ROUTE QUEST

Optimisation - Logistics Unleashed

Link to test cases

Problem Statement

You are tasked with planning the route for a single delivery vehicle operating in a city. The vehicle is responsible for delivering N packages — each package starts at a delivery hub and must be delivered to a unique destination house.

The city is represented as an undirected graph where:

- Nodes are locations (hubs, houses, roads, fuel stations, or empty intersections)

- The undirected edges represent roads with fuel costs

- The vehicle has a fuel tank of limited capacity F

- Fuel stations are scattered throughout the map, and visiting one refuels the tank to full

Your Task

Plan a single route (a sequence of nodes) that:

1. Visits each house at least once

2. Delivers each package only after visiting its corresponding hub at least once

- For each (hub_i, house_i) pair: the last visit to house_i must come after at least one visit to hub_i

3. Never exceeds the fuel tank limit

- Fuel must never drop below 0

- Refueling is only allowed at designated fuel station nodes

Input Format

N T M K F

H1 H2 ... HN

D1 D2 ... DN

S1 S2 ... SK

u1 v1 c1

u2 v2 c2

...

uM vM cM

- N = number of deliveries
- T = number of nodes in the graph(can be a hub, house, fuel station or nothing)
- M = number of roads
- K = number of fuel stations
- F = fuel tank capacity
- H1..HN: indices of delivery hubs
- D1..DN: indices of houses
- S1..SK: indices of fuel stations
- Next M lines describe roads:
- Each line has u v c: an undirected edge between node u and v with fuel cost c

All node indices are between 0 and some T-1, where T is the total number of unique nodes appearing.

Output Format

L

n1 n2 n3 ... nL

- L = length of the route (number of nodes visited)

- Second line: L space-separated node indices representing the full route of the vehicle

Route Rules

- You may start and end at any node

- Nodes (including hubs, houses, and fuel stations) may be visited multiple times

- Refueling to full is allowed only at fuel station nodes

- You must only traverse edges given in the input

- The vehicle starts with a full tank

Delivery Constraints

Your route is considered valid only if:

- Every house D_i is visited at least once

- Before the last occurrence of D_i, there exists at least one occurrence of H_i in the route

- Fuel must never drop below zero at any point

- Only listed edges may be used

Example Input

2 6 5 1 10

0 3

4 5

2

0 1 4

1 2 3

2 3 2

1 4 5

3 5 3

Example Output

9

0 1 2 3 5 3 2 1 4

Scoring

Your code should run within a reasonable time, <10 mins per test case. Since the running time depends on the system, try to keep ~50% leeway. Also, your code should include a readme file with instructions to run.

There are three test cases of varying complexity. The lower complexity cases will have low fuel cost, so there’s more incentive to perform better on the high complexity one.

Your score will be evaluated using a hybrid scoring system:

Route Efficiency - 70%- Based on the total fuel cost of your route, compared to the best submission

Approach and Clarity - 30% - Based on code quality, novelty of approach, and clarity of logic

1. Route Efficiency (70%)

Let:

- C = your total fuel cost across all three test cases

- B = best fuel cost across all valid submissions

Your efficiency score is:

Efficiency = (1 / (1 + (C - B))) × 100

If any of your routes is invalid, your efficiency score is 0.

2. Approach and Clarity (30%)

Criteria include:

Correctness under constraints - 10

Efficient algorithm use - 10

Code readability and documentation - 5

Novelty or optimization idea - 5

The clarity score is normalized to 30% of the total test case score.

Final Score

Final Score = 0.7 × EfficiencyScore + 0.3 × ClarityScore

Example:

- Efficiency Score = 85

- Clarity Score = 25 out of 30

Final = 0.7 × 85 + 25 = 59.5 + 25 = 84.5