• Some general things to think about/talk about:
  
  – What does the method do? Explain what the role of the different parameters is, what is returned and how the method operates.
  – Show some examples both with and without cycles.
  – What is the running time of the method with respect to $|V|$ the number of vertices and $|E|$ the number of edges?
    * How many times is each vertex visited? How much work is done per vertex?
    * How many times is each edge visited/examined?

• Some specific things to think about/talk about:

  – Why are the visited and adjMap passed by reference? Why can we pass the adjMap parameter as a const, but not the visited parameter?
  – What do the if statements in both methods do? How else could we write this?
  – Why do we need the parent parameter?
  – Why do we pass -1 as the parent in our call to dfs_hasCycles?
  – What does adjMap.find(v)->second do?
bool dfs_hasCycles(int v,
        int parent,
        set<int>& visited,
        const map<int,list<int> >& adjMap) {
    bool result = false;
    visited.insert(v);

    list<int> nbrList = adjMap.find(v)->second;

    for (list<int>::iterator nbr = nbrList.begin(); nbr != nbrList.end(); nbr++){
        if (visited.find(*nbr) != visited.end()){
            result = result || dfs_hasCycles(*nbr, v, visited, adjMap);
        }else if(*nbr != parent){
            // we've visited this node, but it's not our parent, i.e. where we just came from
            result = true;
        } // else it's the parent
    }

    return result;
}

bool grop_hasCycles(const map<int, list<int> >& adjMap) {
    set<int> visited;

    for (map<int,list<int> >::const_iterator miter = adjMap.begin();
        miter != adjMap.end(); miter++){
        int v = miter->first;

        if (visited.find(v) == visited.end()) {
            if (dfs_hasCycles(v, -1, visited, adjMap)){
                return true;
            }
        }
    }

    return false;
}