Monash University
Faculty of Information Technology
2^{nd} Semester 2018

FIT2014
Tutorial 7
Complexity: NP and Polynomial Time Reductions

This tutorial gives valuable practice on NP, polynomial-time reductions, and proof by induction, and will help you prepare for questions of that kind on the final exam.

ASSESSED PREPARATION: Question 3.
You must present a serious attempt at this entire question to your tutor at the start of your tutorial.

---

1. A *k-edge-colouring* of a graph G is a function that assigns to each edge a colour from the set $\{1, 2, \ldots, k\}$ such any two edges that have an endpoint in common receive different colours.

Consider the following problem.

EDGE-COLOURING
Input: Graph G, positive integer k.
Question: Does G have a k-edge-colouring?

(a) Prove that EDGE-COLOURING $\in$ NP.

(b) Can you find a value of k such that, if we restrict to that particular k, the problem can be solved in polynomial time?

2. A **short verifier** is a verifier whose certificate consists of ≤ 100 characters.

Let NP[SHORT] be the class of languages with polynomial-time *short* verifiers.

(a) Prove that P = NP[SHORT].

Challenge:

More generally, for any function $f : \mathbb{N} \cup \{0\} \to \mathbb{N} \cup \{0\}$, define NP$[f(n)]$ be the class of languages accepted by polynomial-time verifiers whose certificates consist

of $\leq f(n)$ characters, where n is the input size. So, for example, NP[100] is just NP[SHORT]. In that case the function is just the constant function whose value is always 100. But think about what can happen when f is *unbounded*, i.e., there is no constant b such that $f(n) \leq b$ for all n.

(b) Can you find an *unbounded* function f such that P = NP[$f(n)$]?

(c) Can you find an *unbounded* function f such that NP = NP[$f(n)$]?

3.

(a) Prove that NEARLY SAT belongs to NP.

NEARLY SAT
Input: Boolean expression φ in CNF.
Question: Is there a truth assignment that satisfies all, or all except one, of the clauses of φ?

(b) Give a polynomial-time reduction from SAT to NEARLY SAT and prove that it is such a polynomial-time reduction.

(c) What can we say about NEARLY SAT, given (a) and (b)?

(d) Suppose we alter the Question in the problem definition so that the truth assignment has to satisfy *all except* k of the clauses, where k is some *fixed* number (i.e., it's fixed in advance, and *not* part of the input). Would the problem still be in NP? Could you still give a polynomial-time reduction from SAT to this new problem?

4.

(a) Prove that MOSTLY SAT belongs to NP.

MOSTLY SAT
Input: Boolean expression φ in CNF.
Question: Is there a truth assignment that satisfies at least three-quarters of the clauses of φ?

(b) Give a polynomial-time reduction from SAT to MOSTLY SAT, and prove that it is such a polynomial-time reduction.

(c) What can we say about MOSTLY SAT, given (a) and (b)?

The following information relates to Questions 5–7.

(Questions 5 and 6 are from the final exam in 2014.)

A **Hamiltonian path** in a graph G is a path that includes every vertex of G. All the vertices on the path must be distinct.

A **Hamiltonian circuit** in a graph G is a circuit that includes every vertex of G. All the vertices on the circuit must be distinct.

The decision problems HAMILTONIAN PATH and HAMILTONIAN CIRCUIT are defined as follows:

HAMILTONIAN PATH
Input: Graph G.
Question: Does G have a Hamiltonian path?

HAMILTONIAN CIRCUIT
Input: Graph G.
Question: Does G have a Hamiltonian circuit?

5.

Prove that HAMILTONIAN CIRCUIT belongs to NP.

6.

Give a polynomial-time reduction (a.k.a. polynomial transformation) from HAMILTONIAN PATH to HAMILTONIAN CIRCUIT.

7.

Assuming you know that HAMILTONIAN PATH is NP-complete, what can you say about HAMILTONIAN CIRCUIT from your solutions to Questions 5 and 6?

The following information relates to Questions 8–11.

Before attempting these questions, it is recommended that you revise previous Tutorial exercises about reduction to SATISFIABILITY. These are: Tutorial 1, Q8; Tutorial 5, Q9; Tutorial 6, Q10. If your solutions to any of these were not in Conjunctive Normal Form (CNF), then think about how you would convert them to CNF. See also the notes by FIT2014 tutor Chris Monteith (available under week 3).

Suppose you have a set $T \subseteq A \times A \times A$ of triples, with each element of each triple belonging to some set A of n elements. A *three-dimensional matching* is a subset, $S \subseteq T$, consisting of exactly n triples from T, such that all members of S are disjoint.

These requirements ensure that every member of A appears as a *first* member of some triple in S, and also as the *second* member of some triple in S, and also as the *third* member of some triple in S.

For example, suppose that $A = \{1, 2\}$, and $T = \{(1, 1, 1), (1, 1, 2), (1, 2, 1), (2, 2, 1)\}$. Then T has the 3D matching $S = \{(1, 1, 2), (2, 2, 1)\}$.

If, instead, $T = \{(1, 1, 1), (1, 1, 2), (1, 2, 1), (2, \mathbf{1}, 1)\}$, then T has no 3D matching.

Let 3DM be the language consisting of every set T (consisting of triples) that has a 3D matching.

8.

Prove that 3DM belongs to NP.

9.

Give a polynomial-time reduction (a.k.a. polynomial transformation) from 3DM to SATISFIABILITY.

This should be described as an algorithm.

Show that it is runs in polynomial-time.

10.

Now suppose that $T \subseteq A \times A \times A \times A$ is a set of *4-tuples*, where again $|A| = n$. A *four-dimensional matching* is a subset, $S \subseteq T$, consisting of exactly n *4-tuples* from T, such that all members of S are disjoint.

Let 4DM be the language consisting of those sets T of 4-tuples that have a 4D matching.

Give a polynomial-time reduction from 3DM to 4DM.

Prove that it is a polynomial-time reduction.

11.

For any fixed positive integer k, we can define the language kDM along similar lines: T is a set of k-tuples $(x_1, x_2, \ldots, x_k)$, where each $x_i \in A$ and $|A| = n$; a *k-dimensional matching* is a set of n disjoint members of T; and kDM is the set of all such T that have a k-dimensional matching.

Suppose you have proved that, for all $k \geq 1$, there is a polynomial-time reduction from kDM to $(k + 1)$DM. Use induction to prove that, for all $k \geq 1$ and all $\ell \geq k$, there is a polynomial-time reduction from kDM to ℓ DM.