**فروشنده دوره‌گرد (Traveling Salesman Problem): طراحی و پیاده‌سازی یک الگوریتم برای یافتن مسیر بهینه بین تعدادی شهر**

**شرح پروژه:**

مسئله فروشنده دوره‌گرد (**TSP**) یکی از مسائل مشهور در زمینه بهینه‌سازی است که هدف آن یافتن کوتاه‌ترین مسیر برای بازدید از تعدادی شهر است به‌طوری که هر شهر دقیقاً یک بار بازدید شود و مسیر در نهایت به نقطه شروع بازگردد. این پروژه شامل طراحی و پیاده‌سازی یک سیستم است که با استفاده از روش‌های مختلف، مسیر بهینه را پیدا می‌کند.

**ویژگی‌های پروژه:**

1. **ورودی:**
   * تعداد شهرها.
   * مختصات جغرافیایی هر شهر یا ماتریس فاصله‌ها بین شهرها.
2. **خروجی:**
   * کوتاه‌ترین مسیر بازدید از تمام شهرها.
   * طول مسیر بهینه.
3. **روش‌های حل مسئله:**
   * استفاده از الگوریتم‌های کلاسیک مانند **Brute Force** و **Dynamic Programming**
   * پیاده‌سازی یک الگوریتم تقریبی مانند **Nearest Neighbor** برای بهبود عملکرد در ورودی‌های بزرگ‌تر.
   * ارائه نتایج با مقایسه کارایی و دقت روش‌های مختلف.
4. **ویژگی‌های تکمیلی:**
   * نمایش گرافیکی مسیر بهینه (در صورت امکان(
   * قابلیت تعریف دلخواه فاصله بین شهرها (مانند فاصله اقلیدسی یا فاصله دلخواه(

**اهداف پروژه:**

1. پیاده‌سازی الگوریتم‌های مختلف برای حل مسئله TSP
2. مقایسه عملکرد الگوریتم‌ها بر اساس زمان اجرا و دقت.
3. طراحی یک رابط کاربری ساده برای تعریف داده‌ها و نمایش نتایج.

**الگوریتم‌های مورد استفاده:**

1. **Brute Force**
   * بررسی تمامی ترتیب‌های ممکن بازدید از شهرها و انتخاب کوتاه‌ترین مسیر.
2. **Held-Karp (Dynamic Programming)**
   * استفاده از برنامه‌ریزی پویا برای حل مسئله با پیچیدگی کمتر.
3. **Nearest Neighbor**
   * شروع از یک شهر و بازدید از نزدیک‌ترین شهر در هر مرحله.

**ویژگی‌های خاص:**

1. ذخیره و بازیابی داده‌های ورودی از فایل.
2. امکان افزودن یا حذف شهرها.
3. نمایش گزارش نهایی شامل:
   * ترتیب شهرها در مسیر بهینه.
   * طول مسیر.
   * زمان اجرای الگوریتم.

**موارد نمره مثبت:**

1. پیاده‌سازی گرافیکی مسیر شهرها.
2. ارائه تحلیل‌های آماری از عملکرد الگوریتم‌ها.
3. قابلیت تولید خودکار داده‌های آزمایشی (مانند تولید مختصات تصادفی(
4. بهینه‌سازی الگوریتم‌های تقریبی برای نتایج بهتر.

**ورودی نمونه:**

**روش 1: مختصات شهرها**

Number of cities: 4

City 1: (0, 0)

City 2: (2, 3)

City 3: (5, 1)

City 4: (1, 4)

**روش 2: ماتریس فاصله**

Number of cities: 4

Distance Matrix:

0 10 15 20

10 0 35 25

15 35 0 30

20 25 30 0

**خروجی نمونه:**

Optimal Path: 1 -> 2 -> 4 -> 3 -> 1

Total Distance: 80

Execution Time (Held-Karp): 15 ms

Execution Time (Nearest Neighbor): 5 ms

**مراحل پیاده‌سازی:**

1. تعریف ساختار داده‌ها برای ذخیره شهرها و فاصله‌ها.
2. پیاده‌سازی الگوریتم‌های **Brute Force**، **Held-Karp** و **Nearest Neighbor**
3. طراحی سیستم ورودی/خروجی برای دریافت داده‌ها و نمایش نتایج.
4. ارائه گزارش نهایی شامل مسیر بهینه و مقایسه الگوریتم‌ها.
5. بهینه‌سازی کد و بررسی صحت عملکرد با تست‌های مختلف.

**چالش‌ها و نکات مهم:**

1. مدیریت حافظه و زمان اجرا برای تعداد بالای شهرها.
2. اطمینان از صحت داده‌های ورودی.
3. انتخاب الگوریتم مناسب برای ورودی‌های کوچک و بزرگ.
4. ارائه خروجی کاربرپسند و قابل فهم.

این پروژه می‌تواند به‌عنوان یک تمرین عالی برای یادگیری الگوریتم‌های بهینه‌سازی و تحلیل آن‌ها مورد استفاده قرار گیرد و با افزودن امکانات پیشرفته‌تر به یک ابزار کاربردی تبدیل شود.