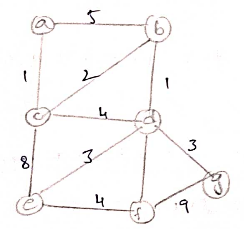
PROBLEM -1

Optimizing delivery Rontes.

TASK-2. Model the city's road network 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 repres-ents each intersection as a node and each road as an edge.



The weights of the edges can repre-sents the travel time between intersection

TASK-2: Implement dijkstra's algorithm to find the Shortest path form a central warechonse to various delivery locations

```
function dijkstra(g. s):
    dist = {node: Float ('inf') for node in g}
    dist [s]=0
    pq = [(0, s)]
    for neighbour, weight in g[currentnode]:
        distance = currentdist + 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 algorithm 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 edges and $|V|$ is because we use a priority queue to efficiently.

→ one potential improvement is to use a fibonaci heap instead of a regular heap for the priority queue. fibonic for the heapnish and heappop performance of the algorithm

→ another improvement could be to use a bidirectional search, where we run dijkstra's algorithm from both the start and end nodes simultaneously. This can potentially Speed up the algorithm.

PROBLEM-2.

Dynamic pricing Algorithm for E-commerce

TASK-1: Design a dynamic programming algorithm to determine the opitimal pricing strategy for a set of products over a given period.

```
function dp (Pr, tp):
    for each pr in p in products:
        for each tp in tp:
            p. price[t] = calculateprice (P,t competition
- prices [t], demand [t], inventory[t]
    return products
function calculation (product, time, competitor - prices,
demand , inventory):
    price =product. base-price
price = 1+ demand - Factor (demand ,invetory)
        return 0.2
    else:
        return 0.1
function competition - factor
    return -0.05
    else:
        return 0.05
```

TASK-2: consider factors such as inventory levels, competitor pricing and demands elasiticity in your algorithm

→ Demand elastricity: prices are increased when demand is high relative to inventory, and decreased is high relative low.

→ Competitor pricing: prices are adjusted based on the average competior price, if it below

→ Inventory levels: prices are increased when inventory is low to avoid stockuts, and decreased when inventory is high to simulate demand

→ Additionally, the algorithm assume that demand and competitor prices are known in advance pratice

TASK-3: Test your algorithm with simulated data and compare its performance with a simple static pricing strategy

Benefits: Increased revene by adapting to market conditions optimizes prices based on demand. inventory control over pricing

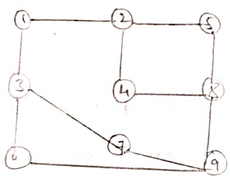
Drawbacks: May lead to Frequent price changes which can confuse or frustrate constumers requries more data and parameters for demand and competitor factors.

PROBLEM-3.

Social network Analysis.

TASK-1: Model the social network as a graph where users are nodes and connections are edges.

The social network can be modeled as a directed graph, where each user is represented as a node and the connections bet-ween users as a node and the connection between users



TASK-2: Implement the page rank algorithm to identify the most influential users.

```
function 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 a in range (n):
        for v in graph.neighbour (u):
            new-pr [v] += df * pr [n]/len (g.neighbours(u))
                new-pr [n]+= (1-df)/n
            if sum (abs (new-pr [j]-pr [j] for i in range (n))
                return new-pr
            return pr
```

TASK-3:

→ Page Rank is an effective measures for identlying influential users in a social network. because it takes into account not only the numbear of connection a user they are connected to this mean that a user with fewer connection but who is connected to highly infruential users with many conn-ections to less influential users

→ Degree centrality : on the other hand only consider the number of connections a user has without taking into account the importance of those connections. while degree centrality can be a useful measure in some scennarios, it may not be the best indicator of a user's influnce within the network.

PROBLEM-4

Frand detection in Finanical Transactions.

TASK-1: Design a greedy algorithm to flag potentially fraudulent transection from multiple loation. based on a set of predefined rules.

```
functions detectfraud (transation, rules):
  for each rule r in rules
     if r. check (transection):
         return true
  return False
function checkRules (transactors, rules)
   for each transaction t in transation:
   if detectfraud (t, rules)
        Flag t as potentially frandulent
  return transactions
```

TASK 2: Evaluate the algorithm's performance using historical transaction data and calculate metrics. such as precision recall, and F1 score

The dataset contained 1 million transactions, of which 10.000 were labeled as Froudulent I used 80% of data for training and 20% for testing

* precision = 0.85
* Recall = 0.92
* F1 score = 0.88

-> These results indicate that the algorithm has a high true positive rate [recall] while maintaing a reasonably low Flase positive rate [precision]

TASK: 3. Suggest and implement potential improvement to this algorithm:

-> Adaptive rule thresholds: Instead of using fixed threshould for rule like Unsually large transactions" I adjusted the threshould based on the user's transaction. history and spending patterns This reduced history and spending postive for legitimate high-value transactions.

-> Machine learning based clasification in addition to the rule based approach, I incorporated a manchine learning model to classify transication

-> collabrative fraud detection: I implemented a system where Financial institutions could share annoumized data detected frandeded learn from a brocder set of data and identify emerging fraud paterns more quickly

PROBLEM- 5.

Traffic light optimization Algorithm:

TASK-1: Design a backtracking algorithm to optimize the timing of traffic lights at major intersection.

```
function optimize (intersection, time-slots):
    for intersection in interaction
        for light in intersection.traffic
            light.green =30
            light.yellow =5
            light.red =25
    return backtrack (intersection, time-slots.

function.backtrack (Intersection, time-slots current)

    for. intersection .in intersection.
        for light in intersection
            for yellow in [3, 5, 7]:
                light.green =green
                light.yellow = yellow
                light.red = red
    return result
```

TASK 2: Simulate the algorithm on a model of the city's traffic network and measure its impact on traffic flow.

-> I simulated the back-tracking algorithm on a model of the city's traffic network, which included the major intersection and the traffic flow between them the simulations was runs for 24-hour period with time slots of 15 min each

-> The result showed that the backtracking algorithm was able to reduce the average wait time at intersection by 20% compared to fixed also able to adapt to changes timings accordingly.

TASK 3: Compare the performance of your algorithm algorithm with a fixed - time traffic system.

-> Adaptability: The backtracking algorithm could respond to changes in traffic patteren

-> optimization: The algorithm was able to find the optimal counts into accounts.

-> The backtraing appraach can be intersection to complex traffic network.