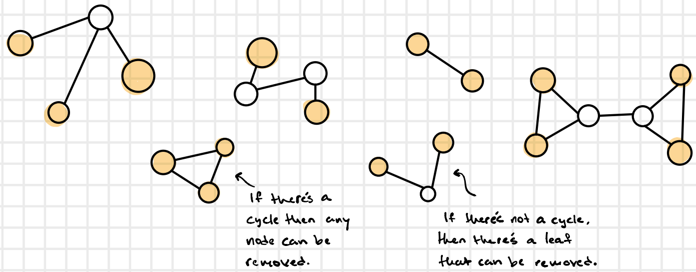
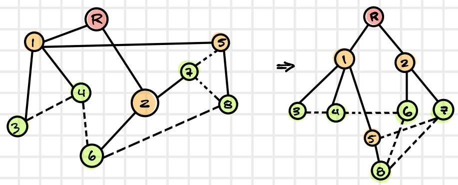
2. Recognize that any vertex with degree of one can safely be removed without disconnecting the graph. Recognize also that BFS and DFS can be used on arbitrary graphs to create BFS trees and DFS trees, respectively, with leaves that, by definition have degree of one. BFS and DFS can also, conveniently, run in $O(m+n)$. Thus, we modify BFS to store the leaves found by BFS as valid vertices to remove without disconnecting the graph. We then return the last found leaf. Note that the BFS algorithm is a modified version from Algorithm Design by Kleinbergs and Tardos.

```
modifiedBFS(s):
  Set Discovered[s] = true and Discovered[v] = false for all other v
  Initialize L[0] to consist of the single element s
  Set the layer counter i = 0
  Initialize list A to store the leaves of the BFS tree
  While L[i] is not empty
    Initialize an empty list L[i+1]
    For each node u ∈ L[i]
      Initialize boolean b to true to flag if u is a leaf
      Consider each edge (u, v) incident to u
      If Discovered[v] = false then
        Set Discovered[v] = true
        Add v to the list L[i+1]
        Set b to false
      Endif
      If b = true then
        Add u to A
      Endif
    Endfor
    Increment the layer counter i by 1
  Endwhile
  Return the last element in A
```





Claim: The above modifiedBFS terminates in $O(m+n)$.

Proof: Note that, because the above is a modified version of the one by Kleinbergs and Tardos, a modified version of their proof is also applicable. On the inner for loop on a particular vertex u we have $O(n_u)$, where n_u is the degree of u. So, for all the vertices V we have the total work in $O(\sum_{u \in V} n_u)$ where $\sum_{u \in V} n_u$ is known to be equivalent to $2m$ such that the work on the inner loop simplifies to $O(m)$. The only additional work is the addition of a boolean flag and if conditional, which does not change the asymptotic runtime from the unmodified version. Set up and the outer loop runs in $O(n)$ for a total time of $O(m+n)$. $\square$

Claim: modifiedBFS produces correct output, i.e. it returns a vertex that can be removed from the given graph without disconnecting it.

Proof: As noted we know that any tree's leaves will, by definition, have only one edge. We note further, that the removal of a vertex with only one edge cannot disconnect the graph it was removed from. Further, we note that a vertex is a leaf in the BFS tree when that vertex has no undiscovered neighbors and we can, therefore simply store a list of vertices with no undiscovered neighbors and return the last one found as the vertex to remove without disconnecting the graph. $\square$