CSC 325 Algorithms

Spring 2015

Assignment 5

Due Date: April 22, 2015 (Wednesday) at the beginning of class

Write a program to solve Exercise 22.2-7 on page 602, which is equivalent to determining whether an undirected graph is bipartite.

Read one or more pairs of wrestler names from standard input. Names are separated by whitespace, so you can read them as strings using the normal >> operator. Each pair represents a rivalry. You can assume that the “rivalry graph” represented by the input is connected. Here is a sample input:

fang sally

shorty zephyr

the\_machine big\_earl

percival shorty

fang the\_machine

sally shorty

You must use a single breadth-first search to determine if the wrestlers can be partitioned into babyfaces and heels. (Hint: BFS/distances/even-odd.) To ensure that the output is unique, you must use the first wrestler in the first rivalry as the source for the breadth-first search; that first wrestler must be designated a babyface if a partition exists. Output the babyfaces and heels in sorted order to standard output, using the exact formats shown below.

BABYFACES

big\_earl

fang

shorty

HEELS

percival

sally

the\_machine

zephyr

A partition exists for this sample input but may not exist in general. If no partition exists, output the following single line:

IMPOSSIBLE

Your program must meet both of the following criteria:

1. It must store the graph using an adjacency-list representation, not an adjacency matrix. You could assign a unique number to each name and then use a vector of adjacency lists, but you might find it easier to use an unordered\_map of adjacency lists, since all the nodes are identified by strings. You can represent each individual adjacency list as a forward\_list or list or vector.
2. It must run in time O(m + n), not counting the time it takes to sort the output, where n is the number of wrestlers and m is the number of rivalries. (Sorting the names will take an additional O(n lg n) time, of course, but you can use C++'s sort for that.)

**Submission:**

1. Upload an electronic copy of your source code to your CSC 325 TRACE folder [\\trace\Class\CSC-325](file:///\\trace\Class\CSC-325).
2. Bring a stapled hardcopy of your source code to class on the due date. Please remember to write your name and the name of your TRACE folder.