PURCHASER PROBLEM

Group:9

Submitted by-
Tapomay Singh Khatri 19ucs064
Rajan Agarwal 19ucs039
Raghav Singhal 19ucs022
Hemang Goyal 19ucs193
Hardik Bhati 19ucs024

ABSTRACT

In this project we are trying to solve the travelling purchaser problem by dividing it into sub-problems and thinking about each sub-problem separately. We are trying to find an efficient algorithm for these problems .

Keywords :

- Travelling Purchaser Problem
- Depth First Search (DFS)
- Breadth First Search (BFS)

INTRODUCTION

The Traveling Purchaser Problem (TPP) has been one of the most studied generalizations of the Traveling Salesman Problem and transportation problem, the TPP has attracted the attention of both researchers in combinatorial optimization and practitioners in recent decades. The problem has been used to model several application contexts and is computationally challenging, dealing at the same time with the supplier's selection, the optimization of the purchasing plan and the routing decisions of the purchaser.

Algorithms Used

DFS:

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. So the basic idea is to start from the root or any arbitrary node and mark the node and move to the adjacent unmarked node and continue this loop until there is no unmarked adjacent node. Then backtrack and check for other unmarked nodes and traverse them. Finally, print the nodes in the path.

BFS:

Breadth first search is a graph traversal algorithm that starts traversing the graph from the root node and explores all the neighbouring nodes. Then, it selects the nearest node and explores all the unexplored nodes. The algorithm follows the same process for each of the nearest nodes until it finds the goal.

The algorithm of breadth first search is given below. The algorithm starts with examining the node A and all of its neighbours. In the next step, the neighbours of the nearest node of A are explored and the process continues in the further steps. The algorithm explores all neighbours of all the nodes and ensures that each node is visited exactly once and no node is visited twice.

Dijkstra:

Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph.Dijkstra's Algorithm works on the basis that any subpath B -> D of the shortest path `A -> D` between vertices A and D is also the shortest path between vertices B and D. Dijkstra used this property in the opposite direction i.e we overestimate the distance of each vertex from the starting vertex. Then we visit each node and its neighbors to find the shortest subpath to those neighbors.
The algorithm uses a greedy approach in the sense that we find the next best solution hoping that the end result is the best solution for the whole problem.

PROBLEM STATEMENT

You are initially at your home with a list of things that you need to buy from the market, now you know a different number of shops that sell different products at different or same price.

Constraints:

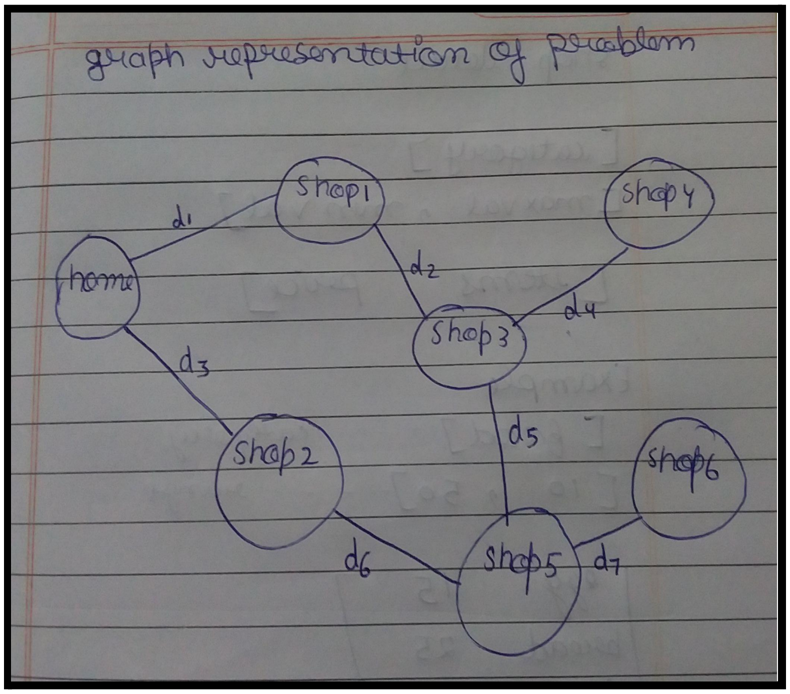
- All shops are at specific locations with weighted routes.
- Problem location is an ideal world without any friction or traffic or other delimiters.

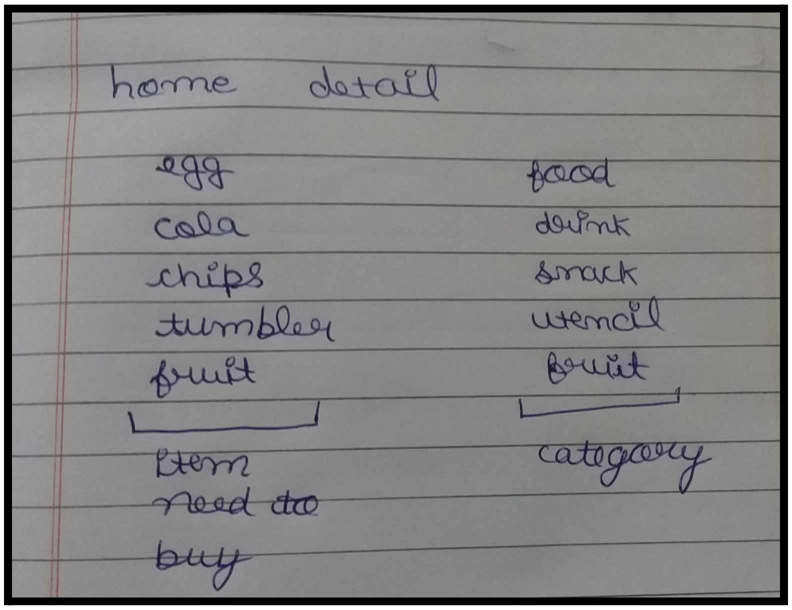
Now you need to know following things:

- Is it possible to buy everything on a list ?
- What is the minimum distance you need to travel to buy most things ?
- What is the minimum (sum total) price in which you can buy most things ?
- What is the best value we can get considering both distance and price together to buy all things ?

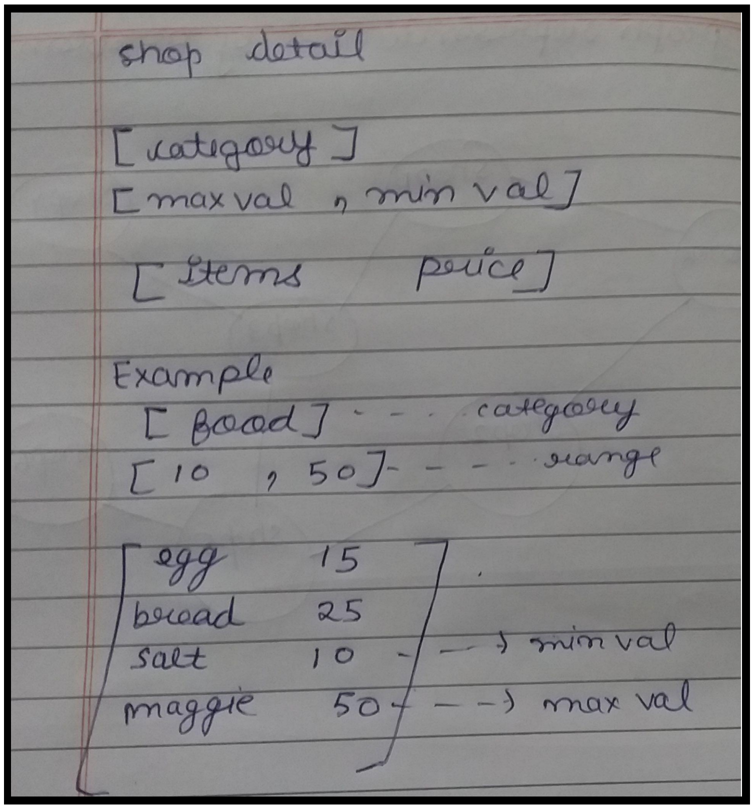
Pre-analysis

Here shops and homes are represented as nodes of graph and each edge of graph is a weighted edge which represents distance between nodes. We also take a distance bound upto which we have to search.




home detail

egg	food
cola	drink
chips	snack
tumbler	utencil
fruit	fruit
item need to buy	category



shop detail

[category]
[max val , min val]

[items price]

Example
[food] - - - - category
[10 , 50] - - - - range

[egg 15
bread 25
salt 10 - - -> min val
maggie 50 - - -> max val]

PROPOSED SOLUTIONS

1. To get a solution for this, what we do is to assign each item on the list which we need to buy a category and each shop would also have a predefined category that shows which kind of item it sells. Now we will start traversing[dfs] through our shops [like a connected graph] ,furthermore we could also define a bound[distance] that defines the maximum distance to which we could travel to buy things on the list.

2. First, we will apply a brute force method to search all paths upto distance bound. First we will check the maximum(or all) items possible to buy on the list. Then, we will check the minimum distance

from the home; then, comparing the distances, we will find the minimum required distance.

3. We would start traversing through our shops as proposed in [2.] , and traverse all paths until the bound. Then, firstly, we will compare how many items we could get in each path and give priority to the path that has more items. Then, we will compare the total sum of the price of items and choose the least one.

4. We are using a greedy function which will decide which path we would select at each step. But this approach does not always give good results . Our greedy algorithm first checks the minimum distance from the current node. Then, if the price of the item at the shop is less than the existing price, we update the price. We keep iterating this process until the constraint is reached. Then; we return the sum total of the price and the distance. NOTE : Our greedy algorithm gives precedence to distance.
   function(x)= min(distance shop)

RESULTS AND OBSERVATIONS

We have observed that solving all parts together means TPP is a NP hard problem but if we try to solve each problem individually and use some constraints we could add some optimality.
Let shops=s, edges=e, list items=n, max(each shop item number)=k, distance bound=d .
Worst case time complexities for some parts:
1) O((s+e)*k*n)
2) Non-polynomial
3) Non-polynomial
4)O((s+e)*(e)*(k*n))

NOVELTY

Some problems like travelling salesman problem, Chinese postman problem, and Aarogya setu app are some real life projects or problems that are similar to this project theme, but, in Travelling Salesman Problem, our aim was only to visit every node exactly once in minimum distance; in Chinese Postman Problem, the shortest path is found and initial and final nodes are same; and Aarogya Setu app finding nearest vaccine center is similar only to second point of our project theme .

CONCLUSION

- The Purchaser Problem is an NP-Hard Problem.
- The great interest around the Purchaser problem is probably due to the fact that it challengingly combines shops and product purchase planning and all this in minimum time. It is clear that optimally solving each subproblem separately does not guarantee to achieve the optimal solution for the problem.
- Our problem has a bi-objective nature, linearly combining in a single objective function the minimization of both traveling and purchasing costs. On the one hand, the traveling costs optimization pushes the customer to select only shops that are strictly necessary to satisfy product demand and on the other hand, the purchasing costs minimization pushes to select more convenient shops.

REFERENCES

- Manerba, D., Mansini, R., & Riera-Ledesma, J. (2017). The Traveling Purchaser Problem and its variants. *European Journal of Operational Research*, *259*(1), 1–18. https://doi.org/10.1016/j.ejor.2016.12.017
- Wikipedia contributors. (2019, January 14). *Traveling purchaser problem*. Wikipedia. https://en.wikipedia.org/wiki/Traveling_purchaser_problem
- GeeksforGeeks. (2021, September 11). *Dijsktra's algorithm*. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
- Wikipedia contributors. (2021, November 17). *Travelling salesman problem*. Wikipedia. https://en.wikipedia.org/wiki/Travelling_salesman_problem
- *Transportation problem: Set 1 (introduction)*. GeeksforGeeks. (2019, November 25). Retrieved November 25, 2021, from https://www.geeksforgeeks.org/transportation-problem-set-1-introduction/.
- Vehicle routing problem. (2021, August 30). Retrieved from https://en.wikipedia.org/wiki/Vehicle_routing_problem