CS41 Homework 6

This homework is due at 11:59PM on Sunday, October 22. Note the usual date, the last day before classes resume after fall break. This is a shorter 7 point homework. You could turn it in before you leave! Write your solution using LaTeX. Submit this homework using **github** as a **.tex** file. This is a **partnered homework**. You should primarily be discussing problems with your homework partner.

It's ok to discuss approaches at a high level with others. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab teammate *while in lab*. In this case, note (in your **homework submission poll**) who you've worked with and what parts were solved during lab.

1. **Enemies on the Move.** Alice and Bob are very active students at University of Pennsylvania. They used to be best friends but now despise each other. Alice and Bob can't stand to be in the same room, or even nearby. However, they each take many classes and are active in several clubs. Is it even possible to avoid each other?

   This can be modeled as a graph problem. The input consists of:

   - a directed graph $G = (V, E)$,
   - an integer $k \leq n$,
   - start vertices s_A and $s_B \in V$, and
   - end vertices t_A and $t_B \in V$.

   In this problem, Alice starts at s_A and wants to travel to t_A, while Bob starts at s_B and wants to travel to t_B. At each time step, either Alice or Bob moves along a single edge. (You can assume they move separately.) At all times, Alice and Bob must be at least k edges apart.

   Design and analyze a polynomial-time algorithm that determines whether Alice and Bob can get where they want to go while maintaining distance.

   (Hint: You may want to write up a subroutine to find all-pairs shortest paths, which we discussed in lab, and use that subroutine here.)

2. **Cycle Detection.** (K&T 3.2) Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. Otherwise, your algorithm should output NO. Your runtime should be $O(n+m)$ for a graph with m edges and n vertices.

   Note: Don't forget edge cases. Don't forget to return the cycle if one is detected.

   (a) Write up pseudocode for your algorithm. Prove that it halts, runs in polynomial time, and is correct.

   (b) Code your algorithm. You have been provided some helper files, which you do not need to edit:

   - **graph.h** specifies the class **Graph**
   - **edge.h** specifies the class **Edge**

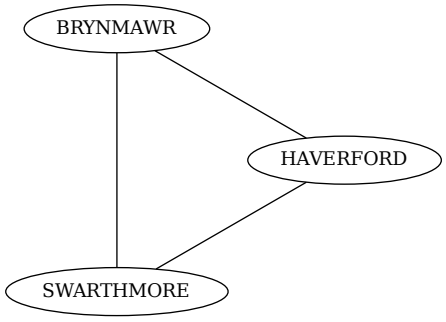
- **dictionary.h** and **stlHashTable.h** are the behind-the-scenes details used in CS35 to keep track of the edges in a graph
- **adjacencyListGraph.h** and **adjacencyListGraph.cpp** provide an interface and implementation for the adjacency list representation of a directed graph
- **adjacencyListUndirectedGraph.h** and **adjacencyListUndirectedGraph.cpp** provide an interface and implementation for the adjacency list representation of an udirected graph
- **detectCycle.cpp** is a simple program to run the detectCycle function
- **graphAlgorithms.h** is a header file declaring the graph algorithm functions reachableBFS, reachableDFS, and detectCycle

You need to edit ONLY the file **graphAlgorithms.cpp** in order to implement the detectCycle function. The file already contains implementations of standard graph algorithms BFS and DFS, as you saw them in CS35.

Some test data has been provided for you. Graphs are given to the program as a file consisting of 4 items: n, the list of vertices, m, and the list of edges. For example, the file:

```
3
SWARTHMORE
BRYNMAWR
HAVERFORD
3
SWARTHMORE BRYNMAWR
BRYNMAWR HAVERFORD
SWARTHMORE HAVERFORD
```

. . . represents the graph:



In the provided test_data folder there is a script you can run to randomly generate test graphs. Of course you can also write your own test graphs (particularly to check corner cases!). You are not required to submit any tests.