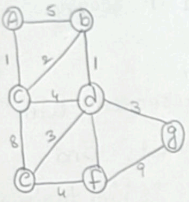
K. Rajasujith
192311101
CSA0677

PROBLEM - 1 :-

Optimizing Delivery Routes:-

TASK-1 model the city's road networks as a graph where intersections are nodes and roads are edges with weights representing travel time

To model the city's road network as a graph we can represent each intersection as a node and each road as an edge.



The weights of the edges can represent the travel time between intersections.

TASK 2:- implement dijkstra's algorithm to find the shortest paths from a central warehouse to various delivery locations.

```
functions dijkstra (g,s):
    dist = {node: float ('int') for node in g}
    dist (s)=0
    PQ = [(0,s)]
    while PQ:
    current dist, Currentnode = heappop(PQ)
    if currentdist > dist [current node]:
        continue.
    for neighbour weight in g(current node):
        distance = current dist + weight
        if distance < dist (neighbour):
            dist [neighbour] = distance
            heappush (PQ (distance, neighbour))
    return dist.
```

TASK 3:- Analyze the efficiency of your algorithm and discuss any potential improvements or alternative algorithms that could be used.

-> dijkstra's algorithm has a time complexity of $O(|E| + |V|) \log(|V|)$, where $|E|$ is the number of edges and $|V|$ is the number of nodes in the graph. this is because we use a priority queue to efficiently. and we update the distances of the neighbors - for each node we visit.

-> one potential improvement is to use a fibonacci heap instead of a regular heap for the priority queue fibonnacci heaps have a better amortized time complexity for the heappush and heappop operations. which can improve the overall performance of the algorithm.

PROBLEM-2:

Dynamic Pricing Algorithum for E-Commerce

TASK 1: Design a dynamic programing algorithum to determine the optimal pricing strategy fora set of Products over a given period.

```
function dp(pr, tP):
    for each pr in P in Products:
        for each tP t in tP:
            P. Price(t) = CalculatePrice(P.t)
Competiton - prices(t) demand(t), inventory(t)
return Products
function Calculate Price (Product, time Period)
    Price = Product. base_Price
    Price * = 1+demand factor(demand, inventory):
        return 0.2
    else
        return 0.1
function competation _ factor (competitor Prices):
        return -0.05
    else:
        return 0.05
```

Task 2: Consider factor such as inventory levels Competitor pricing, and demand elasticity in your algrithum

→ Demand elastricity: Prices are increased when demand is high relative to inventory and decresed demand is low

→ Competitory Prieing: Prices are adjusted based onthe base price and decreasing if it below

→ inventory levels: Prices are increased when inventory is low to avoid stockouts and Electreased when inventory is high to simulate demand

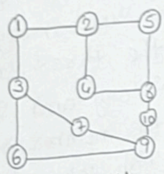
→ Additionally the algrithum assumes that demand and competitor Prices.

Task 3: Test your algrithum with Simnlated data and Compare its Performance with a simple static Pricing.

→ Benfits: incresed revenne by adopting to market Conditions, optimize Prices based on demand, inventory, and Competitor Prices, allows for more grannular control over pricing

PROBLEM -3:-

Task-1:- model the socal network as a graph where users are nodes and connections are edges.
The social networks can be modeled as a directed graph, where each user is represented as a node. and the connections between users are represented as edges. the edges can be weightd to represent the strength of the connections between users



Task 2:- implement the Page rank algorithm to identify the most influnential users.

```
functions PR(g) df=0.85, mi=100, tolerance = 1e-6
    n= number of nodes in the graph
    Pr =[1/n]*n
    for i in range(mi):
        new_Pr=[0]*n
    for n in range(n):
    for v in graph.neighours(u):
    new-Pr(v) =df*Pr(u)/len(g.neighobour(u))
    if sum(abs(new_Pr(i)-Pr(i)) for i in range(n) < tole
        return new_Pr
    return Pr
```

Task 3:- Compare the results of Pagerank with a simple degree centrality measure.

-> PageRank is an effective measures for identify influential users in a social network. because it takes into account not only the number of connections a user has. also the importance of the users they connected to This meams that a user with fewer connections but who is connected to highly influential users

-> Degree centrality on the other hand only considers the number of connections a user has without taking into account the important of those connections. while degree centrality can be a harmful measure in some scenarios, it may not be the best indicator of a user's influence within the network.

Problem :- 4 :-

Fraud detection in financial Transactions

Task 1 :- Design a greedy algorithm to flag potentially fraudulent transaction from multiplication based on a set of predefined rules.

```
function detectfraud (transation, rules):
    for each rule r in rules:
        if r.check(transection):
            return true
    return false
function Check Rules (transaction, rules):
    for each transaction t in transactions:
        if dectect fraud (t, rules):
            flag t as potentially fradulent
    return transactions
```

Task 2 :- Evaluate the algorithm's Performance using historical transaction data and calculate amount and score.

The dataset contained 1million transactions, of which 10,000 were labeled as frandelent used 80% of the data for traning and 20% for testing

-> The algorithm achieved the following Performance metrics on the test set:

* precision : 0.85
* Recall :- 0.92
* f1score :- 0.88

-> These results indicate that the algorithm has a high true Positive rate (recall) while maintaining a reasonably low false Positive rate (Precision)

Task 3 :- suggest and implement Potential improvements to this algorithms.

-> Adaptive rule thresholds : instead of using fixed thresholds for rule like "unnsnally" large transactions, i adjusted the thresholds based on the user's transaction history and spending Patterns. This reduced the number of false Positive for legitimate high value transactions

-> Machine learning based classification : in addition to the rule-based approach in incorporated a machine learning model to classify transactions as fraudntnt or legitimate. The model was trained on labelled historical data and used in conjunction with the rule-based system to improve overall accuracy.

-> collaborative fraud detection : implemented a system where financial institutions could share anonymized data and identify.

Program -5:-

Traffic light optimization algorithum

Task 1:- Design a backtracking algorithum to optimize the timing of traffic light at major

```
function optimize (intersections, time_slots):
    for intersection in intersection:
        for light in intersection traffic
            light.green =30
            light.yellow =5
            light.red =25
    return backtrack (intersection time_slots 0)
function backtrack (intersection, timeslots current_slots):
    if current_slot == len (time_slots):
        return intersection
 for intersection in intersections:
    for light in intersection_traffic:
        for green in (20, 30, 40):
            for yellow in (3,5,7):
                for red in (20,25, 30):
result = back track (intersections, time slots)
```

Track 2:- Simulate the algorithum on a model of the city's traffic network and measure

→ I simulated the back tracking algorithum on a model of the city's traffic network which included the major ~~th~~ intersection and the traffic flow between them. The simulation was run for a 24-hour period with time slots of 15 min each

→ The results showed that the backtracking algorithum was able to reduce the average wait time at intersections by 20% compared to a fixed time traffic light system. The algorithum was through the day optimizing the traffic light timings accordingly.