Homework 3

# Problem

## a)

We can model Beth as a graph.

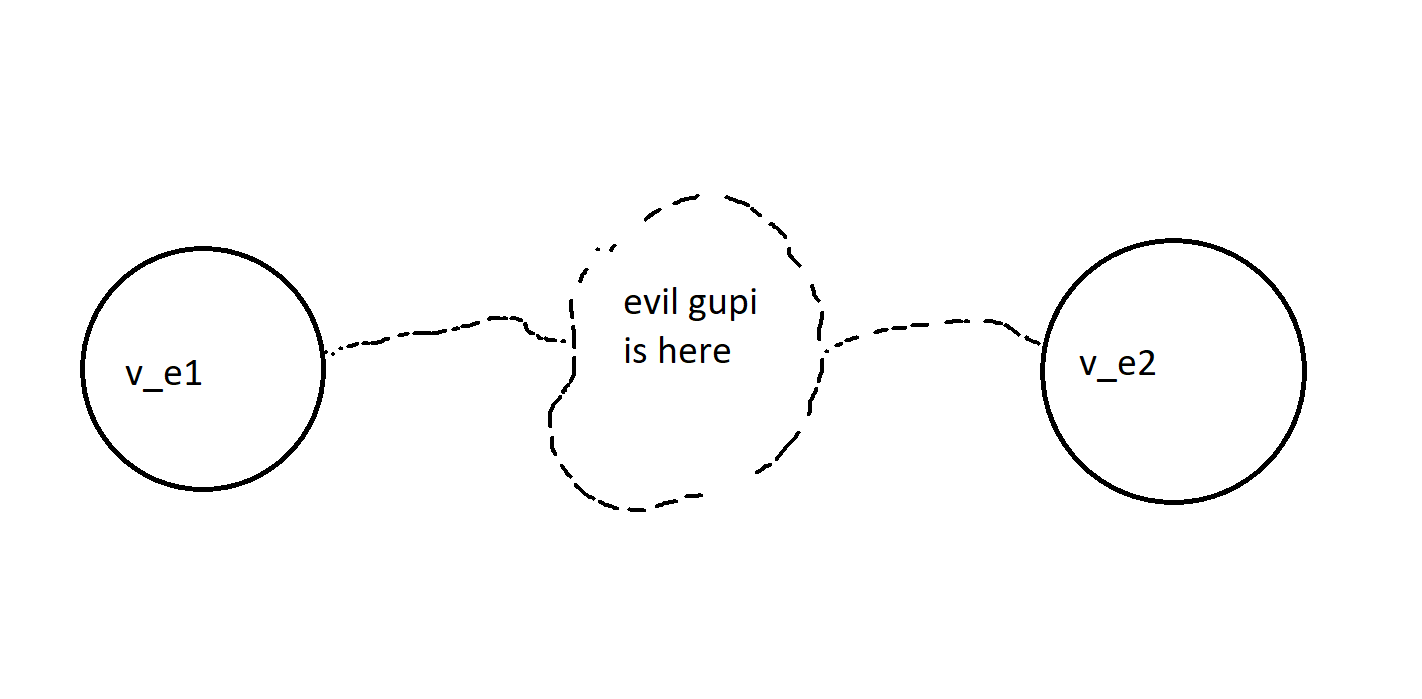
Each intersection is a vertex, and the roads connecting the intersections are edges. The two entrances can be considered as two intersections (i.e. can be modeled as two vertices).

Since we do not want to be kidnapped by the evil gupi, we in turn cannot include the vertex (or intersection) where the gupi is sitting at in our graph. We also need to remove the edges touching this vertex.

The problem now becomes a graph connectivity problem. We want to know if there exists a path connecting the two entrances nodes and .

With the setup above, we can provide a counterexample to show that it is not always possible to go through the city. Imagine the situation where there’s only one intersection connecting the two entrances, and the evil gupi is located at that single intersection.

In the figure below, the dotted node is where the evil gupi is located at, so it is not part of the graph. It is easy to see that there is no path from to .



# b)

Not always possible.

# c)

We can use BFS to check if there’s a path from to (remember to remove the node where the evil gupi is and all edges touching that node from the graph).

We set the initial node to be . Keep a set to make sure we don’t revisit each node. Also, as we generate each node we keep track of what the parent of each node is in the BFS search.

If at any point we generate during the BFS we can declare that we can safely pass through Beth. To obtain the path, we can use the parent info at to trace back to iteratively to output the path.

However, if the BFS terminates before generating , we have to declare that there’s no way to pass through Beth. To return to , simply use the parent node information to go back.