CS 341: Algorithms @ Waterloo **Last Updated:** Jan 25, 2016
Homework 2 **Due Date: Feb 8, 2016, 11:59pm**

The full mark is 50. This homework is counted 5% of the course grade.

For written problems, please write pseudocode for the algorithms, prove their correctness and analyze their time complexity.

---

1. **Programming Problem: Cut Vertices and Cut Edges** (18 marks)

   Implement an $O(m + n)$ time algorithm for finding all cut vertices and cut edges in an undirected graph.

   Input: The first line will have two positive integers, n and m, which represent the number of vertices and the number of edges in the graph respectively, where the vertex set of the graph is $\{1, \ldots, n\}$. Then, there will be m lines following, one for each undirected edge. Each line will have two numbers in $\{1, \ldots, n\}$, representing the two vertices of the edge. You can assume that the input graph is connected. You can also assume that $1 \leq n \leq 10000000$ and $1 \leq m \leq 30000000$.

   Output: The output will consist of two lines. In the first line, print the number of cut vertices, and then list all the cut vertices in increasing order. (If zero, then the list is empty.) In the second line, print the number of cut edges, and then list all the cut edges in lexicographical order. (If zero, then the list is empty.) See the sample output for the precise format.

   Sample Input:
   ```
   5 5
   1 2
   1 3
   5 4
   4 3
   3 5
   ```
   Sample Output:
   ```
   2 1 3
   2 (1,2) (1,3)
   ```

2. **Written Problem: Sorting by Reversals** (8 marks)

   A permutation is an array A of n distinct numbers from 1 to n. A reversal on the permutation takes an interval on the array, and inverts the elements' order in that interval. For example, the reversal of interval $[3, 5]$ transforms the permutation (4,1,2,6,3,5) to (4,1,3,6,2,5). Any permutation A of size n can be sorted to $(1, 2, \ldots, n)$ by a sequence of reversals. For a given permutation, one would like to find the shortest reversal sequence to sort the permutation.

   The problem is useful in bioinformatics, where each number in the permutation indicates a gene (or a conserved stretch on the genome). Two genomes of closely related species may have the same set of genes but the genes are in different sequential orders on the genome. Biologists know that reversal is a common type of genome-rearrangement events. Thus, the least number of reversals needed to convert one genome to another becomes one of several common genome distances used in bioinformatics. Unfortunately this problem is an NP-hard problem so there is no polynomial time algorithm (unless

P=NP). But when both n and the distance is small, an exponential time algorithm may be good enough.

Show how to model this problem as a graph problem. Then describe an algorithm with time complexity $O(\min\{n^{2k}, n! \cdot n^2\})$ to find the shortest sequence of reversals. Here n is the size of the permutation and k is the least number of required reversals.

3. **Written Problem: Number of Shortest s-t Paths** (12 marks)

   Given an undirected graph $G = (V, E)$ and two vertices s and t, design an algorithm to return the number of different shortest s-t paths between s and t. Two paths are the same if they have exactly the same sequence of vertices; otherwise they are different. (You may imagine that this is a measure of how close are two people in a social network.)

   You will get full marks if the time complexity of the algorithm is $O(|V| + |E|)$ word operations and the proofs are correct. (We assume that the answer fits in one word, so that each operation of adding two numbers can be done in one word operation.)

4. **Written Problem: One-Way Streets** (12 marks)

   Imagine in a city where all the streets are two-way streets. Perhaps because the streets are too narrow, the government would like to see if it is possible to make all streets one way, so that there is still a way to drive from any street to any other street in the city.

   Show how to model this problem as a graph problem. Then design an algorithm to determine if it is possible, and if possible to find one solution to assign the directions.

   You will get full marks if the time complexity of the algorithm is $O(n + m)$ and the proofs are correct, where m is the number of streets in the city and n is the number of intersections in the city.

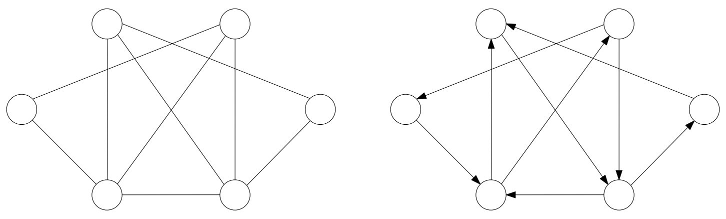

Figure 1: There is a solution in this example.

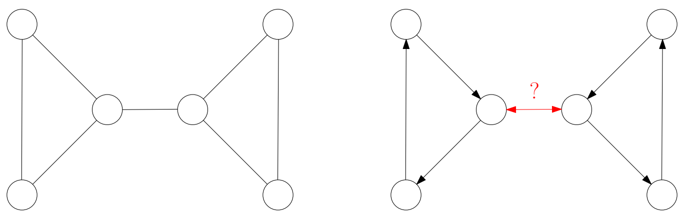

Figure 2: There is no solution in this example.