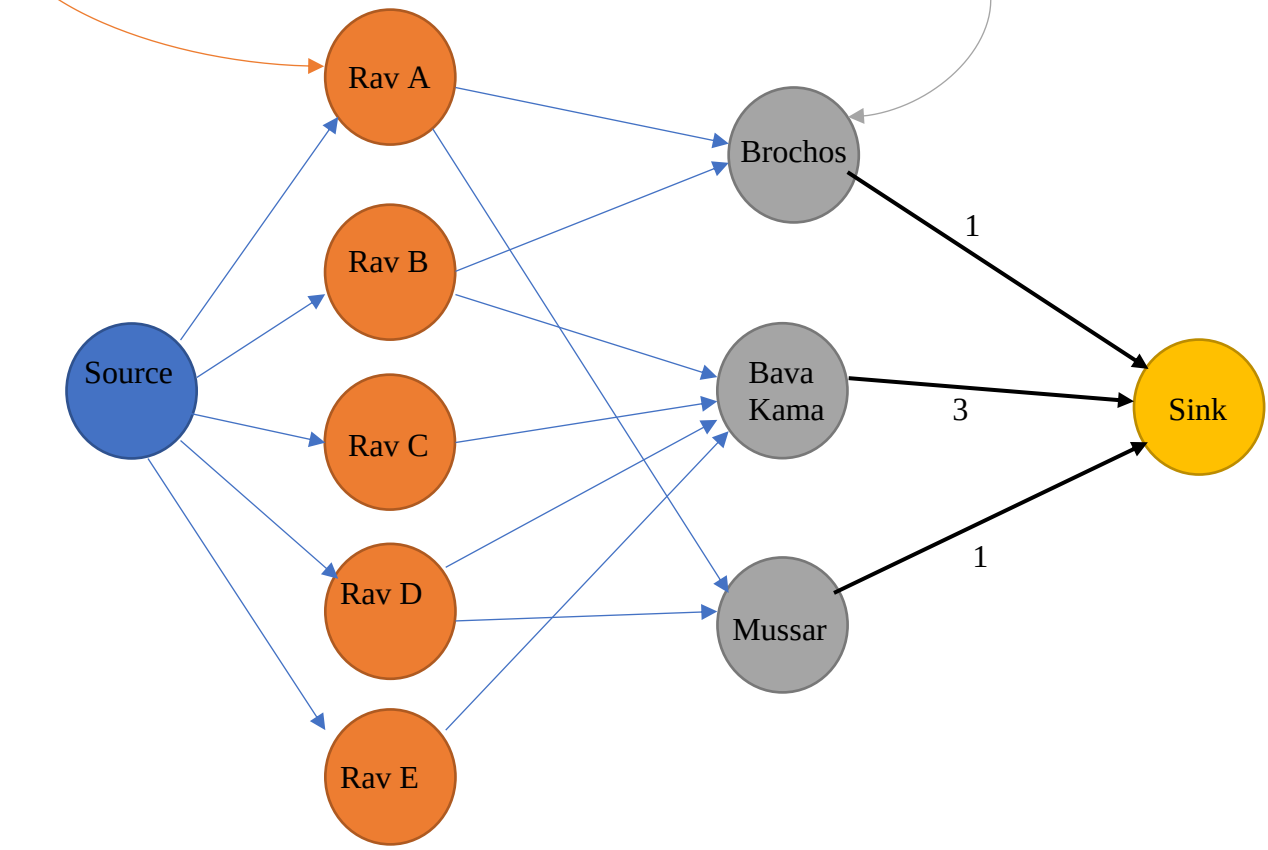
Adam Frenkel

Model: Create a graph with source and sick vertices. Then, create vertices for each Rebbe. Next, connect the source vertex to each of the Rebbes with edges with capacities of one. Then, add vertices for each of the help topics that received a request. Following this, make an edge, with a capacity of one, from every Rebbe to the help topics that they are able to help with. Finally, add an edge from each help topic to the sink vertex. The capacity of each of those edges is the number of requests for that particular topic.

Justification: Based on the proof of correctness for bipartite matching problem on the slides, you can prove that this is correct. This problem can be viewed as a simple one to one matching. Whenever there is a help topic with more than one request (x requests), instead of viewing it as one topic with a flow of x to the source, you can view it as x separate topics each with a flow of one to the source. In the example below, that would mean that there would be three Bava Kamas, each with an arrow from the rebbes that can help with that topic going in (with a flow of 1) and each with a flow of one to the source. This is now the exact same problem as bipartite matching, which has already been proven to be correct.

Diagram: Next page.

```
public void sample(){
    HelpFromRabbeim hfr = new HelpFromRabbeim();
    List<HelpFromRabbeimI.Rebbe> rebbes = new ArrayList<>();
    List<HelpFromRabbeimI.HelpTopics> hts1 = new ArrayList<>();
    hts1.add(HelpFromRabbeimI.HelpTopics.BROCHOS);
    hts1.add(HelpFromRabbeimI.HelpTopics.MUSSAR);
    HelpFromRabbeimI.Rebbe A = new HelpFromRabbeimI.Rebbe(1,hts1);
    rebbes.add(A);
    Map<HelpFromRabbeimI.HelpTopics,Integer> htMap = new HashMap<>();
    htMap.put(HelpFromRabbeimI.HelpTopics.BROCHOS,1);
    ...
    System.out.println(hfr.scheduleIt(rebbes,htMap));
}
```



KEY:

All Blue arrows all have a capacity of 1, every time.
The **Black** arrows capacity is the number of requests for help for that topic.
The Orange and Gray arrows are just there for mapping the code to the graph; they aren't part of the graph.
It should be noted that the sample code is only a snippet of the entire code, as indicted by the ellipsis.

Solution:
A – Brochos
B – Bava Kama
C – Bava Kama
D – Mussar
E – Bava Kama