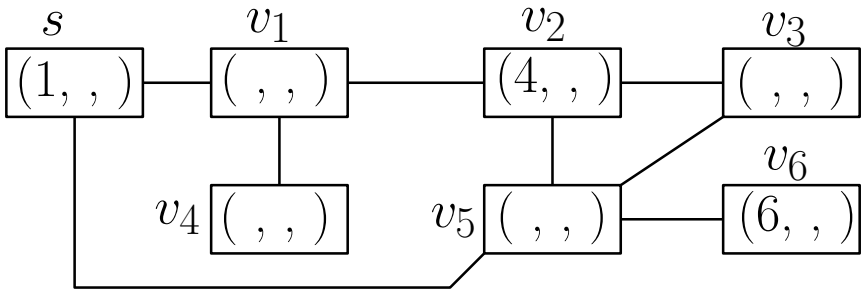
COMP160: Algorithms, **Homework 9**

- To obtain full credit, you must justify your answers. When describing an algorithm, do not forget to **state any assumptions that you make**, analyze its running time and explain why the algorithm is correct.
- Although not specifically stated, you can assume that we look for algorithms that are as fast as possible, and bounds as tight as possible.
- You may discuss these problems with others, but remember to write the answers on your own. In case of doubt, cite any source you used to do the assignment.
- Remember to submit each question in a separate page.

1. The algorithm FindAllCutVertices was run on the following graph but then most of the answers were lost, leaving only the few numbers shown:



   (a) Indicate which edges are part of the DFS tree. Orient edges back towards the source.

   (b) For each vertex, fill in the missing numbers ($v.\texttt{disc}, v.\alpha, v.\beta$).

   (c) For each cut vertex, give the specific condition (using numbers from the graph) that qualifies it as a cut vertex. Does this match with your intuition?

2. You are given a weighted undirected and connected graph $G_1 = (V, E)$ (with n vertices and m edges). We are also given its MST T_1. Suppose that we add a vertex v to G_1, along with some weighted edges from v to other vertices in G (at least 1 edge is added and at most n of them). Let G_2 be the modified graph and T_2 its modified MST.

   (a) (warm-up) What is a simple method we can use to compute the MST of G_2? What is the runtime? Justify your answer with 1-2 sentences.

   (b) Now let's try to make the algorithm faster. If it needs to beat standard algorithms we must use the extra information we have. Show how to compute T_2 using G_1, T_1 and G_2 (those three input graphs are given to you in any format you want).

   Do not forget to justify correctness and runtime of your algorithm (as a function of n and m). You may assume that all edge weights are distinct.

   Hint: rather than going straight to the algorithm, try to prove some property that will allow you to skip checking many edges (say, something of the form *the MST of G_2 will not have edges whose weight is a prime number*).

   Note: As usual, points are given for efficiently as well as giving a well structured proof.

3. Consider a weighted graph $G = (V, E)$ where all edge weights are integers in the range $(1, \ldots, k)$, for some $k = O(1)$. Assuming an MST of G exists, show how fast can you compute it as a function of n and m.

   Note: The naive solution runs in $O(m \log n)$ time. Ideally, you can produce a solution that runs faster. There is at least one solution you can write in a short paragraph using tools covered only in this course. For an extra challenge, try to find a linear time algorithm.