1 Introduction

Just as it was for decidability, *reducibility* is a key idea in analysing efficiency. In the context of efficiency rather than computability, reduction of instances of PROB1 to instances of PROB2 means that if we can solve PROB2 efficiently, then we can also solve PROB1 efficiently. Fundamentally this means that the reduction must preserve the "type" of each instance (i.e., whether the answer should be "yes" or "no") and the reduction itself must be an efficient procedure.

Before formalising that though, let's look at a few more examples of problems.

2 Problems on graphs

A particularly rich source of interesting algorithmic problems come from graphs. Recall that, for our purposes, a (simple) graph, G, consists of a set, V of *vertices*, and a set E of *edges*, each of which is an unordered pair of distinct elements of V. The elements of an edge are called its *endpoints* and if two vertices are the endpoints of an edge they are said to be *adjacent*.

Note that to represent a graph having n vertices requires listing the set of edges. However, there are at most $n(n-1)/2$ edges and so the total amount of space required to describe a graph with n vertices is $O(n^2)$.

- An *independent set* in a graph is a subset, $I \subseteq V$, such that no pair $\{v, w\}$ with $v, w \in I$ is an edge.
- A *clique* in a graph is a subset, $K \subseteq V$, such that every pair $\{v, w\}$ with $v, w \in I$ is an edge.
- A *walk* in a graph is a sequence $v_0, v_1, \ldots, v_k$ of vertices such that each pair $\{v_i, v_{i+1}\}$ for $0 \leqslant i < k$ is an edge. The *length* of a walk is the number of edges it traverses (i.e., k, one less than the number of vertices). A *path* is a walk containing no repeated vertices.

- A *cycle* in a graph is a walk $v_0, v_1, \ldots, v_k$ with $v_0 = v_k$ and containing no other repeated vertices
- A k-colouring of a graph is a function $c : V \to \{1, 2, \ldots, k\}$ with the property that for each edge $e = \{v, w\}$, $c(v) \neq c(w)$. That is, the endpoints of each edge get different colours.

Graphs (and small variations on the same theme) are flexible and ubiquitous models that arise in all sorts of situations. They also have lots of naturally associated decision problems. For instance:

INDEPENDENT-SET
Instance: A graph G and a positive integer k
Problem: Is there an independent set in G having (at least) k elements?

CLIQUE
Instance: A graph G and a positive integer k
Problem: Is there a clique in G having (at least) k elements?

HAS-PATH
Instance: A graph G and a pair of vertices $v, w \in G$
Problem: Is there a path in G from v to w?

IS-CONNECTED
Instance: A graph G
Problem: Is the graph G connected, i.e., is there a walk in G between any two vertices?

HAS-CYCLE
Instance: A graph G
Problem: Is there a proper cycle in G?

HAMILTON-CYCLE
Instance: A graph G
Problem: Is there a cycle in G that visits every vertex?

HAMILTON-PATH
Instance: A graph G
Problem: Is there a path in G that visits every vertex?

k-COLOURING
Instance: A graph G
Problem: Is there a k-colouring of G?

Why not include k as a parameter in the last problem, i.e., as part of the instance? Well, we could, but we have our reasons.

Some of these problems are (or seem to be) hard, while others are easier. Perhaps you remember which are which from COSC 201?

In fact, all of HAS-PATH, IS-CONNECTED, HAS-CYCLE, and 2-COLOURING are in **P**. As for the rest ...

3 Polynomial-time reducibility

How can we compare the difficulty of problems that might or might not be in **P** (or **NP**)? We can modify the idea of Turing-reducibility that we used to show that certain problems were undecidable (because if we could decide them we could decide HALT) in a way that incorporates the time limits that **P** or **NP** impose.

A *polynomial-time reduction* of PROB to L is an algorithm for converting instances of PROB to words in Σ^* such that the time required for the conversion is bounded by a polynomial and such that all affirmative instances of PROB are converted to elements of L while negative instances are converted to elements not in L.

Let's consider some polynomial-time reductions. First a pretty trivial one:

Claim. There is a polynomial-time reduction from INDEPENDENT SET to CLIQUE

We need to show how to convert instances of INDEPENDENT SET to instances of CLIQUE. An instance of INDEPENDENT SET is a pair (G, k) consisting of a graph and a positive integer. We can define a graph $\bar{G}$ which has the same vertices of G but such that every edge of G is a non-edge in $\bar{G}$ and vice versa. But independent sets in G become cliques in $\bar{G}$ so (G, k) is a positive instance for INDEPENDENT SET if and only if $(\bar{G}, k)$ is a positive instance for CLIQUE. Since the transformation $G \to \bar{G}$ is easily accomplished in polynomial-time, this gives the required reduction.

The point here is that in some sense independent sets and cliques are just two versions of the same thing (switching the notion of edges and non-edges).

Claim. There is a polynomial-time reduction from k-COLOURING to $(k+1)$-COLOURING for any positive integer k.

This is a bit trickier. Given G we will define a new graph G^+ such that G^+ has a $(k+1)$-colouring if and only if G has a k-colouring. The idea is to add one more vertex, adjacent to every vertex of the original graph. Now if the new graph is $(k+1)$-colourable, the new vertex must be given some colour. But then no other vertex can get that colour since they're all adjacent to the new vertex so there are only k colours remaining to colour the rest of the graph. In other words, just as we require G^+ is $(k+1)$-colourable if and only if G is k-colourable.

Let's do one more.

Claim. There is a polynomial-time reduction from HAMILTON-CYCLE to HAMILTON-PATH.

This one's a bit trickier. We want to convert a graph G to a graph G^* in such a way that G^* has a Hamiltonian path if and only if G has a Hamiltonian cycle. Choose any vertex v of G. If G has a Hamiltonian cycle then any such must pass through v at some point. Add three new vertices x, y and z. The first, x is adjacent only to v and the second, y is adjacent only to all the neighbours of v, and the third z is adjacent only to y. That's the graph G^*. If G was Hamiltonian, then G^* has a Hamiltonian path – from x to v, around the cycle until you're just about to return to v and thence to y and on to z. On the other hand if G^* has a Hamiltonian path, then since x and z have only a single neighbour each it must start at one (say x – it doesn't matter) and finish at z. The second last vertex before z must be y and the one before that must be a neighbour, w, of v. Now follow the same path in G except ignore the xv start, and when you get to w go on to v. That's a Hamilton cycle in G.

There's an easier way to do this if we allow a more complicated notion of reducibility. Consider the set of graphs we can build from a graph G by "splitting" each edge. That is, for each edge $e = \{v, w\}$, form a new graph G_e that has two additional vertices v' and w', two additional edges $\{v', v\}$ and $\{w', w\}$ and remove the edge e. Now, if this graph has a

Hamilton-path it must start at v' and end at w' (or vice-versa). But, this can happen only if G had a Hamilton cycle using the edge e. So here we haven't reduced one instance of HAMILTON CYCLE to one instance of HAMILTON PATH, but rather to polynomially many instances in such a way that if any one of them is affirmative then so was the original, and otherwise not.

This last is an instance of what's called a *truth-table reduction*, rather than a *many-one reduction* which is the one we introduced earlier (also called a *Karp reduction*). There is an even more general notion of reduction called a *Cook reduction*. See the wikipedia page on polynomial-time reduction (and links therein) for more details. For any of these reductions we have:

Theorem 3.1. *If there is a polynomial-time reduction from* PROBA *to* PROBB *and* PROBB *belongs to* **P** *(respectively* **NP***), then* PROBA *belongs to* **P** *(resp.* **NP***).*

The "proof" is "to solve a problem from PROBA use the polynomial-time reduction to produce an instance (or instances) of problem(s) in PROBB and then use the (non-deterministic) polynomial-time algorithm available to solve it (those)."

4 NP-hard

Suppose we could find some problem REALLYHARD which had the property that for every problem, PROB in **NP** there was a polynomial-time reduction from PROB to REALLYHARD. Then, that says REALLYHARD is a really hard problem, since any efficient algorithm for solving it would resolve *every* problem in **NP**. We say that such problems are **NP**-hard. If, in addition, they happen to belong to **NP**, then we say they're **NP**-complete.

Conversely, any reduction of REALLYHARD to some other problem also shows that other problem must be really hard.

Problem: Do such languages exist?

Answer: **YES**! And much more – *many natural problems in optimisation and search are* **NP***-complete*. For example, FAIR-DIVISION is such a problem, as are all the graph problems listed above except HAS-PATH, HAS-CYCLE, IS-CONNECTED and 2-COLOURING.

5 Tutorial problems

0. Draw some graphs. Make sure you're happy with notions like walk, path, cycle etc.
1. What is the least number of edges that a connected graph with n vertices can have?
2. (*) What is the greatest number of edges that a graph on n vertices containing no triangle (clique of size 3) can have?
3. Show that in a graph G if there is a walk from v to w then there is also a path from v to w.
4. Find a graph with five vertices that has no clique nor independent set of size 3. Doe such a graph exist with six vertices?
5. Suppose that we have a polynomial-time reduction from PROB1 to PROB2.
   (a) If the time required to convert an instance of size n is bounded by n^4, then how large an instance of PROB2 might we get from an instance of size n for PROB1?
   (b) If PROB2 $\in \mathbf{P}$ and the time required to resolve an instance of size n of PROB2 is bounded by n^3, what is an upper bound for the time required to resolve an instance of PROB1 through the reduction?
6. Confirm that HAS-PATH, IS-CONNECTED, HAS-CYCLE, and 2-COLOURING are in $\mathbf{P}$.