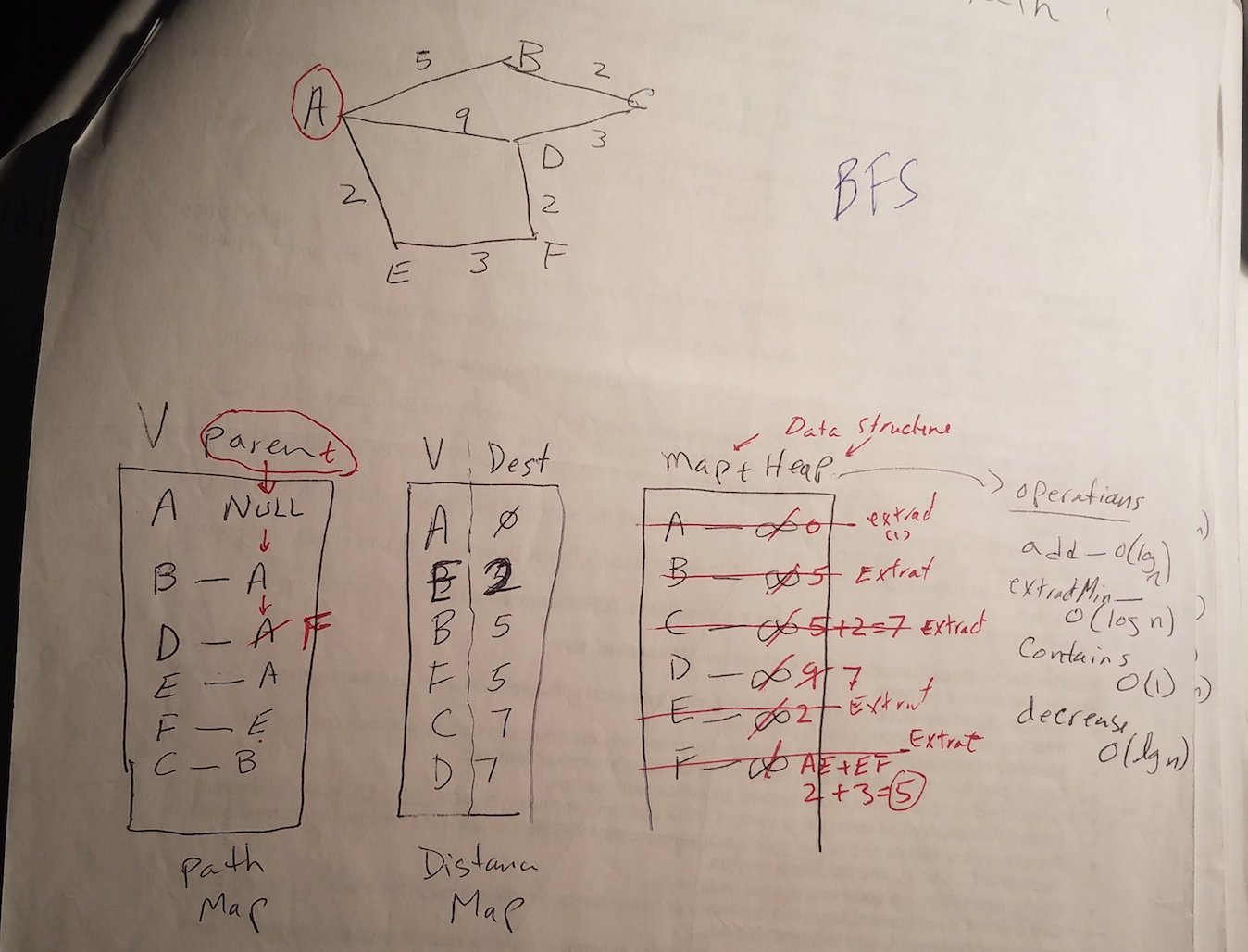
# Data Structures and Algorithms

# INFO 6205

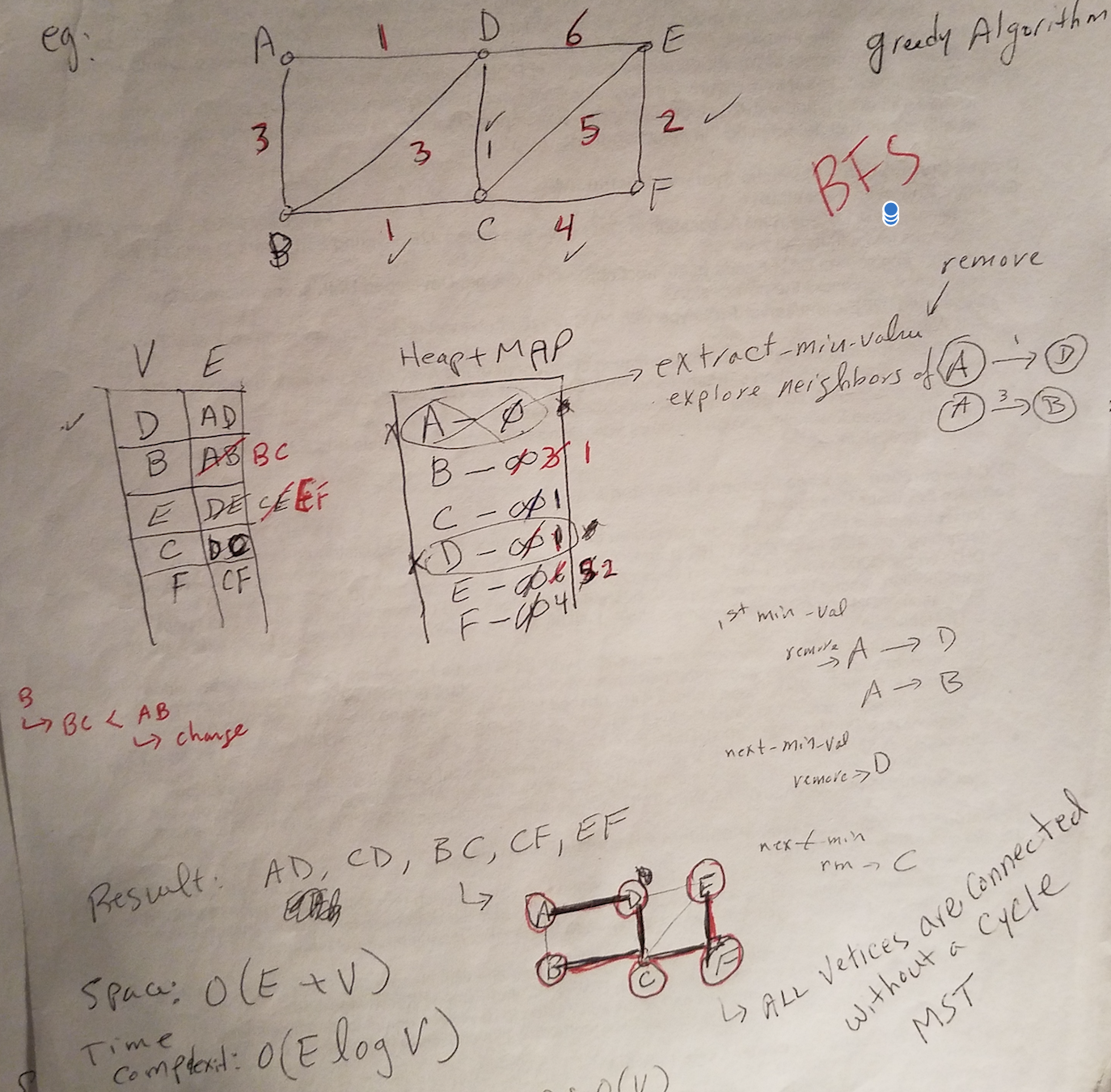
# Homework 9

# Due: November 21, 2020

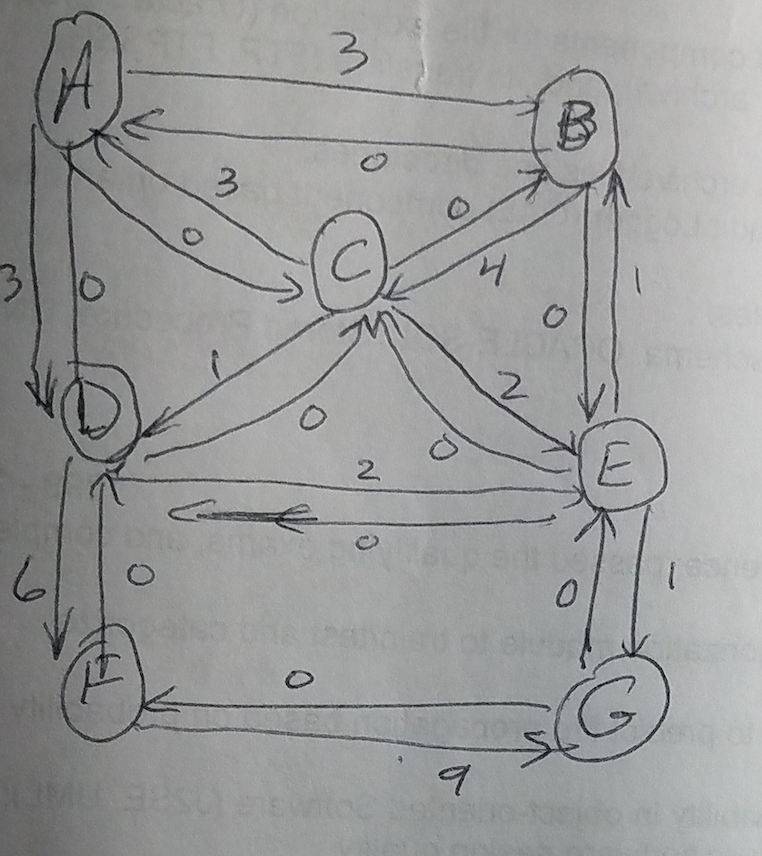
Put all your java, compiled class files and documentation files into a zip file named Homework9.zip and submit it via the dropbox on canvas before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

1. The following graph finds shortest-path from source vertex A to any destination vertex on the graph. Explain the data structures and show step-by-step algorithm to final solution presented in data structures. Next show how to use Path Map data structure to go from A to any destination.

2. The following graph finds Minimum Spanning Tree using Prim’s algorithm. Explain the data structures showing step-by-step algorithm to find the final solution presented in data structures. Next, show how to use edge-table data structure to show the solution for minimum spanning tree.



3. Consider the following Ford-Fulkerson Network Flow graph problem, show step-by-step algorithm including data structures and augmented-paths to find Maximum Flow from source A to destination G



4. Consider Ford-Fulkerson Algorithm:

<https://www.tutorialspoint.com/Ford-Fulkerson-Algorithm>

Explain step-by-step of how the code algorithm works. Is there a recursive nature to the the algorithm? What is its termination point?

5. Identify each of these algorithms, Explain in details

A)

1. Make a queue (Q) with all the vertices of G (V);

2. For each member of Q set the priority to INFINITY;

3. Only for the starting vertex (s) set the priority to 0;

4. The parent of (s) should be NULL;

5. While Q isn’t empty

6. Get the minimum from Q – let’s say (u); (priority queue);

7. For each adjacent vertex to (v) to (u)

8. If (v) is in Q and weight of (u, v) < priority of (v) then

9. The parent of (v) is set to be (u)

10. The priority of (v) is the weight of (u, v)

B)

1. T is defined to be the empty set;

2. For each vertex v of G, make the empty set out of v;

3. Sort the edges of G in ascending (non-decreasing) order;

4. For each edge (u, v) from the sorted list of step 3.

If u and v belong to different sets

Add (u,v) to T;

Get together u and v in one single set;

5. Return T

C)

1. Sort G into L;

2. Set the distance to the source to 0;

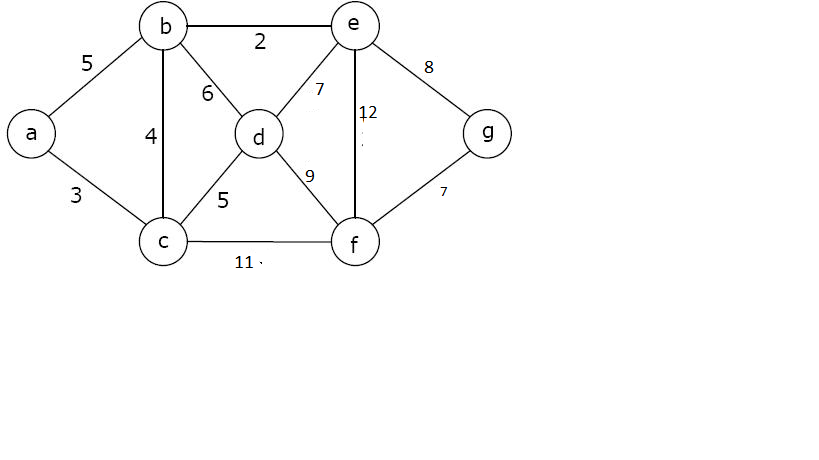
3. Set the distances to all other vertices to infinity;

4. For each vertex u in L

5. - Walk through all neighbors v of u;

6. - If dist(v) > dist(u) + w(u, v)

7. - Set dist(v) <- dist(u) + w(u, v);

6. Consider the following Graph, ` 

a) Dijkstra Shortest-Path algorithm step-by step, show data structures

b) Prim’s Algorithm step-by-step, show data structures

c) Write Java code, compile and test for (a) and (b) using Algorithms

d) Compare Space and Time complexity between the two algorithms