الغاية من الجلسة:

تطبيق خوارزمية النمل من أجل إيجاد أقصر مسافة

مقدمة:

خوارزمية النمل تحاكي عملية بحث النمل عن مصدر الطعام حيث تتحرك مجموعة من النمل متجهة نحو مصدر الطعام وتقوم بإفراز فيرمون على طول الطريق ولكن للفيرمون "زمن بقاء" يزول مع مرور الزمن يستفيد النمل من هذه الخاصية في معرفة مدى قرب مصدر الطعام كلما كانت نسبة الفيرمون أكبر كلما كان مصدر الطعام "الحل" أقرب

خوارزمية النمل تتبع نفس النهج السابق في الخوارزميتين السابقتين ولكن مع علمية تعديل للفيرمون والتي تمثل fitness عند كل عملية تحرك للنمل باتجاه مصدر الطعام.

في هذه الجلسة سنقوم بتطبيق عدة عمليات وهي:

- توليد نمل بمواضع محددة تمثل مصادر الطعام
- تطبيق تابع انتقال الى مصدر طعام
- تعديل قيمة الفيرمون

```
import matplotlib.pyplot as plt


def distance(point1, point2):
return np.sqrt(np.sum((point1 - point2)**2))
```

تابع الخوارزمية الوسطاء: عدد مصادر الطعامو عدد النملات وعدد مرات تكرار الخوارزمية ألفا وبيتا هما معاملان يتم استخدامها في حساب احتمالية اختيار مصدر الطعام التالي ضمن المسار qمعامل يستخدم في تقليل قيمة الفيرمون

```
def ant_colony_optimization(points, n_ants, n_iterations, alpha, beta, evaporation_rate, Q):


n_points = len(points)
pheromone = np.ones((n_points, n_points))
best_path = None
best_path_length = np.inf
for iteration in range(n_iterations):
paths = []
```

```
path_lengths = []
for ant in range(n_ants):
visited = [False]*n_points
current_point = np.random.randint(n_points)
visited[current_point] = True
path = [current_point]
path_length = 0
while False in visited:
unvisited = np.where(np.logical_not(visited))[0]
probabilities = np.zeros(len(unvisited))
for i, unvisited_point in enumerate(unvisited):
probabilities[i] = pheromone[current_point, unvisited_point]**alpha /
distance(points[current_point], points[unvisited_point])**beta
probabilities /= np.sum(probabilities)
next point = np.random.choice(unvisited, p=probabilities)
next_point np.random.choice(unvisited, p probabilities)
path.append(next_point)
path_length += distance(points[current_point], points[next_point])
visited[next_point] = True
current_point = next_point
paths.append(path)
path_lengths.append(path_length)
if path_length < best_path_length:
best_path = path
best_path_length = path_length
pheromone *= evaporation_rate
for path, path_length in zip(paths, path_lengths):
for i in range(n_points-1):
pheromone[path[i], path[i+1]] += Q/path_length
pheromone[path[-1], path[0]] += Q/path_length
```