Computer Science CSC263H
St. George Campus

March 16, 2017
University of Toronto

Homework Assignment #5

Due: March 30, 2017, by 5:30 pm

- You must submit your assignment as a PDF file of a typed (**not** handwritten) document through the MarkUs system by logging in with your CDF account at:

  `markus.teach.cs.toronto.edu/csc263-2017-01`

  To work with one or two partners, you and your partner(s) must form a group on MarkUs.

- If this assignment is submitted by a group of two or three students, the PDF file that you submit should contain for each assignment question:

  1. The name of the student who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.

- The PDF file that you submit must be clearly legible. To this end, we encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.

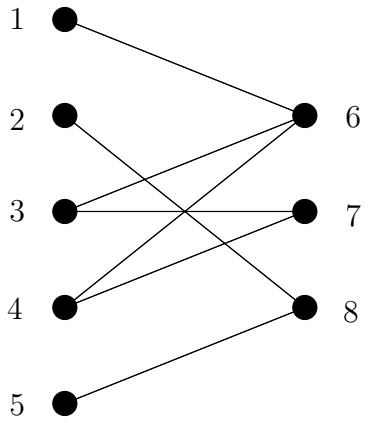
- By virtue of submitting this assignment you (and your partners, if you have any) acknowledge that you are aware of the homework collaboration policy that is stated in the csc263 course web page: `http://www.cs.toronto.edu/~sam/teaching/263/#HomeworkCollaboration`.

- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbook, by referring to it.

- Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision, and conciseness of your presentation.

Question 1. (12 marks) Prove that every connected undirected graph G contains at least one vertex whose removal (along with all incident edges) does *not* disconnect the graph. Give an algorithm that for any connected graph $G = (V, E)$, given by its adjacency list, finds such a vertex in $O(|V| + |E|)$ time in the worst case. Justify your algorithm's correctness and worst-case time complexity.

Question 2. (24 marks) Let $G = (V, E)$ be an undirected graph. G is ***bipartite*** if the set of nodes V can be partitioned into two subsets V_0 and V_1 (i.e., $V_0 \cup V_1 = V$ and $V_0 \cap V_1 = \varnothing$), so that every edge in E connects a node in V_0 and a node in V_1. For example, the graph shown below is bipartite; this can be seen by taking V_0 to be the nodes on the left and V_1 to be the nodes on the right.



Note that G is bipartite if and only if every connected component of G is bipartite.

a. (8 marks) Prove that if G is bipartite then it has no *simple cycle* of odd length.[1] Hint: Give a proof by contradiction.

b. (8 marks) Prove that if G has no simple cycle of odd length (i.e., every simple cycle of G has even length) then G is bipartite. (Hint: Suppose every simple cycle of G has even length. Perform a BFS starting at any node s. Assume for now that G is connected, so that the BFS reaches all nodes; you can remove this assumption later on. Use the distance of each node from s to partition the set of nodes into two sets, and prove that no edge connects two nodes placed in the same set.)

c. (8 marks) Describe an algorithm that takes as input an undirected graph $G = (V, E)$ in adjacency list form. If G is bipartite, then the algorithm returns a pair of sets of nodes (V_0, V_1) so that $V_0 \cup V_1 = V$, $V_0 \cap V_1 = \varnothing$, and every edge in E connects a node in V_0 and a node in V_1; if G is not bipartite, then the algorithm returns the string "not bipartite." Your algorithm should run in $O(n + m)$ time, where $n = |V|$ and $m = |E|$. Explain why your algorithm is correct and justify its running time. (Hint: Use parts (a) and (b).)

Question 3. (24 marks) You are riding your bike and you have an aerial map of the routes you could take. This map contains the (x, y)-coordinates of n bike pump stations. There is a straight bikeway that goes directly between any two bike pump stations. You want to travel from station s to station t. Since your tires aren't very good, the best route from s to t is one that minimizes the distance that you have to travel between any two successive stations on this route. Your task is to design an $O(n^2 \log n)$-time algorithm for finding a best route.

a. Restate this task as a graph problem. To do so: (i) describe the graph: its vertices, edges, and edge weights, and (ii) restate the task as a graph problem on this graph.

b. We learned several algorithms that construct trees from an input graph. One these trees can be used to find the desired path efficiently. Explain how and prove it.

c. Describe a corresponding algorithm for solving this problem in plain and clear English.

d. Explain why the worst-case running time of your algorithm is $O(n^2 \log n)$.

NOTE: your algorithm can use any algorithms that we studied in the course as "black-boxes," and you can refer to their worst-case time complexity as stated in lecture or in the textbook.

[1] Recall that a cycle is simple if it contains each node at most once.