STEVENS INSTITUTE OF TECHNOLOGY

Computer Science (Master's)

CS 590 A - ALGORITHMS
ASSIGNMENT 5

Submission to:
Professor In Suk Jang

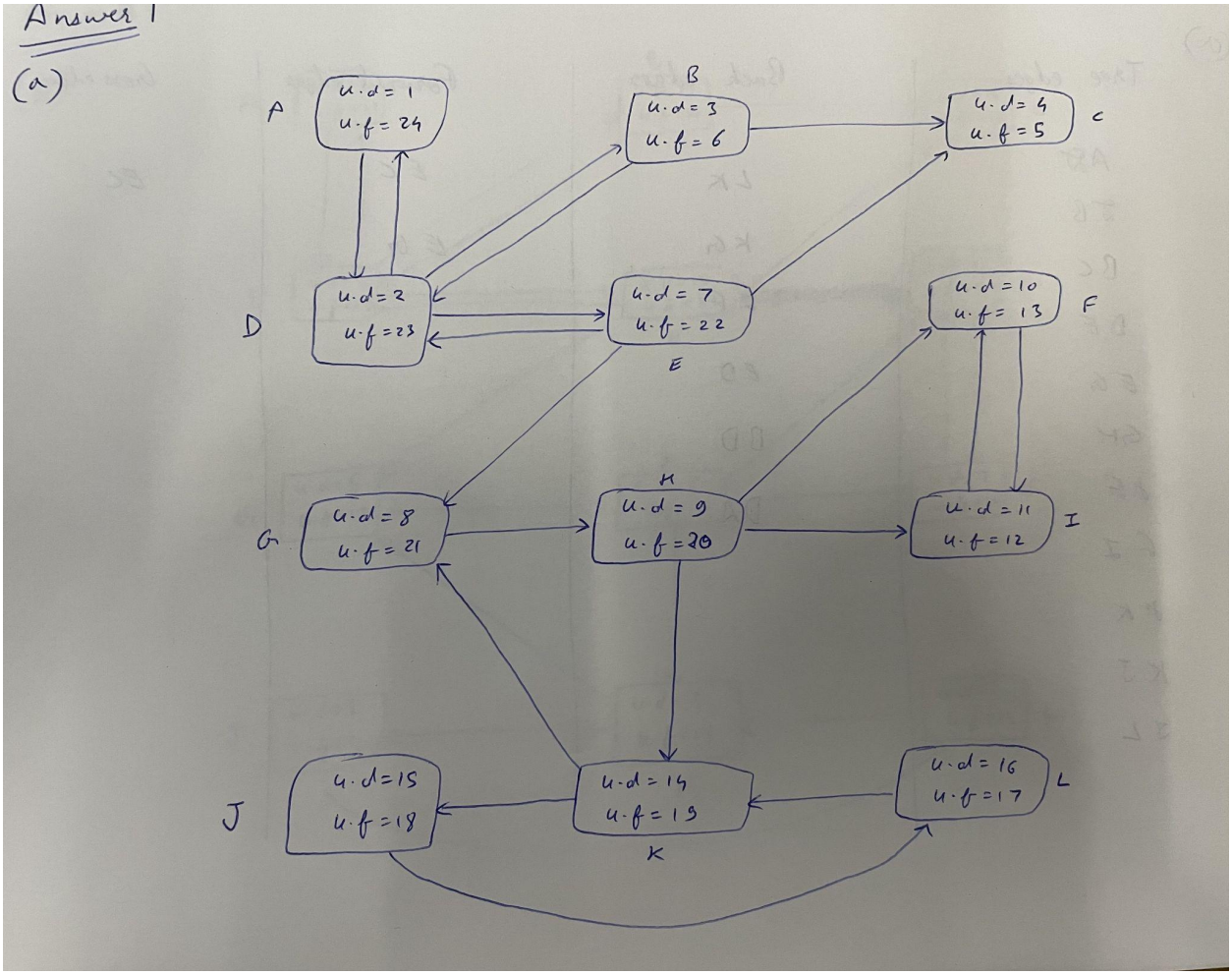
Submission By:
Chetan Goyal
CWID: 20005334

Date: December 11, 2021

Answer 1

Part (a)

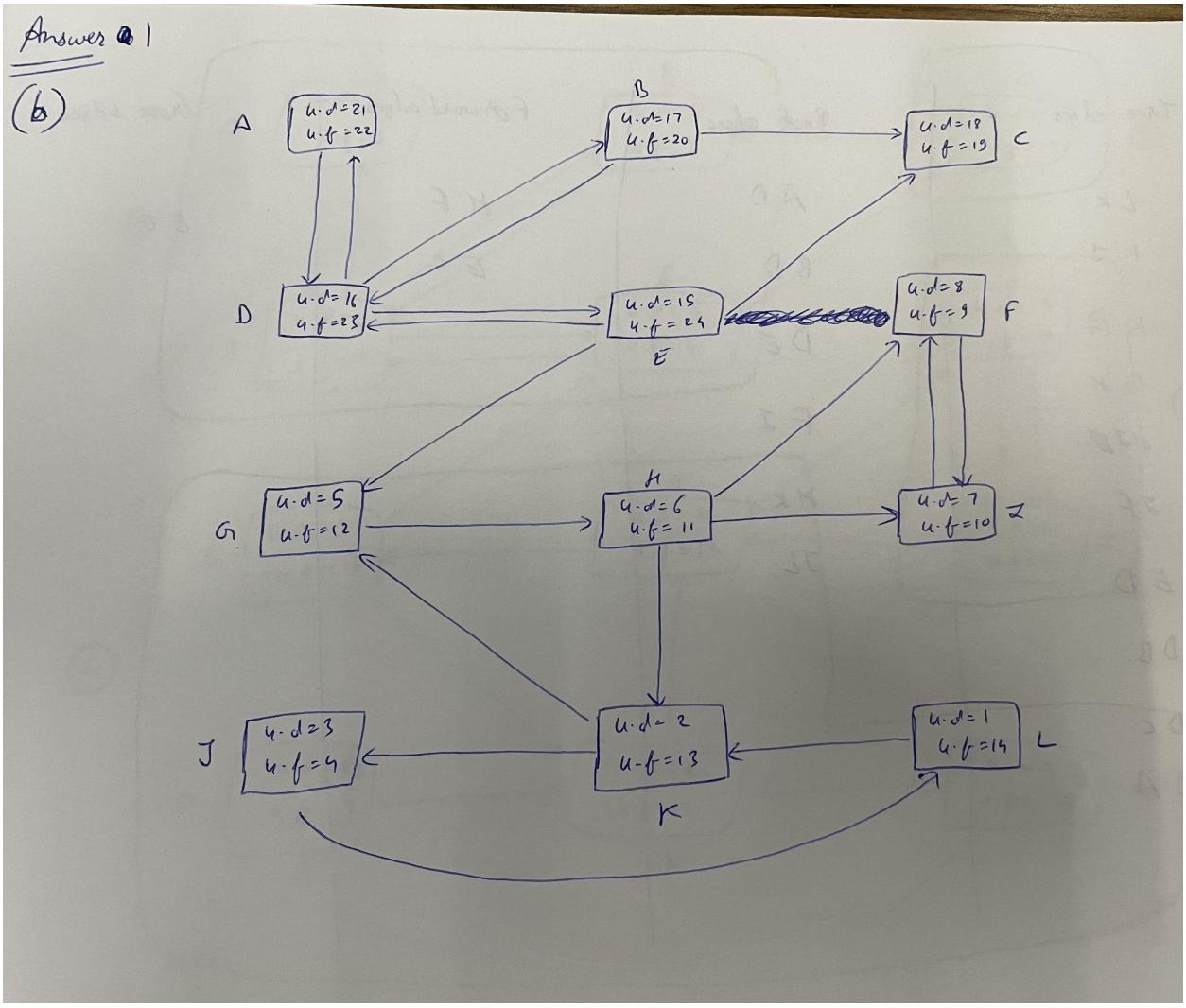
In the attached image a DFS has been performed in increasing order from A to L and their discovery and finish times are given within the image.



Tree Edges - AD, DB, BC, DE, EG, GH, HF, FI, HK, KJ, JL
Back Edges - LK KG HF IF ED BD DA
Forward Edges - EC HI
Cross Edges - EG

Part (b)

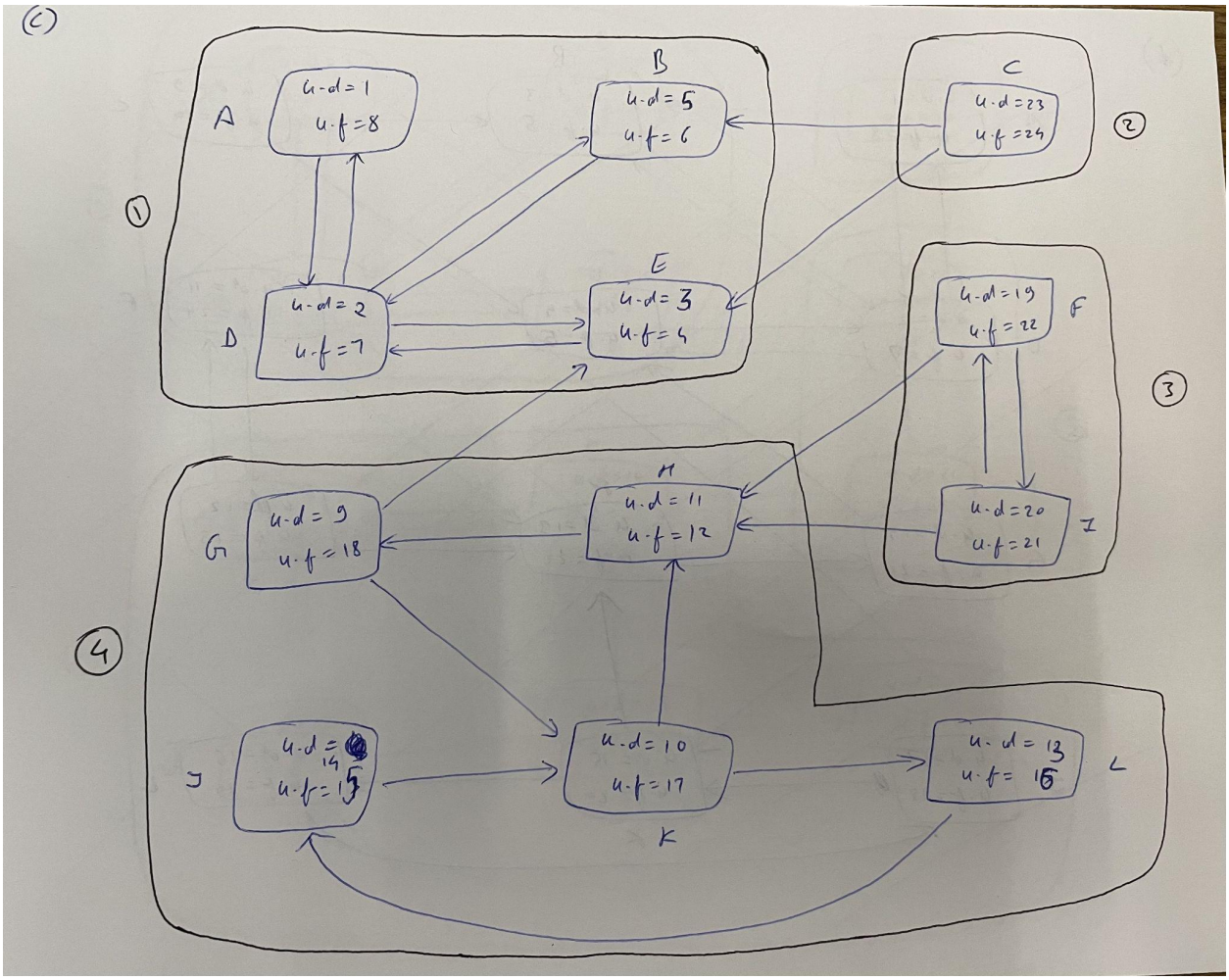
In the attached image a DFS has been performed in decreasing order from L to A and their discovery and finish times are given within the image.



Tree edges - LK, KJ, KG, GH, HI, IF, ED, DB, DC, DA
Back edges - AD, BD, DE, FI, HK, JL
Forward edges - HF, EC
Cross edges - EG

Part (c)

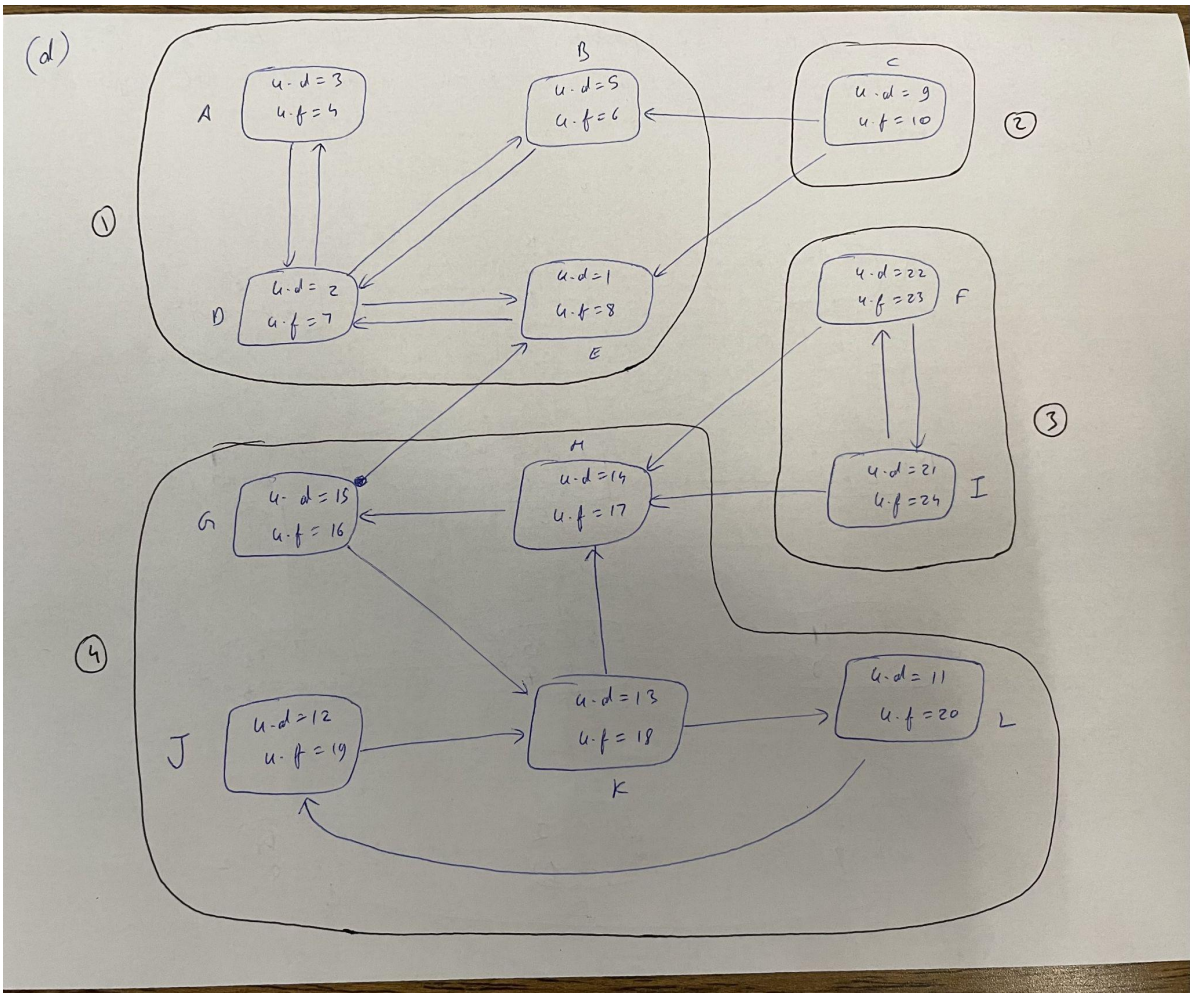
For the SCC problem, we have first taken a transpose of our given graph, as can be seen by the edges of the graph. Then we have done a DFS again, as per the requirement of the SCC algorithms, i.e, we performed a DFS in the decreasing order of finish time (u.f), as obtained for this in part (a). The strongly connected components have been highlighted in a black box and with their number.
The discovery and finish times are given within the vertices of the graph, in the attached image.



SCC 1- ABDE
SCC 2- C
SCC 3- FI
SCC 4 - GHKLJ

Part (d)

For the SCC problem, we have first taken a transpose of our given graph, as can be seen by the edges of the graph. Then we have done a DFS again, as per the requirement of the SCC algorithms, i.e, we performed a DFS in the decreasing order of finish time (u.f), as obtained for this in part (b). The strongly connected components have been highlighted in a black box and with their number.
The discovery and finish times are given within the vertices of the graph, in the attached image.



SCC 1- ABDE
SCC 2- C
SCC 3- FI
SCC 4 - GHKLJ

Answer 2

Analysis

In the given codes, we have written in C++ to perform a BFS and DFS traversal of the graph.

Breadth First Search (BFS)

A standard BFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The algorithm works as follows:

1. Start by putting any one of the graph's vertices at the back of a queue
2. Take the front item of the queue and add it to the visited list
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty

The graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the BFS algorithm on every node.

Time complexity of the BFS algorithm is represented in the form of O(V+E), where V is the number of nodes and E is the number of edges.
Space complexity of the BFS algorithm is O(E).

Depth First Search (DFS)

A standard DFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

The purpose of the algorithms is to mark each vertex as visited while avoiding cycles.

The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack
2. Take the top item of the stack and add it to the visited list
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack
4. Keep repeating steps 2 and 3 until the stack is empty

Time complexity of the DFS algorithm is represented in the form of O(V+E), where V is the number of nodes and E is the number of edges.
Space complexity of the DFS algorithm is O(E).