1. (4 points) Consider the following function from FlowEdge.java. This function is used to allocate flow to an edge in a residual graph. Fill in the blanks with the appropriate type of edge.

```java
public void addResidualFlowTo(int vertex, double delta) {

    // allocate flow to a _BACKWARDS___________ edge
    if (vertex == v) flow -= delta;

    // allocate flow to a _FORWARD___________ edge
    else if (vertex == w) flow += delta;

    else throw new IllegalArgumentException("Illegal endpoint");

    if (Double.isNaN(delta))
        throw new IllegalArgumentException("Change in flow = NaN");
    if (!(flow >= 0.0))
        throw new IllegalArgumentException("Flow is negative");
    if (!(flow <= capacity))
        throw new IllegalArgumentException("Flow exceeds capacity");

}
```

2. (2 points) What is the runtime of Eager Prim's?

$\Theta(elgv)$

3. (2 points) What is the runtime of Dijkstra's algorithm?

$\Theta(elgv)$
4. (7 points) Give the pseudocode for Kruskal’s algorithm.

Add all edges in the graph to a PQ. Remove min edges from the PQ and add them to the MST so long as they do not create a cycle in the MST.
5. (10 points) Use Dijkstra’s algorithm to determine the shortest path from vertex A to vertex G in the following graph. In addition to stating the shortest path from A to G, state the order in which you visit the vertices of the graph in finding this shortest path.

Shortest path: A D F G of weight 20
Visited order: A B C D E F G