Worth: 7.5% (*best four of the five assignments*) **Due:** *before* 6:00pm on Wed. 3 April

Required filename for MarkUs submission: `a5.pdf` **Remember to write the *full name* and *MarkUs username* of *every* group member (up to three) prominently on your submission.**

---

*Please read and understand the policy on Collaboration given on the Course Information Sheet. Then, to protect yourself, list on the front of your submission **every** source of information you used to complete this homework (other than your own lecture and tutorial notes). For example, indicate clearly the **name** of every student from another group with whom you had discussions, the **title and sections** of every textbook you consulted (including the course textbook), the **source** of every web document you used (including documents from the course webpage), etc.*

*For each question, please write up detailed answers carefully. Make sure that you use notation and terminology correctly, and that you explain and justify what you are doing. Marks **will** be deducted for incorrect or ambiguous use of notation and terminology, and for making incorrect, unjustified, ambiguous, or vague claims in your solutions.*

---

1. **Graph Colouring.** Consider the family of decision problems "k-COLOUR," for positive integers k, defined as follows. (Note that this defines infinitely many decision problems, one for each k).

   Input: Undirected graph $G = (V, E)$.

   Question: Is it possible to assign a colour to each vertex of G (where the same colour can be assigned to more than one vertex), so that every edge of G has endpoints with different colours and no more than k different colours are used?

   For example, if G is a cycle on 5 vertices, the answer is "no" for the 2-COLOUR problem but "yes" for the 3-COLOUR problem.

   Assume that 3-COLOUR is *NP*-complete.

   Write a detailed proof that k-COLOUR is *NP*-complete for all $k > 3$.

2. **Graph Colouring, continued.** Prove that 3-COLOUR is polynomial-time self-reducible.

   Start by writing down a precise definition of the search problem you are trying to solve, and explain what you are doing (and to what purpose) at each stage. Write your algorithms in pseudo-code (with comments to make your intentions clear), and remember to include a detailed argument of correctness and runtime analysis.

   HINT: For any undirected graph $G = (V, E)$ and any vertices $u, v \in V$, let G_{uv} be the graph obtained by merging u and v into a single vertex (and getting rid of duplicate edges). For example, if $G = (V, E)$ with $V = \{a, b, c, d, e\}$ and $E = \{(a, b), (b, c), (c, d), (d, e), (e, a)\}$, then $G_{be} = (V_{be}, E_{be})$ with $V_{be} = \{a, be, c, d\}$ and $E_{be} = \{(a, be), (be, c), (c, d), (d, be)\}$. Think about how the colourability of G relates to the colourability of G_{uv} for pairs $u, v \in V$ that are **not** adjacent (i.e., $(u, v) \notin E$).

THERE WILL BE ONE MORE QUESTION ON APPROXIMATION ALGORITHMS, COMING SOON IN PART II!