Contact Graph Routing

In a network with mobile nodes, the vertices of a graph keep moving. Links are established or abolished over time. A full path from a source node S to a destination node D may never exist, but it may still be possible to go from S to D in multiple "hops" with possible delay after each hop (i.e. go to an intermediate node, wait until a new link is established, go to another intermediate node, …and so on). Classical graph representation of such a network cannot be used to find best paths, since such a graph keeps changing continuously.

Such a time-varying graph can be represented by a "contact graph" $G = (C, E)$ defined as follows:

- Each vertex $c_i \in C$ represents a contact between two network nodes (sender and receiver of this contact)
- Sender and receiver nodes of each contact are stored in two attributes of a vertex: $c_i.snd$ and $c_i.rcv$
- The start and end time of each contact are also stored in two attributes of a vertex: $c_i.start$ and $c_i.end$
- The transmission delay of a contact is stored in the $c_i.owlt$ attribute

Note that actual network nodes are NOT graph vertices in a contact graph; they are only stored as sender and receiver attributes of the vertices. An edge between two vertices c_i and c_j represents a wait before the next contact is used.

See sections 3 and 4 of the paper "Routing in the Space Internet: A contact graph routing tutorial" for a more detailed explanation of contact graph routing. For this project you will not need the *rate*, *volume*, *MAV*, *suppr*, *suppr_nh*, *fbtx*, *lbtx*, *lbrx* and *EVL* attributes mentioned in the paper so you can safely ignore those.

The goal of this project is to implement the Dijkstra algorithm on contact graphs as explained in Algorithm 1, Algorithm 2, and Algorithm 3 in section 4 of the paper. Dijkstra tries to find a valid path in the contact graph minimizing the "arrival time" as its metric (which corresponds to the "path cost" in regular Dijkstra).

The input for the project is a contact graph represented in file "ContactList.txt". The graph in the file contains 190 contacts (with IDs 1 through 190) among 12 network nodes (numbered 1 through 12). Each line in the text file specifies the id, start time, end time, sender, receiver, and owlt attributes of a contact.

Your program should perform the following tasks:

- Read the contact graph info from the file.
- Implement a minimum priority queue using a heap, which will be used in Dijsktra.
- Implement Dijkstra as outlined in section 4 of the paper, but you must use your min priority queue instead of the linear search used by the CSP() function in the paper.
    - You don't have to include the parts in provided algorithms dealing with suppr, suppr_nh and MAV.
    - Assume $owlt_{mgn} = 0$, so no need to include this either.
- Use your algorithm to find an optimal path from node #1 to node #12.
- Print the contact ids corresponding to the optimal path and the resulting best arrival time.

You must submit the following by the project deadline:

- All of your code
- Clear comments included in the code
- Output showing the optimal path and the arrival time