COMP3121/9101
Algorithm Design

Tutorial 9
Intractability

Before the Tutorial

Before coming to the tutorial, try and answer these discussion questions yourself to get a firm understanding of how you're pacing in the course. No solutions to these discussion points will be officially released, but feel free to discuss your thoughts on the forum. Your tutor will give you some comments on your understanding.

- Explain what it means for a problem to be in the following classes: P, NP, NP-hard, NP-complete.
- What does it mean for an algorithm to run in *polynomial time*?
- Read through the problem prompts and think about how you would approach the problems.

Tutorial Problems

Problem 1. Let $G = (V, E)$ be a directed and edge-weighted graph, where V is the set of vertices and E is the set of edges. Consider two vertices $s, t \in V$. Recall that the minimum cut problem is the problem of computing the capacity of the smallest cut that completely disconnects s and t.

(a) Explain why the *minimum cut* problem is *not* in P.

(b) Convert the *minimum cut* problem into a new problem that is in P. From now on, we will call this new problem the *minimum cut* problem instead.

- Is there a known algorithm that solves a similar problem?

(c) We now consider a generalisation of the minimum cut problem. Instead, given two subsets of vertices S, T and an integer k, define the *subset minimum cut* problem to be the problem of determining whether there exist a cut in G of capacity at most k that completely disconnects all vertices in S from all vertices in T; that is, no vertex in S can be connected to a vertex in T.

Given an instance of the *subset minimum cut* problem, transform the instance into an instance of the *minimum cut* problem.

- How should you convert the subsets $S, T \subseteq V$ into two vertices $s, t \in V$?

(d) Solve the corresponding *minimum cut* problem and explain how that solves the *subset minimum cut* problem.

(e) Hence, show that the *subset minimum cut* problem is also in P.

Problem 2. In the *dynamic programming* topic, you saw an algorithm for the *Longest Increasing Subsequence* (LIS) and the *Longest Common Subsequence* problems. Here, we will give a direct relationship between LIS and LCS. Recall the problems below.

- **Longest Increasing Subsequence** (LIS): Given an integer sequences A of length n, find the longest increasing subsequence of A.
- **Longest Common Subsequence** (LCS): Given two sequences A and B of lengths m and n respectively, find the longest subsequence common to both A and B.

Suppose that there exist an algorithm that solves LCS in $T(N)$ time, where $N = \max(m, n)$.

(a) Give an $T(n) + O(n \log n)$-time algorithm that solves LIS by converting any instance of LIS to an instance of LCS.

(b) Briefly explain why your algorithm is correct.

Problem 3. Let A and B be two problems in class NP, and suppose that f is a polynomial-time reduction from A to B. In each scenario, what further information can you deduce about problems A and B?

(a) A is a problem in class P.

(b) A is a problem in class NP-hard.

(c) B is a problem in class P.

(d) B is a problem in class NP-hard.

After the Tutorial

After your allocated tutorial (or after having done the tutorial problems), review the discussion points. Reflect on how your understanding has changed (if at all).

- In your own time, try and attempt some of the practice problems marked [K] for further practice. Attempt the [H] problems once you're comfortable with the [K] problems. All practice problems will contain fully-written solutions.
  - To best prepare for the final exam, ensure that you are comfortable with answering the problems marked [K].
  - If you have time, attempt the problems marked [H] as these problems are targeted towards high-achieving students in COMP3121/9101.
  - Attempt the problems marked [E] if you are comfortable with the problems marked [H] and have time.
- It is also worthwhile to look over your assignment problems and understand how to formulate a response worth full marks.