ASSIGNMENT

Open-Ended Problem (OEP) ADA

NAME :: ANAND DASANI

ENROLLMENT NO :: 190180107012

DATE :: 05/10/2021

SUBJECT :: ADA

Reference:: Cormen, Wikipedia

BRTS Route Design Problem

Problem Statement

Bus Rapid Transit System is a very popular system of public transport that takes people on dedicated routes from major city spots to distant locations.

A thorough analysis is necessary to implement this system because there is a huge infrastructure cost for making dedicated lanes, bus stops, signaling technology, and of course the bus itself. If you are given this problem and you are supposed to find an optimal solution where your algorithm provides optimal route design, what kind of input parameters will you require? How many of such specific arguments can be processed by your algorithm? How will you define the optimal solution?

As Per My Observation

This Problem is an optimization problem where objectives are defined, its constraints are determined, and a methodology is selected and validated for obtaining an optimal solution.

Based on this description of the problem, I have proposed a three-layer structure for organizing the problem approach.

1. objectives
2. Parameters
3. Methodology

1. **Objective:** The "Objectives" layer incorporates the goals set when designing a public transportation system such as the **minimization of the costs** of the system or the **maximization of the quality of services** provided.

2. **Parameters:** The "Parameters" layer describes the operating environment and includes the design variables expected to be derived for the transit network route layouts, frequencies as well as environmental and operational parameters affecting and constraining that network.

3. **Methodology:** Finally, the "Methodology" layer covers the **logical**-mathematical framework and algorithmic tools necessary to formulate and solve the problem.

My AIM while solving this problem (If I am given this problem and I am supposed to find an optimal solution where my algorithm provides optimal route design) will be as follow:

1. **User benefit maximization**
2. **Operator cost minimization**
3. **Total welfare maximization**
4. **Capacity maximization**

To Make it MORE optimal I will also include some constraints like:

1. **Passenger Maximization**
2. **Delay Minimization** (less than 30 mins.)
3. **Each Region must be approachable**
4. **Fleet Maximization**

SOLUTION

This is an NP-Hard Problem

Reason:

Optimization Problems – An optimization problem asks, "What is the optimal solution to problem X?"

– Examples:

- 0-1 Knapsack
- Fractional Knapsack
- Minimum Spanning Tree
- Decision Problems

A decision problem - is one with a yes/no answer

– Examples:

- Does graph G have an MST of weight ≤ W?

Optimization/Decision Problems

- An optimization problem tries to find an optimal solution
- A decision problem tries to answer a yes/no question
- Many problems will have a decision and optimization versions – Eg: Traveling salesman problem
- optimization: find the hamiltonian cycle of minimum weight • decision: is there a hamiltonian cycle of weight ≤ k

– **Polynomial-time**: O(n2), O(n3), O(1), O(n log n)

– **Not in polynomial time**: O(2n), O(n n), O(n!)

What does NP-hard mean?

A lot of times you can solve a problem by reducing it to a different problem.

I can reduce Problem B to Problem A if, given a solution to Problem A, I can easily construct a solution to Problem B. (In this case, "**easily**" means "**in polynomial time**.“).

- A problem is NP-hard if all problems in NP are polynomial-time reducible to it, ...
- Every problem in NP is reducible to HC in polynomial time. Ex:- TSP is reducible to HC. Example: lcm(m, n) = m * n / gcd 13 (m, n)

SAME WITH THIS PROBLEM, IF WE CONSIDER ALL THE POSSIBLE MODULES THEN THIS PROBLEM IS NOT POSSIBLE TO SOLVE IN POLYNOMIAL TIME IT WILL TAKE EXPONENTIAL TIME.

ADDRESSING THE QUESTIONS

Q1. Does only one design strategy efficiently solve this problem?

No !!, Because Many Optimization Algorithms will be required for the different modules.

Q2. Are we using any data structures?

For single modules like Connecting all Major Areas including Residential and mixed public stops, Yes!! we are using **Graph** Data Structure to represent the **Stops as Node** and **Road as Vertices,** Weights of the graph will be the Cost for reaching the particular destination from the source.

Q3. For the solution to work with an increase in values, what comes out to be the order of growth for my algorithm?

I will be using **Floyd-Warshall Algorithm** for the above particular module Because it will find All Pair Shortest Path from One Location to another Location.

So the order of growth of my algorithm will be **Theta(n3).**

Q4. Am I proving the order of growth for my algorithm using proper mathematical notations?

I think Theta is a perfect notation for this particular problem because it will take n3 time exactly.

Q5. Can I reduce this problem or part of the problem to any solved computational problem in polynomial time?

Yes!! single modules - Connecting all Major Areas including Residential and mixed public stops, we can use **Floyd-Warshall Algorithm** which will take polynomial time and the part of the whole BRTS Problem can be reduced to polynomial time.

Q7. Have I identified properties like loop invariants in my algorithm to prove its correctness?

Partially I Have identified the loop invariants in the Floyd-warshall algorithm.

Q8. Am I able to provide the order of growth in BigOh or theta?

Yes, I have provided the order of growth for part of the BRTS problem in theta.

Q7. What is the growth in Omega for my algorithm?

Omega(n3).

Below is the General Algorithm for Floyd-Warshall Algorithm

FLOYD WARSHALL ALGORITHM

```
let dist be a |V| × |V| array of minimum distances initialized
to ∞ (infinity)

for each edge (u, v) do
    dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
for each vertex v do
    dist[v][v] ← 0
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j]
                dist[i][j] ← dist[i][k] + dist[k][j]
            end if


let dist be a |V| X |V| array of minimum distances initialized
to ∞
let next be a |V| X |V| array of vertex indices initialized to
null

procedure FloydWarshallWithPathReconstruction() is
    for each edge (u, v) do
        dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
        next[u][v] ← v
    for each vertex v do
        dist[v][v] ← 0
        next[v][v] ← v
    for k from 1 to |V| do // standard Floyd-Warshall implement
        for i from 1 to |V|
            for j from 1 to |V|
                if dist[i][j] > dist[i][k] + dist[k][j] then
                    dist[i][j] ← dist[i][k] + dist[k][j]
                    next[i][j] ← next[i][k]


procedure Path(
u, v)
    if next[u][v] = null then
        return []
    path = [u]
    while u ≠ v
        u ← next[u][v]
        path.append(u
```