thị
  + Áp dụng thuật toán DFS để duyệt qua hệ thống mạng, bắt đầu từ một máy chủ bất kỳ.
  + Tìm và liệt kê tất cả các máy tính có thể truy cập từ máy chủ ban đầu.
  + Kiểm tra xem có bất kỳ máy nào trong hệ thống không thể truy cập từ máy chủ ban đầu hay không.

**Dữ liệu thử nghiệm:** [**DFS**](Large_DFS_Test_Data.csv)

**Chủ đề 3: Lập kế hoạch di chuyển cho xe vệ sinh.**

Giả sử bạn được giao nhiệm vụ lên lịch cho xe vệ sinh quét đường sao cho tất cả các con đường trong khu vực đô thị được dọn sạch mà xe không phải đi lại nhiều lần trên cùng một tuyến nhằm tối ưu hóa thời gian và chi phí vận hành.

**Yêu cầu:**

* **Đầu vào:**
* Các **đỉnh** đại diện cho các giao lộ hoặc điểm nút giao thông.
* Các **cạnh** đại diện cho các đoạn đường nối giữa các giao lộ.
* Mỗi đường phố được quét đúng một lần.
* Xe quét phải quay trở lại điểm xuất phát (chu trình Euler) mà không cần đi lại trên cùng một đường.
* **Đầu ra: Bản đồ tuyến đường** cho xe quét đường sao cho tất cả các đường trong khu vực được dọn sạch mà xe không phải đi lại nhiều lần trên cùng một tuyến.

**Nhiệm vụ:**

1. Nhập vào danh sách các giao lộ và danh sách các con đường nối giữa các giao lộ đó.
2. Áp dụng thuật toán Fleury tìm chu trình Euler biểu diễn tuyến đường cho xe vệ sinh.
3. In ra tuyến đường đó.

**Dữ liệu thử nghiệm:**

**Chủ đề 4: Tối ưu hóa mạng lưới viễn thông**

Giả sử bạn được giao nhiệm vụ thiết kế mạng cáp quang hoặc hạ tầng truyền thông, kết nối các trạm viễn thông sao cho tổng chi phí kết nối đường cáp là tối thiểu.

**Yêu cầu:**

**Đầu vào:**

* Một đồ thị liên thông, vô hướng với các đỉnh và cạnh.
* **Đỉnh**: Đại diện cho các trạm
* **Cạnh**: Đại diện cho kết nối giữa các đỉnh, với mỗi cạnh có **trọng số** tương ứng (chi phí kết nối)

#### **Đầu ra**: cây khung nhỏ nhất (MST) kết nối tất cả các trạm với tổng chi phí nhỏ nhất, thông qua việc sử dụng thuật toán Prim.

**Nhiệm vụ cụ thể:**

1. Nhập vào danh sách các trạm và chi phí kết nối giữa các trạm đó.
2. Áp dụng thuật toán Prim tìm CKNN kết nối tất cả các trạm.
3. In ra cách kết nối đó.

**Chủ đề 5: Ứng dụng thuật toán Bell man – Ford trong quản lý rủi ro tài chính**

Giả sử bạn bắt đầu với 1 USD và thực hiện các giao dịch chuyển đổi qua nhiều loại tiền tệ. Nếu có chênh lệch tỷ giá, bạn có thể quay lại với nhiều hơn 1 USD, đây là hiện tượng arbitrage. Nó xảy ra khi tỷ giá không nhất quán, cho phép mua rẻ ở một thị trường và bán đắt ở nơi khác. Bạn cần hát hiện các cơ hội **arbitrage** trong các giao dịch tiền tệ quốc tế. Giúp nhà đầu tư tìm kiếm các cơ hội sinh lời từ sự chênh lệch tỷ giá giữa các thị trường khác nhau.

**Yêu cầu đầu vào:**

* **Đồ thị có hướng**, trong đó:
  + **Đỉnh** đại diện cho các thực thể tài chính (ví dụ: sàn giao dịch, ngân hàng, hoặc loại tiền tệ).
  + **Cạnh có trọng số** đại diện cho **biến động giá**, **lãi suất**, hoặc **tỷ giá hối đoái** giữa các thực thể tài chính. Trọng số có thể dương hoặc âm.
* **Đỉnh nguồn**: Một thực thể tài chính ban đầu mà ta cần tính toán giá trị giao dịch so với các thực thể còn lại.
* Dữ liệu có thể chứa **trọng số âm** để biểu thị các **lãi suất âm** hoặc sự **mất giá** trong thị trường tài chính.

**Yêu cầu đầu ra:**

* Đường đi ngắn nhất từ đỉnh nguồn đến tất cả các đỉnh khác (thực thể tài chính) trong đồ thị.

#### **Nhiệm vụ**:

#### Sử dụng thuật toán Bellman-Ford tính toán đường đi ngắn nhất từ một thực thể tài chính (đỉnh nguồn) đến các thực thể khác trong mạng lưới tài chính, bao gồm cả những trọng số âm.

**Chủ đề 6: Ứng dụng thuật toán BFS trong việc tìm đường đi trên một bản đồ địa hình**

Giả sử bạn có một bản đồ địa hình với các ô có thể chứa đất (0) hoặc nước (1). Bạn cần tìm đường đi từ vị trí xuất phát (S) đến vị trí đích (E) mà chỉ có thể di chuyển qua các ô chứa đất. Di chuyển có thể được thực hiện lên, xuống, trái, phải.

**Đầu Vào**

1. **Bản đồ địa hình**:
2. **Vị trí xuất phát**
3. **Vị trí đích**

### **Đầu Ra**

* Đường đi ngắn nhất từ S đến E.
* Danh sách các ô mà người chơi sẽ đi qua:

**Chủ đề 7: Tối ưu hóa mạng máy tính**

Giả sử bạn là một kỹ sư mạng máy tính và bạn cần kết nối một số máy chủ trong một trung tâm dữ liệu. Bạn có danh sách các kết nối khả thi giữa các máy chủ cùng với chi phí kết nối. Bạn muốn thiết lập một mạng lưới tối ưu với chi phí tối thiểu, đồng thời đảm bảo rằng tất cả các máy chủ đều được kết nối với nhau.

Sử dụng thuật toán Kruskal trong việc thiết kế và tối ưu hóa mạng máy tính trên.

### Dữ liệu đầu vào

Dữ liệu đầu vào cho thuật toán Kruskal sẽ là danh sách các cạnh của đồ thị cùng với chi phí tương ứng.

**Đầu ra**

Đầu ra của thuật toán sẽ là danh sách các kết nối trong mạng máy tính tối thiểu và tổng chi phí lắp đặt.

### **Chủ đề 8: Tối ưu hóa lựa chọn đồ vật trong bài toán ba lô bằng thuật toán nhánh cận**

Bạn là một nhà thám hiểm và đang chuẩn bị cho một chuyến đi dài. Bạn có một ba lô với trọng lượng tối đa cho phép là b kg. Bạn có một số đồ vật cần mang theo, mỗi đồ vật có một trọng lượng và một giá trị khác nhau. Mục tiêu của bạn là tối ưu hóa việc lựa chọn đồ vật để tối đa hóa tổng giá trị mang theo mà không vượt quá trọng lượng tối đa của ba lô.

#### Yêu cầu đầu vào

1. **Trọng lượng tối đa của ba lô**: Một số nguyên dương (ví dụ: 10 kg).
2. **Danh sách đồ vật**: Một danh sách chứa thông tin về từng đồ vật, bao gồm:
   * ID đồ vật
   * Trọng lượng (kg)
   * Giá trị (tiền tệ)

#### Yêu cầu đầu ra

1. **Danh sách đồ vật được chọn**: Một danh sách các ID của đồ vật đã chọn.
2. **Tổng giá trị**: Một số nguyên thể hiện tổng giá trị của các đồ vật đã chọn.
3. **Tổng trọng lượng**: Một số nguyên thể hiện tổng trọng lượng của các đồ vật đã chọn.

#### Nhiệm vụ cụ thể

1. **Phát triển thuật toán**: Sử dụng phương pháp nhánh cận để tìm kiếm giải pháp tối ưu cho bài toán ba lô.
2. **Thực hiện kiểm tra ràng buộc**: Đảm bảo rằng tổng trọng lượng của các đồ vật không vượt quá trọng lượng tối đa của ba lô.
3. **In ra kết quả**: Hiển thị danh sách các đồ vật được chọn, tổng giá trị và tổng trọng lượng.
4. **Phân tích và so sánh**: So sánh kết quả với các phương pháp khác (nếu có) để đánh giá hiệu quả của thuật toán nhánh cận.

### **Chủ đề 9: Mô hình hóa mạng xã hội**

Trong mạng xã hội, người dùng tương tác với nhau thông qua việc kết nối (follow, friend). Mỗi người dùng có thể ảnh hưởng đến những người khác trong mạng lưới. Mục tiêu của bài toán này là sử dụng lý thuyết đồ thị để mô hình hóa mạng xã hội, phân tích mối quan hệ giữa người dùng, tìm kiếm các cộng đồng và xác định những người ảnh hưởng trong mạng.

#### Yêu cầu đầu vào

1. **Dữ liệu người dùng**: Một danh sách chứa thông tin về người dùng trong mạng xã hội. Mỗi người dùng có thể có các thuộc tính như:
   * ID người dùng
   * Tên người dùng
   * Số lượng người theo dõi (followers)
   * Số lượng người bạn (friends)
2. **Mối quan hệ giữa người dùng**: Một ma trận hoặc danh sách các cạnh biểu diễn mối quan hệ giữa các người dùng:
   * Nếu người dùng A theo dõi người dùng B, có một cạnh từ A đến B trong đồ thị.

#### Yêu cầu đầu ra

1. **Mô hình đồ thị**: Biểu diễn mạng xã hội dưới dạng đồ thị, trong đó các đỉnh là người dùng và các cạnh là mối quan hệ.
2. **Người ảnh hưởng**: Xác định những người dùng có ảnh hưởng lớn nhất trong mạng lưới bằng các chỉ số như Degree Centrality

Nhiệm vụ cụ thể

1. **Xây dựng đồ thị**:
   * Sử dụng dữ liệu đầu vào để xây dựng đồ thị mạng xã hội.
   * Sử dụng thư viện đồ thị như NetworkX (Python) để dễ dàng thao tác.
2. **Xác định người ảnh hưởng**:
   * Tính toán các chỉ số trung tâm cho từng người dùng và xác định người ảnh hưởng dựa trên các chỉ số này.
   * Trình bày kết quả và phân tích những người có ảnh hưởng cao nhất.
3. **Báo cáo kết quả**:
   * Tổng hợp và báo cáo các kết quả phân tích, cộng đồng và người ảnh hưởng.
   * Đưa ra khuyến nghị dựa trên kết quả phân tích.

### Dữ liệu thử nghiệm

Sử dụng một tập dữ liệu mẫu với khoảng 100 người dùng và mối quan hệ giữa họ. Bạn có thể tạo dữ liệu giả hoặc sử dụng dữ liệu từ các nguồn mở như các tập dữ liệu mạng xã hội (ví dụ: Facebook, Twitter).

**Chủ đề 10: Bài toán đếm và ứng dụng trong sắp xếp đội ngũ nhân viên cho một sự kiện.**

Một công ty tổ chức sự kiện và cần sắp xếp n nhân viên vào k vị trí trên sân khấu. Các vị trí này có thể là: người điều phối, người thuyết trình và người hỗ trợ kỹ thuật, ... Tuy nhiên, có một só điều kiện đặc biệt như là một số nhân viên không thể đứng cạnh nhau hay chỉ cố định tại 1 vị trí,… do tính chất công việc.

#### Yêu cầu đầu vào:

1. **Số lượng nhân viên (N)**: Số nhân viên có sẵn để tham gia sự kiện.
2. **Số lượng vị trí (K)**: Số vị trí cần lấp đầy trong đội ngũ.
3. **Các điều kiện đặc biệt**: Ví dụ như một số nhân viên không thể đứng cạnh nhau hoặc một số vị trí chỉ có thể do một số nhân viên nhất định đảm nhiệm.

#### Yêu cầu đầu ra:

1. **Tổng số cách sắp xếp**: Số lượng cách có thể sắp xếp các nhân viên vào các vị trí.
2. **Danh sách các cách sắp xếp (nếu cần)**: Một hoặc một vài ví dụ về cách sắp xếp cụ thể.

#### Nhiệm vụ cụ thể:

1. **Xác định công thức**: Sử dụng các nguyên tắc trong toán rời rạc (như hoán vị, tổ hợp) để xây dựng công thức tính số cách sắp xếp.
2. **Lập chương trình**: Viết một chương trình bằng Python để tính toán số cách sắp xếp dựa trên đầu vào.
3. **Xử lý điều kiện đặc biệt**: Cài đặt các điều kiện để đảm bảo rằng các sắp xếp không vi phạm các quy tắc đã đề ra.
4. **Thử nghiệm**: Thực hiện các thử nghiệm với các bộ dữ liệu khác nhau để đảm bảo tính chính xác của chương trình.

### **Chủ đề 11: Tối ưu hóa lộ trình vận chuyển hàng hóa trong một thành phố**

Một công ty giao hàng cần vận chuyển hàng hóa từ kho (điểm xuất phát) đến một khách hàng ở một địa điểm cụ thể (điểm đích) trong một thành phố. Công ty muốn tìm lộ trình ngắn nhất về thời gian hoặc khoảng cách để tiết kiệm chi phí và thời gian giao hàng.

#### Yêu cầu đầu vào:

1. **Đồ thị**: Một đồ thị có hướng đại diện cho mạng lưới đường phố trong thành phố, trong đó:
   * Các đỉnh (node) là các nút giao thông (điểm xuất phát, điểm đến, và các giao lộ).
   * Các cạnh (edge) là các đoạn đường giữa các nút, với trọng số tương ứng là thời gian hoặc khoảng cách di chuyển.
2. **Điểm xuất phát (source)**: Đỉnh đại diện cho vị trí bắt đầu của lộ trình.
3. **Điểm đích (destination)**: Đỉnh đại diện cho vị trí cần đến.

#### Yêu cầu đầu ra:

1. **Lộ trình tối ưu**: Danh sách các đỉnh (đường đi) từ điểm xuất phát đến điểm đích với tổng trọng số (thời gian hoặc khoảng cách) nhỏ nhất.
2. **Tổng trọng số**: Giá trị tổng thời gian hoặc khoảng cách của lộ trình tối ưu.

#### Nhiệm vụ cụ thể:

1. **Mô hình hóa đồ thị**: Xây dựng mô hình đồ thị từ dữ liệu thực tế về đường phố trong thành phố.
2. **Triển khai thuật toán Dijkstra**: Viết chương trình bằng Python để triển khai thuật toán Dijkstra và tìm lộ trình tối ưu từ điểm xuất phát đến điểm đích.
3. **Phân tích độ phức tạp**: Phân tích độ phức tạp của thuật toán Dijkstra trong bối cảnh của đồ thị đã cho (đặc biệt là số lượng đỉnh và cạnh).
4. **Tối ưu hóa và mở rộng**: Đề xuất các phương pháp tối ưu hóa cho thuật toán Dijkstra để cải thiện hiệu suất trong các tình huống thực tế phức tạp hơn.

### **Chủ đề 12: Tối ưu hóa mạng lưới đường điện trong một thành phố**

Một công ty điện lực cần xây dựng một mạng lưới đường điện để cung cấp điện cho các khu vực khác nhau trong một thành phố. Mục tiêu là kết nối tất cả các khu vực này với chi phí thấp nhất có thể. Mỗi khu vực được đại diện bằng một đỉnh trong đồ thị, và các kết nối giữa các khu vực (đường dây điện) được đại diện bằng các cạnh với trọng số tương ứng là chi phí xây dựng.

#### Yêu cầu đầu vào:

1. **Đồ thị**: Một đồ thị có hướng hoặc vô hướng đại diện cho các khu vực và các kết nối giữa chúng, trong đó:
   * Các đỉnh (node) là các khu vực cần cấp điện.
   * Các cạnh (edge) là các kết nối giữa các khu vực, với trọng số tương ứng là chi phí xây dựng đường dây điện.
2. **Số lượng khu vực (n)**: Một số nguyên dương đại diện cho số lượng khu vực.

#### Yêu cầu đầu ra:

1. **Mạng lưới tối ưu**: Danh sách các cạnh (kết nối) trong mạng lưới đường điện với tổng chi phí thấp nhất.
2. **Tổng chi phí**: Giá trị tổng chi phí xây dựng mạng lưới điện tối ưu.

#### Nhiệm vụ cụ thể:

1. **Mô hình hóa đồ thị**: Xây dựng mô hình đồ thị từ dữ liệu thực tế về các khu vực và các kết nối giữa chúng. Có thể thu thập dữ liệu từ bản đồ địa lý hoặc các nguồn dữ liệu địa phương.
2. **Triển khai thuật toán Prim**: Viết chương trình bằng Python để triển khai thuật toán Prim và tìm mạng lưới điện tối ưu.
3. **Phân tích độ phức tạp**: Phân tích độ phức tạp của thuật toán Prim trong bối cảnh của đồ thị đã cho (đặc biệt là số lượng đỉnh và cạnh).
4. **Tối ưu hóa và mở rộng**: Đề xuất các phương pháp tối ưu hóa cho thuật toán Prim hoặc mở rộng để cải thiện hiệu suất trong các tình huống thực tế phức tạp hơn.

### **Chủ đề 13: Tối ưu hóa lộ trình du lịch cho một nhóm khách du lịch**

Một công ty du lịch muốn lập kế hoạch cho chuyến đi của nhóm khách du lịch đến một số thành phố trong một quốc gia. Mục tiêu là tìm ra lộ trình ngắn nhất để giảm thiểu chi phí và thời gian di chuyển giữa các thành phố. Chuyến đi bắt đầu và kết thúc tại cùng một thành phố, và mỗi thành phố chỉ được thăm một lần.

#### Yêu cầu đầu vào:

1. **Đồ thị**: Một đồ thị vô hướng, trong đó:
   * Các đỉnh (node) đại diện cho các thành phố.
   * Các cạnh (edge) giữa các đỉnh đại diện cho khoảng cách hoặc thời gian di chuyển giữa các thành phố, với trọng số tương ứng.
2. **Danh sách thành phố**: Một danh sách các thành phố mà nhóm khách du lịch muốn thăm.

#### Yêu cầu đầu ra:

1. **Lộ trình tối ưu**: Danh sách các thành phố theo thứ tự mà nhóm khách du lịch sẽ thăm, bao gồm cả thành phố xuất phát.
2. **Tổng khoảng cách hoặc thời gian**: Giá trị tổng khoảng cách hoặc thời gian di chuyển của lộ trình tối ưu.

#### Nhiệm vụ cụ thể:

1. **Mô hình hóa đồ thị**: Xây dựng mô hình đồ thị từ dữ liệu thực tế về các thành phố và khoảng cách giữa chúng.
2. **Triển khai thuật toán nhánh cận**: Viết chương trình bằng Python để triển khai thuật toán nhánh cận tìm kiếm lộ trình tối ưu.
3. **Phân tích độ phức tạp**: Phân tích độ phức tạp của thuật toán nhánh cận trong bối cảnh của đồ thị đã cho (đặc biệt là số lượng đỉnh và cạnh).
4. **Tối ưu hóa và mở rộng**: Đề xuất các phương pháp tối ưu hóa cho thuật toán

### **Chủ đề 14: Phân tích mạng xã hội: Tìm kiếm thành phần liên thông trong mạng người dùng**

Trong một mạng xã hội, mỗi người dùng được đại diện bởi một đỉnh trong đồ thị, và các mối quan hệ giữa người dùng được đại diện bởi các cạnh. Mục tiêu là tìm các thành phần liên thông trong mạng này, tức là các nhóm người dùng mà mọi người trong nhóm đều có thể kết nối với nhau thông qua các mối quan hệ. Điều này giúp hiểu rõ hơn về cấu trúc của mạng xã hội và xác định các cộng đồng khác nhau.

#### Yêu cầu đầu vào:

1. **Đồ thị**: Một đồ thị vô hướng, trong đó:
   * Các đỉnh (node) đại diện cho người dùng.
   * Các cạnh (edge) đại diện cho mối quan hệ giữa các người dùng.
2. **Số lượng người dùng (n)**: Một số nguyên dương đại diện cho số lượng người dùng trong mạng.

#### Yêu cầu đầu ra:

1. **Danh sách các thành phần liên thông**: Một danh sách các thành phần liên thông trong mạng xã hội, mỗi thành phần chứa các người dùng (đỉnh) có thể kết nối với nhau.
2. **Số lượng thành phần liên thông**: Tổng số thành phần liên thông trong mạng xã hội.

#### Nhiệm vụ cụ thể:

1. **Mô hình hóa đồ thị**: Xây dựng mô hình đồ thị từ dữ liệu thực tế về người dùng và mối quan hệ giữa họ.
2. **Triển khai thuật toán tìm kiếm thành phần liên thông**: Viết chương trình bằng Python để triển khai thuật toán tìm kiếm thành phần liên thông, có thể sử dụng DFS hoặc BFS.
3. **Tối ưu hóa và mở rộng**: Đề xuất các phương pháp tối ưu hóa cho thuật toán hoặc mở rộng với các thuật toán khác để tìm các cộng đồng trong mạng xã hội.

### **Chủ đề 15: Tối ưu hóa lộ trình giao hàng cho dịch vụ vận chuyển**

Một công ty giao hàng cần xác định lộ trình tối ưu để giao hàng đến nhiều địa điểm khác nhau trong một thành phố. Mỗi điểm giao hàng có thể được coi là một đỉnh trong đồ thị, trong khi các tuyến đường giữa các điểm giao hàng là các cạnh, với trọng số đại diện cho khoảng cách hoặc thời gian di chuyển. Mục tiêu là tìm đường đi ngắn nhất từ kho hàng đến từng điểm giao hàng một cách hiệu quả.

#### Yêu cầu đầu vào:

1. **Đồ thị giao thông**: Danh sách các điểm giao hàng (đỉnh) và các tuyến đường giữa chúng (cạnh) với trọng số (thời gian hoặc khoảng cách).
2. **Điểm xuất phát**: Đỉnh bắt đầu (kho hàng).
3. **Danh sách điểm giao hàng**: Một danh sách các điểm cần giao hàng.

#### Yêu cầu đầu ra:

1. **Lộ trình tối ưu**: Danh sách các điểm giao hàng theo thứ tự mà tài xế nên thực hiện để tối ưu hóa thời gian hoặc khoảng cách.
2. **Tổng thời gian hoặc khoảng cách**: Tổng trọng số của lộ trình tối ưu.

#### Nhiệm vụ cụ thể:

1. **Mô hình hóa đồ thị**: Xây dựng mô hình đồ thị từ dữ liệu thực tế về các tuyến đường trong thành phố.
2. **Triển khai thuật toán Dijkstra**: Viết chương trình bằng Python để triển khai thuật toán Dijkstra nhằm tìm đường đi ngắn nhất từ kho hàng đến các điểm giao hàng.
3. **Phân tích độ phức tạp**: Phân tích độ phức tạp của thuật toán Dijkstra trong bối cảnh của đồ thị đã cho (số lượng đỉnh và cạnh).
4. **Tối ưu hóa và mở rộng**: Đề xuất các phương pháp tối ưu hóa cho thuật toán để cải thiện hiệu suất tìm kiếm.

### **Chủ đề 16: Tối ưu hóa mạng lưới vận tải**

Trong một mạng lưới giao thông, một số tuyến đường có thể có trọng số âm do các yếu tố như giảm giá cước hoặc chính sách khuyến mãi. Mục tiêu là xác định lộ trình tối ưu từ một điểm xuất phát đến một điểm đích, đồng thời xử lý tình huống có trọng số âm mà vẫn đảm bảo không có chu trình âm.

#### Yêu cầu đầu vào:

1. **Đồ thị mạng**: Danh sách các đỉnh (các điểm giao thông) và các cạnh kết nối chúng, bao gồm trọng số (cước phí hoặc khoảng cách, có thể âm).
2. **Điểm xuất phát**: Đỉnh bắt đầu (điểm xuất phát).
3. **Điểm đích**: Đỉnh kết thúc (điểm đích).

#### Yêu cầu đầu ra:

1. **Lộ trình tối ưu**: Danh sách các điểm theo thứ tự để đi từ điểm xuất phát đến điểm đích.
2. **Tổng trọng số**: Tổng trọng số (cước phí hoặc khoảng cách) của lộ trình tối ưu.
3. **Cảnh báo chu trình âm**: Thông báo nếu có chu trình âm trong đồ thị.

#### Nhiệm vụ cụ thể:

1. **Mô hình hóa đồ thị**: Xây dựng mô hình đồ thị từ dữ liệu về mạng lưới giao thông với các trọng số tương ứng.
2. **Triển khai thuật toán Bellman-Ford**: Viết chương trình bằng Python để triển khai thuật toán Bellman-Ford nhằm tìm đường đi ngắn nhất.
3. **Phân tích độ phức tạp**: Phân tích độ phức tạp của thuật toán Bellman-Ford trong bối cảnh của đồ thị đã cho (số lượng đỉnh và cạnh).
4. **Tối ưu hóa và mở rộng**: Đề xuất các phương pháp tối ưu hóa cho thuật toán hoặc mở rộng với các thuật toán khác như Floyd-Warshall cho các bài toán về đường đi.