

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

```
56 struct Queue* createQueue();
57 void enqueue(struct Queue* q, int);
58 int dequeue(struct Queue* q);
59 int isEmpty(struct Queue* q);
```

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.

```
762  
763 int calculateCacheHash(int src, int dest)  
764 {  
765     return ((src * 2767) + (dest * 2767)) % CACHE_SIZE;  
766 }  
767
```

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.

```
49 struct CacheEntry {  
50     int src;  
51     int dest;  
52     struct Path * path;  
53 };  
54
```

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 find the path it frees the linked list and returns NULL.

```
334         if(!nodeFound)
335         {
336             struct Path* tempFree;
337             while (path != NULL)
338             {
339                 tempFree = path;
340                 path = path->next;
341                 free(tempFree);
342             }
343             return NULL;
344         }
345         else
346         {
347             return path;
348         }
349     }
```

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 awoken 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 its index. It

```
583     if(entry)
584     {
585         path = entry->path;
586         cacheHit = 1;
587         sprintf(buffer, "Thread #d: path found in da
588         int val = write(log_fd, buffer, strlen(buffer
589         memset(buffer, '\0', 256);
590     }
591     else
592     {
593         sprintf(buffer, "Thread #d: no path in datab
594         val = write(log_fd, buffer, strlen(buffer));
595         memset(buffer, '\0', 256);
596         path = bfs(graph, srcNodes[index], destNodes[
597         sprintf(buffer, "Thread #d: path calculated:
598         val = write(log_fd, buffer, strlen(buffer));
599         memset(buffer, '\0', 256);
600         struct CacheEntry * newEntry = (struct CacheE
601         newEntry->src = srcNodes[index];
602         newEntry->dest = destNodes[index];
603         newEntry->path = path;
604         insertIntoCache(newEntry);
605     }
```

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 successful testing. I started the server and made 3 queries

- 0 to 5
- 0 to 252
- 0 to 2201

Client Output:

```

ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem Hw/final$ ./client -a 127.0.0.1 -p 57568 -s 0 -d 5
IP Address: 127.0.0.1
Port: 57568
Source Node: 0
Destination Node: 5
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 0 to 5
Server's response to (57568): 0->10->9->8->7->6->5
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem Hw/final$ ./client -a 127.0.0.1 -p 57568 -s 0 -d 252
IP Address: 127.0.0.1
Port: 57568
Source Node: 0
Destination Node: 252
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 0 to 252
Server's response to (57568): 0->10->9->8->7->6->5->4->3->2->1->252
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem Hw/final$ ./client -a 127.0.0.1 -p 57568 -s 0 -d 2201
IP Address: 127.0.0.1
Port: 57568
Source Node: 0
Destination Node: 2201
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 0 to 2201
Server's response to (57568): 0->10->9->8->7->6->5->4->3->2->1->252->251->250->249->248->247->246->177->147->124->190
7->1906->1905->1904->1842->1786->1394->852->665->520->3002->2064->762->754->753->353->176->145->426->424->264->179->1
28->127->121->1903->1902->1901->1900->1669->1418->1021->491->258->144->1899->1898->1897->1896->1895->1591->1287->1097
->826->703->2216->2214->2213->2212->2211->1475->808->62->2210->2209->2208->2207->2206->1412->1064->873->586->51
3->4181->4180->2018->1726->1644->1389->1349->114->64->51->983->755->351->123->922->921->920->919->918->917->916->915-
>914->326->1074->1073->1072->1071->1070->1069->1068->1067->900->776->1245->856->559->368->367->266->149->143->1246->4
27->2063->390->122->666->174->129->3582->3581->3580->3579->3578->3577->2494->1383->1342->628->146->142->125->1317->12
6->2752->2751->2750->2749->2748->1990->1556->938->667->352->148->2001->175->2122->947->946->700->697->369->2098->717
->265->2415->2414->2413->2412->2411->2410->2409->960->940->2228->2227->2226->2225->2224->2194->2193->1787->31->17->207
6->698->3418->3417->3416->3415->3414->3413->3216->3023->1573->3679->3678->3677->3672->2333->1980->1334->1227->1118->1
860->3141->3140->3139->3138->3137->2690->1945->626->331->38->2640->2639->2638->2637->2636->2635->2634->1620->1581->89
8->1963->1962->1961->1960->1959->1198->1196->1194->422->2760->2759->2758->2757->2756->2581->2433->2392->1336->5128->5
12->5126->2808->1273->1272->911->693->476->3395->3394->3162->2880->2399->2125->2040->1414->823->1397->1396->1395->13
93->1392->1391->1390->732->1952->1951->1950->1949->1948->1947->1946->106->102->91->359->358->357->356->355->354->423-
>2817->2816->2815->2814->2813->2812->2811->2810->2809->180->2820->2819->2818->2624->1708->1376->1213->1047->451->2042
->2284->2283->2282->2281->2280->2279->2278->2277->1004->2975->2974->2973->2972->2493->1688->2977->2976->1570->969->82
4->822->821->820->819->818->759->756->119->238->695->398->397->396->395->394->393->392->391->261->108->2062->2061->20
60->2059->2058->2057->1451->1428->658->578->2083->2082->2081->2080->2079->2078->864->778->219->133->3750->4794->4679-
>4586->4172->1481->715->4129->3530->2032->1978->1433->3229->3228->3227->3211->2964->2451->2359->815->782->327->630->6
29->5942->5941->5940->5338->5171->5048->3337->285->2075->2074->2073->2072->2071->2070->1535->503->222->132->3168->316
7->3166->3165->3164->3163->3109->2362->1534->1036->2077->696->4302->4301->4300->4153->3946->3801->3096->1140->1066->1
65->3376->3375->3374->3373->2184->1491->859->792->945->944->943->942->941->939->547->534->253->2580->2579->2578->1956-
>1350->468->459->152->3772->3771->3336->1867->1606->924->5584->5483->5325->5209->4539->3849->3715->3344->3066->1575

```

Server Log File

```
1 Path To Input File: Gnutella08.txt
2 Port: 57568
3 Path To Log File: ./log.txt
4 Initial Thread Count: 5
5 Max Thread Count: 10
6 Graph loaded. Node Count : 6301 Edge Count : 20777
7 Thread #0: waiting for connection
8 Thread #1: waiting for connection
9 Thread #2: waiting for connection
10 Thread #3: waiting for connection
11 Thread #4: waiting for connection
12 A connection has been delegated to thread id #0 system load %20.00
13 Thread #0: searching database for a path from node 0 to node 5
14 Thread #0: no path in database, calculating 0->5
15 Thread #0: path calculated: 0->10->9->8->7->6->5
16 Thread #0: responding to client and adding path to database
17 Thread #0: waiting for connection
18 A connection has been delegated to thread id #0 system load %20.00
19 Thread #0: searching database for a path from node 0 to node 252
20 Thread #0: no path in database, calculating 0->252
21 Thread #0: path calculated: 0->10->9->8->7->6->5->4->3->2->1->252
22 Thread #0: responding to client and adding path to database
23 Thread #0: waiting for connection
24 A connection has been delegated to thread id #0 system load %20.00
25 Thread #0: searching database for a path from node 0 to node 2201
26 Thread #0: no path in database, calculating 0->2201
27 Thread #0: path calculated: 0->10->9->8->7->6->5->4->3->2->1->252->251->250->249-
28 Thread #0: responding to client and adding path to database
29 Thread #0: waiting for connection
30
31 Termination signal received, waiting for ongoing threads to complete.
32 All threads have terminated, server shutting down.
33
```


Test2 No Path Test

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

Client Output

```
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem HW/final$ ./client -a 127.0.0.1 -p 57568 -s 10 -d 20
IP Address: 127.0.0.1
Port: 57568
Source Node: 10
Destination Node: 20
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 10 to 20
Server's response to (57568): Server's response (57568): NO PATH
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem HW/final$
```

Server Log File

```
1 Path To Input File: Gnutella08.txt
2 Port: 57568
3 Path To Log File: ./log.txt
4 Initial Thread Count: 5
5 Max Thread Count: 10
6 Graph loaded. Node Count : 6301 Edge Count : 20777
7 Thread #0: waiting for connection
8 Thread #1: waiting for connection
9 Thread #2: waiting for connection
10 Thread #3: waiting for connection
11 Thread #4: waiting for connection
12 A connection has been delegated to thread id #0 system load %20.00
13 Thread #0: searching database for a path from node 10 to node 20
14 Thread #0: no path in database, calculating 10->20
15 Thread #0: path not possible from node 10 to 20
16 Thread #0: waiting for connection
17
18 Termination signal received, waiting for ongoing threads to complete.
19 All threads have terminated, server shutting down.
20 |
```

Test3 Cache Test

I tested 4 queries

- 0 to 252
- 0 to 6
- 0 to 6
- 0 to 252

Client Output

```
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem Hw/final$ ./client -a 127.0.0.1 -p 57568 -s 0 -d 252
IP Address: 127.0.0.1
Port: 57568
Source Node: 0
Destination Node: 252
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 0 to 252
Server's response to (57568): 0->10->9->8->7->6->5->4->3->2->1->252
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem Hw/final$ ./client -a 127.0.0.1 -p 57568 -s 0 -d 6
IP Address: 127.0.0.1
Port: 57568
Source Node: 0
Destination Node: 6
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 0 to 6
Server's response to (57568): 0->10->9->8->7->6
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem Hw/final$ ./client -a 127.0.0.1 -p 57568 -s 0 -d 6
IP Address: 127.0.0.1
Port: 57568
Source Node: 0
Destination Node: 6
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 0 to 6
Server's response to (57568): 0->10->9->8->7->6
ahmet@DESKTOP-BJRK82V:/mnt/c/Users/Ahmet/Desktop/Sistem Hw/final$ ./client -a 127.0.0.1 -p 57568 -s 0 -d 252
IP Address: 127.0.0.1
Port: 57568
Source Node: 0
Destination Node: 252
Client (57568) connecting to 127.0.0.1:57568
Client (57568) connected and requesting a path from node 0 to 252
Server's response to (57568): 0->10->9->8->7->6->5->4->3->2->1->252
```

Server Log File

```
1 Path To Input File: Gnutella08.txt
2 Port: 57568
3 Path To Log File: ./log.txt
4 Initial Thread Count: 5
5 Max Thread Count: 10
6 Graph loaded. Node Count : 6301 Edge Count : 20777
7 Thread #0: waiting for connection
8 Thread #2: waiting for connection
9 Thread #1: waiting for connection
10 Thread #3: waiting for connection
11 Thread #4: waiting for connection
12 A connection has been delegated to thread id #0 system load %20.00
13 Thread #0: searching database for a path from node 0 to node 252
14 Thread #0: no path in database, calculating 0->252
15 Thread #0: path calculated: 0->10->9->8->7->6->5->4->3->2->1->252
16 Thread #0: responding to client and adding path to database
17 Thread #0: waiting for connection
18 A connection has been delegated to thread id #0 system load %20.00
19 Thread #0: searching database for a path from node 0 to node 6
20 Thread #0: no path in database, calculating 0->6
21 Thread #0: path calculated: 0->10->9->8->7->6
22 Thread #0: responding to client and adding path to database
23 Thread #0: waiting for connection
24 A connection has been delegated to thread id #0 system load %20.00
25 Thread #0: searching database for a path from node 0 to node 6
26 Thread #0: path found in database: 0->10->9->8->7->6
27 Thread #0: responding to client
28 Thread #0: waiting for connection
29 A connection has been delegated to thread id #0 system load %20.00
30 Thread #0: searching database for a path from node 0 to node 252
31 Thread #0: path found in database: 0->10->9->8->7->6->5->4->3->2->1->252
32 Thread #0: responding to client
33 Thread #0: waiting for connection
34
35 Termination signal received, waiting for ongoing threads to complete.
36 All threads have terminated, server shutting down.
37
```