Program 2 Report

Xiaoyong Liang

XL5432

Pseudocode

1. create an array *graph*, in which each element at index *i* is an array of *Edge* that are connected to device *i*. //used for finding edges connected a node

2. create array of *Node resultGraph* where each *Node* represents a device.

3. **for each** *Node n* in *resultGraph*:

    if *n* is the start node, assign to *n* Tstart

    else set *n* to Tfinal+1

    end if

4. Create a PriorityQueue *queue*, add start node to *queue*.

5. **while** *queue* is not empty:

    current <- first element from *queue*, remove *current* from *queue*.

    if *current* value is greater than *Tfinal*, no path, **end while**.

    if *current* device is final destination, found a path, **end while**.

    add *current* to *Set visited*.

    for each unvisited neighbors *neighbor* of device *current* from *resultGraph*

        e -< edge connecting *current* and *neighbor*

        otherDevice -< device id of *neighbor*

        if (value of *current* is less than or equal to timestamp of *e*

            and value of *neighbor* is greater than timestamp of *e*

and timestamp of *e* is less than or equal to *Tfinal)*:

set value of *neighbor* to timestamp of *e*

add *neighbor* to *queue* based on its value

add key *otherDevice* with value *e* to *trackBack* map.

end if

end for

end while

6. **if** not found any path, print “0”. **end**

7. **else if** found a path:

set *key* to destination device.

while trackBack contain the key *key*

get the edge *e* by *key* from *trackBack,* add *e* to list *trace* at first index

set key to the other device *e* is connected to

end while

8. print list *trace*, containing edges each representing a connection in path.

Time Complexity

The time complexity for this algorithm is O(E + V), ignoring time for queueing and dequeueing operations and reconstructing the trace. Each visited node is added to the visited list so that they are evaluated exactly once. Moreover, we construct a backtrack map to find a node’s parent when tracing back the path after we find a solution; this makes sure that edges are not reevaluated. We can ignore the time complexity for reconstructing our path after we found a path because it uses a map so that the time complexity of acquiring every parent is linear.

Proof of correctness

We denote the value of a node to the timestamp of that device in which it can broadcast the queried message. The nodes inside the priority queue is always reachable from start node during the given timeframe. In each of iteration of while loop, the node with the highest priority is evaluated first and added to the visited set. This ensures that the value of a visited node is never overridden by a smaller value later. If there is a path to end node within the given time, then we would find it. It is because the path must be a sequence with increasing timestamps, and we wouldn’t miss any paths as we evaluate nodes with smaller timestamps first.