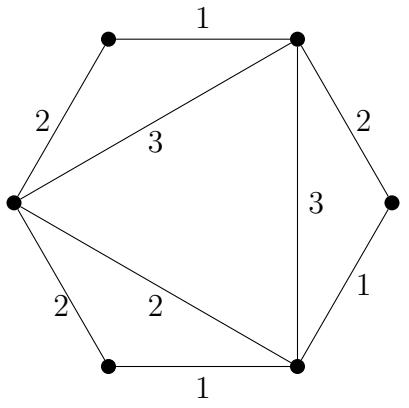
MATH-566 HW 9

Due **Nov 4** before class. Just bring it before the class and it will be collected there.

1: (*Gomory-Hu Tree*)
Construct Gomory-Hu Tree for the following graph. Numbers on edges correspond to capacities.



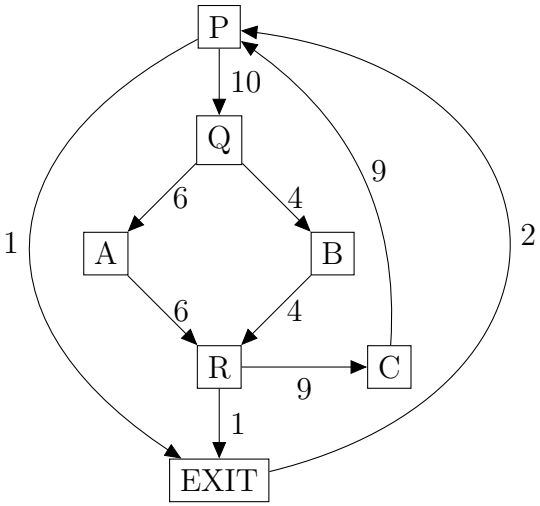
2: (*Profiling computer programs*)
Recall that code in a compiler can be represented as *basic blocks* connected by directed edges. This graph is called *control-flow graph*, it is a rooted graph (it has a special entry vertex called *root*), has a special vertex called $EXIT$. In addition, it has a special edge $EXIT \to root$.

Every block of code has at most two leaving edges and may have several in edges. In order to optimize the code, the compiler needs to know how frequently is the program traversing each edge (and how often each basic block is executed). Measuring transition on every edge is very costly. But the same measurement can be done if we take a spanning tree (in the underlying undirected graph) and count only transitions on edges that are not part of the spanning tree. Show that it is possible to reconstruct how many times each bloc/edge were used from the knowledge of non-tree edges.

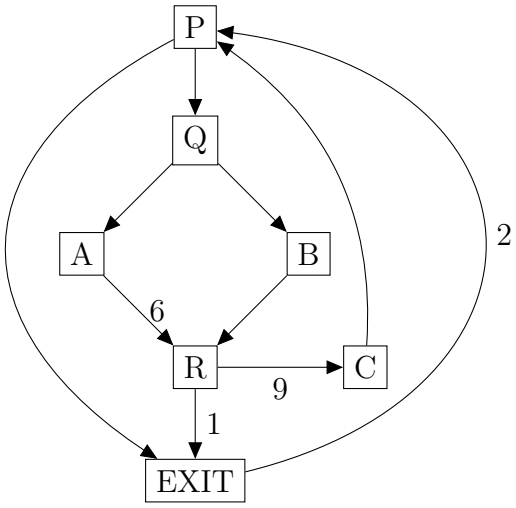
Example: Consider the following program:

```
while P do
    if Q then
        A
    else
        B
    if R then break
    C
endwhile
```

The corresponding control-flow graph is depicted below. Suppose we run the program twice. Numbers on edges correspond to how many times each edge was traversed. Notice that the diagram satisfies Kirchoff's laws and it is actually a circulation.



Now assume that we measure the number of transitions only on some edges, where the skipped edges form a tree. Notice that it is possible to reconstruct the values on edges that were not measured.



Your task is to show that this method works in general. (If you don't like the compiler motivation, here is an alternative definition - suppose there is a network with a circulation on it. Suppose that value of the circulation is not know on edges that form a tree. How to compute the values of the unknown edges?)

More details about the algorithm can be found at `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.5085&rep=rep1&type=pdf`