CSE344

FINAL

REPORT

Ahmet Ergani

161044011

**Summary**: Project lacks a few things:

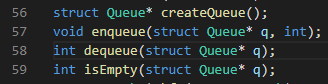
* Server is not a Daemon
* Writers are not prioritized
* There are no timestamps

Rest of the project works well. The reason for this insufficiency is poor time management. I deserve the punishments. Yet still I think my program is still robust

**STRUCTURES**

I implemented a Queue, Graph and a Cache data structure and their element structures for this project.

**Queue**

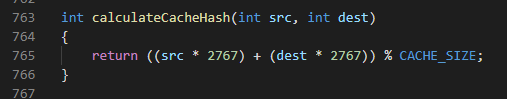
Queue is basic linked list based implementation. There are 4 supported methods and it is used during BFS

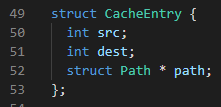
**Graph**

Graph is a traditional implementation that holds a visited array and Adjacency list. Whenever an edge is added, adjList is rearranged accordingly. When the graph is fully loaded we can access the neighbours of a node by simply iteration over it’s linked list. initGraph method initializes this structure by reading it from the file. (Warning I assumed that the Node and Edge counts are provided at the 3. Line during this process).

**Cache**

This structure works as a primitive hash map. It creates an index using a non-complex hash algorithm and saves the calculated paths.

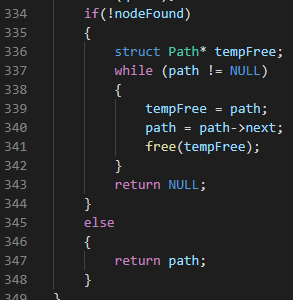


Paths can be overwritten so even if acquired path is not null, a check is necessary. That is why cache entries also have source and destination nodes.

**METHODS**

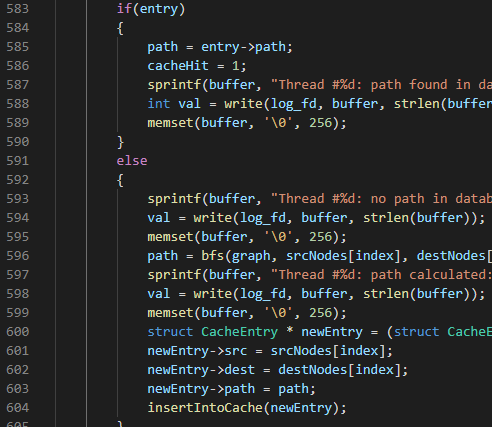
Methods that are not straightforward are explained here. Basic methods can be understood from the comments in the source code.

**bfs()**

This method creates a linked list and traverses the graph according to Breadth First algorithm while adding the visited nodes into this linked list. If it finds the destination node it returns the created linked list. If it cannot dinf the path it frees the linked list and returns NULL.

**assignTask()**

Assigns the client to the first available thread. And awakes it.

**requestHandler()**

This is the worker thread function. At the beginning it tries to lock a mutex that is initially locked. When it is awaken by the assignTask function it first checks whether the server is shutting down or not. If it is not, it acquires the nodes and the socket assigned to global array at it’s index. It first checks if the path is stored in db, if not it calls the bfs function. Afterwards it sends the path to the client node by node and if it calculated this path it stores the path into db

**observer()**

This is the observer thread function. It calculates the system load and calls expandThreads function if necessary.

**APPROACH**

-I created adjacency lists while loading the graph to drastically reduce the time spent by BFS function

-I created a cache using hash algorithm to reduce time complexity to O(1).

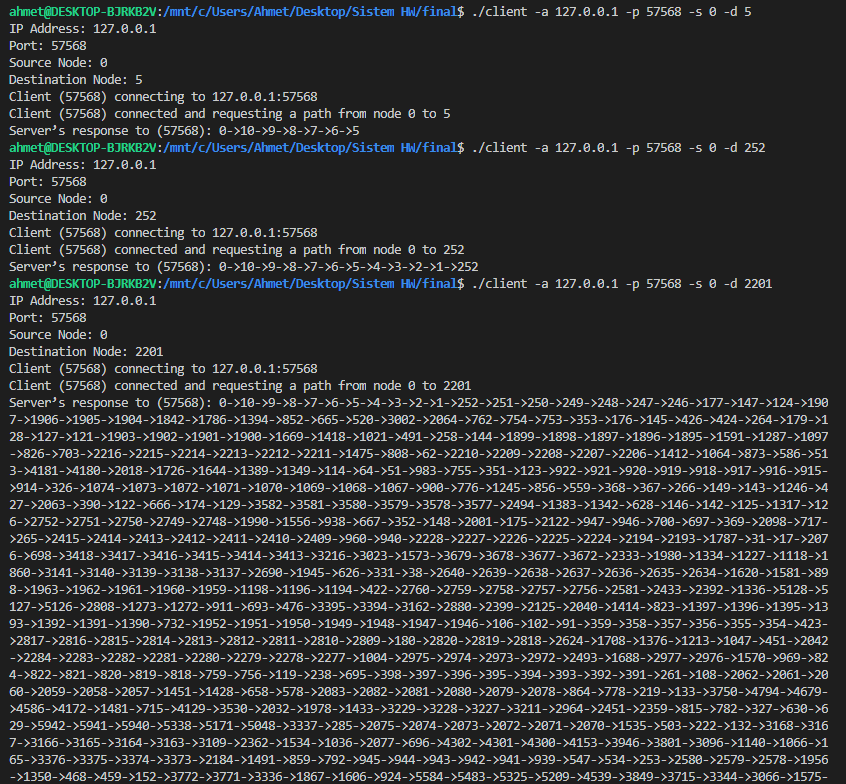
**TESTS**

**Test1 Vanilla Test**

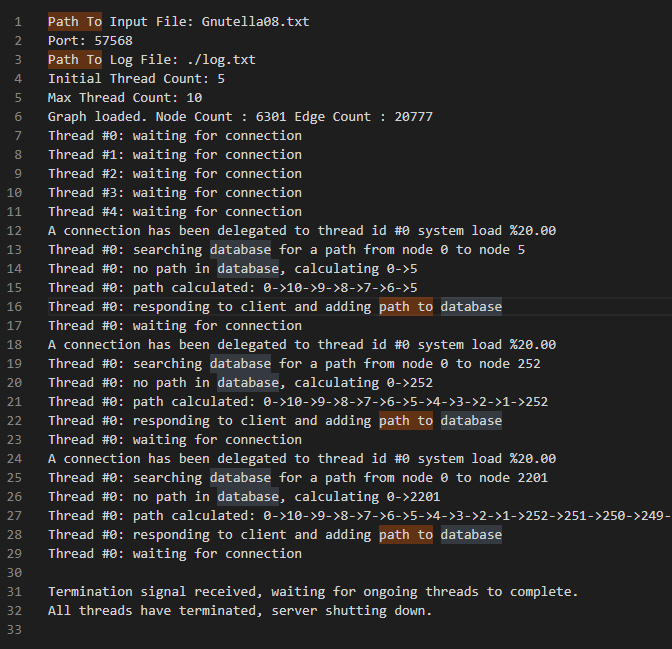
I used the provided graph for successfull testing. I started the server and made 3 queries

* 0 to 5
* 0 to 252
* 0 to 2201

**Client Output:**



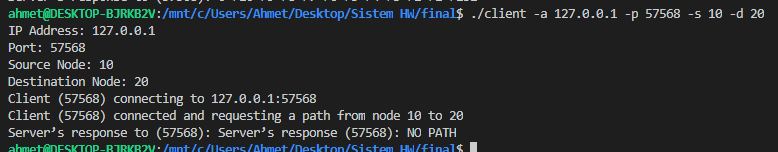
**Server Log File**



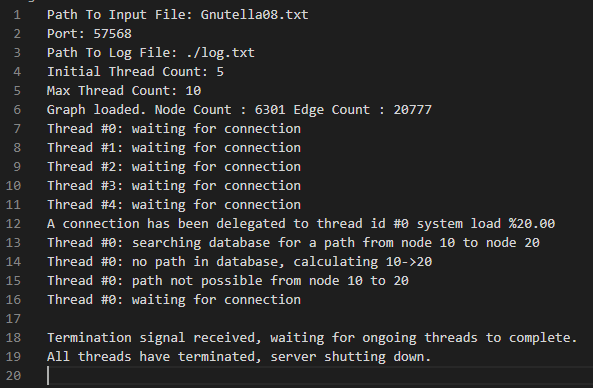
**Test2 No Path Test**

Additionally I tested for a non-existent path which is 10 to 20

**Client Output**

****

**Server Log File**

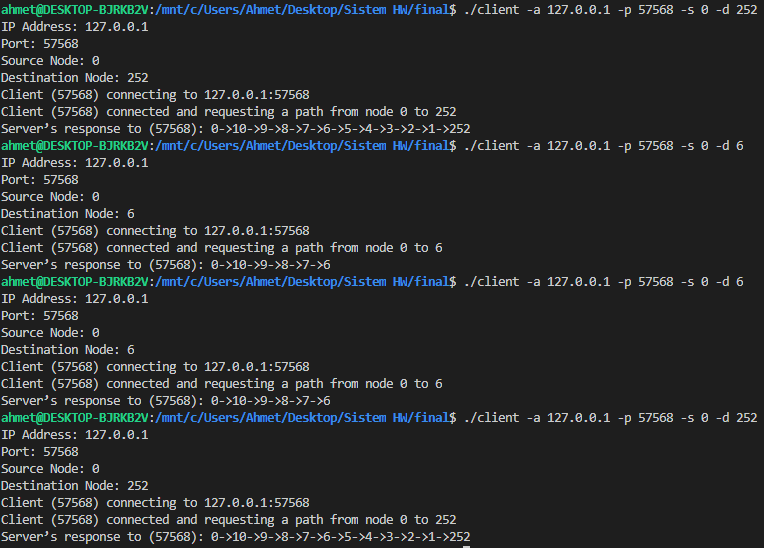
****

**Test3 Cache Test**

I tested 4 queries

* 0 to 252
* 0 to 6
* 0 to 6
* 0 to 252

**Client Output**

****

**Server Log File**

