Name: Michael-S
Roll No: 417
Div: D

PAGE NO.:
DATE: / /

Graph Case Study

Objective: Usage of appropriate data structures & algorithms in designing a graph problem.

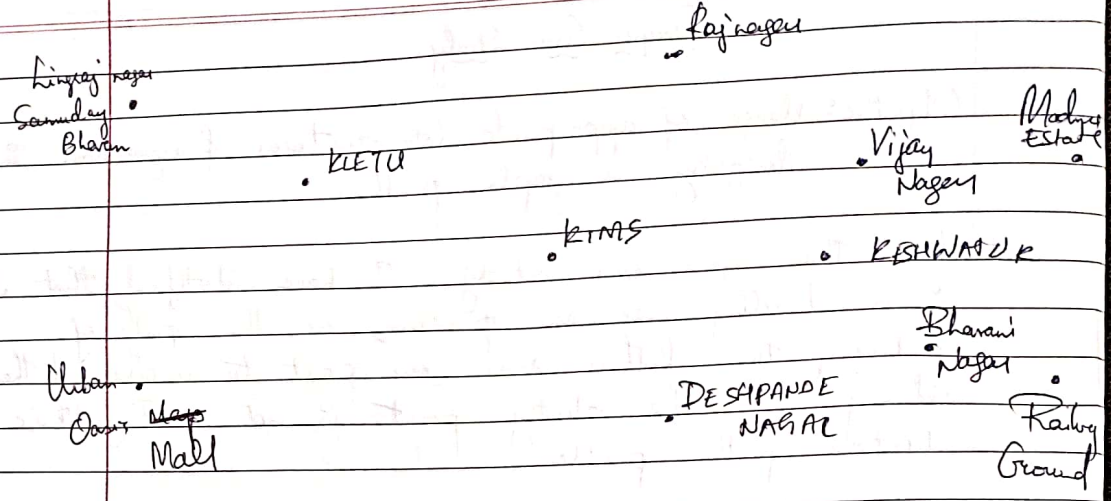
* From the given case study, I have identified that it is a travelling salesman problem, as the path of delivering the letters from one point to another & then returning to the starting point is not a Dijkstra's shortest path problem.

* So, to identify the shortest possible path of delivering the letters, I'll frame the cost matrix of the given matrix & further analyse the problem.

* For the construction of the cost-matrix, I have made few assumptions & noted down few constraints ↓

1) Elf shall be travelling by car to deliver the invitation.
2) The cost incurred will be the time taken for travel from one point to another.
3) I have found out the time required by Google Maps, which also has a constraint, i.e., The best/shortest path found out is based on the traffic conditions at a particular time, In my case at around 11:30 am, where the traffic is maximum.
4) So, the route may/may not be the shortest path at a given point time, so I am assuming a stand still condition wherein the cost at that particular time will be considered.
5) Also, I have considered the places to be isolated from each other, meaning when I travel from one point to another, the other places will be not be traversed upon. (This is an issue faced by me to distinguish places)

* Graph of graph ↓



(Not to Scale & not drawing edges due to complexity)

* Cost matrix (in minutes)

	KL	LN	UO	KI	DN	KE	VN	RN	BN	RG	ME
KLETU	0	4	10	5	8	6	8	9	10	9	12
Lingaj Nagar	4	0	11	12	17	15	17	13	19	19	21
Urban Oasis	10	11	0	12	12	13	15	18	14	14	17
KIMS	5	12	12	0	8	6	8	10	9	8	13
Deshpande	8	17	12	8	0	7	9	10	6	5	9
Keshwapur	6	15	13	6	7	0	2	7	7	5	6
Vijay Nagar	8	17	15	8	9	2	0	7	6	6	6
Raj Nagar	9	13	18	10	10	7	7	0	10	10	10
Bharani Nagar	10	19	14	9	6	1	6	10	0	2	6
Railway G	9	19	14	8	5	5	6	10	2	0	6
Madhva Estate	12	21	17	13	9	6	6	10	6	6	0

* My inferences from the cost matrix

⇒ If I notice the places given, they are actually areas, as in Keshwapur is a huge area. So, from madhva estate to Keshwapur [Ex: My school St Mary's] is nearer than going to Railway ground, but the time taken for both is 6 minutes.

⇒ So is from KLETU or KIMS to corresponding areas.

⇒ Deshpande Nagar, Keshwapur, Vijay Nagar, Raj Nagar & Bharani Nagar are areas, so the google maps

Scanned with CamScanner

PAGE NO.:
DATE: / /

Pin down to a particular point in the map of that area, so my findings are according to it

* There are also many connecting & intersection roads/streets which further increase the complexity of the graph.
* And it isn't a straight path as seen from other graph examples. Real time involves curves & ebbs & flows to the traversal.
* So, the cost matrix is constructed based on above constraints & assumption.

* Designing of solution :-

From the problem, the cost of travelling back to same spot is allowed, but that would make no sense for Eli to go back & move ahead. Normally, we go randomly to one place to another, but we don't return to the spot (either we were in a loop (or) we lost our way (or) we forgot something (or) we sketch out the traversal by going back as that is the only way we know)

So, I'll try to go to each place exactly once, create a possible minimum cost & further do analysis.

⇒ It is a kind of Hamiltonian cycle of minimum weight wherein we traverse through each node exactly once.

⇒ There are few conditions we need to satisfy for it i.e. many edges which we'll have, from each vertex to every other vertex, so we'll be able to find a Hamiltonian cycle.

⇒ Also sum of the degrees of any two non-adjacent vertices is greater than or equal to n (vertices).

* We'll move forward,

PAGE NO.:
DATE : / /

* So, the code may lead to an incomplete Hamiltonian cycle, as in case of many nodes.
* Also it selects the best optimum route available without thinking of the future route.

* So, we can compute the solutions of all the sub-problems starting with the smallest. Whenever computing a solution requires solutions for smaller problems using recursion. For TSP, we cannot know which subproblems to solve, so we solve them all.

* Kinda Dynamic Programming to be employed, so using the bidirectional graph & cost matrix that is symmetric.
1) We'll start from KLETU (given), so we KLETU to all the points we'll analyse (non-visited vertices). The main problem is split into smaller problems.
2) We take from KLETU to each point & find the weight of it & then we taken the minimum of the further traversal as in consider 1, 2, 3, 4.
1→2 = 3, 2→3 = 5, 3→4 = 6, 2→4 = 3, 4→3 = 5
∴ A(1, {2,3,4}) = consider 1 to 2 & further
= {(1,2) + Min[A(2, {3,4})]
= 4 + Min[(5+6) or (3+5)]
= 4 + 8
= 12 & we do so on for 3 & 4

⇒ We'll take the minimum of the sub problems. Consider (1,2) is min so now we'll see further A(2,(3,4)) & so on same process is continued

⇒ We are splitting the main prob into sub problems & solving to get answer (It is kinda brute force but better than brute force (n-1)! ways).

⇒ Once we reach A(4,{}), it will be the reaching base condition in recursion which returns 0 (zero distance).

PAGE NO.:
DATE : / /

* Also we'll have to add the weight of the path to reach that particular distance to the source i.e if $1 \to 2 \to 3 \to 4$ has 10 cost & $4 \to 1$ has 1 cost 1 has to be added in the end to reach that starting part i.e $1 \to 2 \to 3 \to 4 \to 1 = 10+1 = 11$ final cost.

* Do the recursive Equation (found online)
$T(i,S) = \min((i,j) + T(j, S-\{j\}))$; $S \neq \phi$; $j \in S$;
S is set that contains non-visited vertices.
$= (i,i)$; $S \neq \phi$ This is base condition for recursive equation.

Here,

* $T(i,s)$ means we are travelling from a vertex "i" and have to visit set of non-visited vertices "S" and have to go back to starting point.

* $(i,j) \Rightarrow$ cost of path from node i to j

* From Equation, we find cost to all the nodes (i,j) and from that node to remaining using recursion $(T(j, S-\{j\}))$.

* Also, it is not guaranteed that every vertex is connected to other vertex (In mine only to itself is 0, rest are connected). So cost will be MAX = 9999 (as min won't exceed it).

* If S is empty that means all the nodes are visited & we take the distance of the last node to starting node & finish the process.

* Time Complexity

Dynamic Programming → Sub-problems

PAGE NO.:
DATE: / /

After reaching i^{th} node find the next min. distance to that i^{th} node is a sub-problem

$\Rightarrow$ Solve by recursive we get total $(n-1)2^{(n-2)}$ Sub problem (excluding 1st vertex, excluding next source)

$\therefore O(n \cdot 2^n)$

$\Rightarrow$ Each sub-problem will take $O(n)$ time (find path to the remaining $(n-1)$ nodes)

$\Rightarrow$ $\therefore$ Total Time complexity is $O(n \cdot 2^n) * O(n) =$
$O(n^2 \cdot 2^n)$ // Not good but is TSP

$\Rightarrow$ Space complexity same. (Recursion laughing behind ~~at~~ your ~~the~~ back)

* Thus, this my analysis, took help of google, in terms of map and ideas, also I crossed over few other possible strategies that I'll analyse forward :

1) Back Tracing
2) Branch & Bound Algorithm
3) Christofides Algorithm.

And find any other possible solution in the future (if possible). For now. That's All Folks.

P.S: Sorry for the delay in submission sir, went through some issues but tried to complete it as best as I could.
Thank you for your labs & your guidance. Hopefully next sem you'll take our OS subject. Thank you once again.