Exercises 9

12 March

All questions are unassessed.

1. The Hamiltonian cycle problem HAMCYCLE is: Given a graph G, does G have a Hamiltonian cycle?
(a) Show that HAMCYCLE is in NP.
(b) Show that HAMCYCLE is NP-hard. You may assume that HAMPATH is NP-hard (lecture notes page 79).
[Hint: Define an appropriate reduction by adding nodes and/or arcs to the graph.]

2. The problem COMPOSITE is as follows: Given a natural number n, is n composite (i.e. not a prime number)? Show that COMPOSITE is NP. Define the associated verification problem VER-COMPOSITE.

3. The problem GRAPHISOM is as follows: Given two graphs G_1, G_2, is there an isomorphism between G_1 and G_2? Show that GRAPHISOM is NP. Define the associated verification problem VER-GRAPHISOM.

4. Show that many-one reduction $\leq$ on decision problems (lecture notes pages 77-78) is (a) reflexive and (b) transitive.

5. Let D and D' be decision problems. Show that if D is NP-hard and $D \leq D'$ then D' is also NP-hard.

6. In the lectures we showed that both $\mathrm{TSP(D)}$ (slide 306) and $\mathrm{VRPC(D)}$ (slide 317) are NP-complete. Explain why $\mathrm{TSP(D)} \sim \mathrm{VRPC(D)}$ (slide 299).

7. (a) Show that exponentiation m^n of natural numbers can be carried out with only polynomially many multiplications and divisions. [Hints: Remember that the input size is $\log m + \log n$, not $m + n$. Try writing a recursive program which goes into cases depending on whether the exponent n is even or odd.]
(b) Given that multiplication and division are p-time, does this show that exponentiation is a p-time operation?

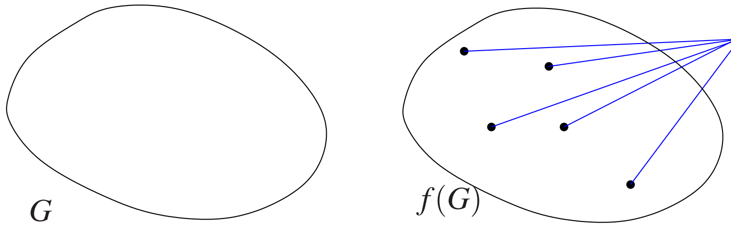
8. The exam timetabling problem is: given a set of exams E, sets of candidates L_e (each $e \in E$), and a set of times T, is there a timetable which has no clashes? Here a timetable is a function $f : E \to T$, and a clash is when a candidate has to take two exams which are scheduled at the same time.
Explain why the exam timetabling problem is in NP.

Answers to Exercises 9

1. (a) Much as for HAMPATH (lectures), we guess a sequence of nodes $x_1, \ldots, x_n$ and then check in p-time that it is a Hamiltonian cycle for G, that is,

- G has n nodes
- all x_i are distinct and nodes of G
- there is an arc in G between x_i and x_{i+1} ($i = 1, \ldots, n$) (including one between x_n and x_1)

(b) We show HAMPATH $\leq$ HAMCYCLE. Since HAMPATH is NP-hard, HAMCYCLE is also NP-hard (Question 5). We need a reduction function f such that G has a Ham path iff $f(G)$ has a Ham cycle. We do not know where the Ham path might start or finish and so we need a way of converting any possible Ham path into a Ham cycle. Let G have n nodes. Let f add a new node x to G, and join every node of G to x by n new arcs. This construction is p-time, as we add p-many nodes (only one) and arcs (n).



Suppose G has a Ham path with endpoints y, z. Then $f(G)$ has a Ham cycle where we add the arcs (y, x) and (x, z).
Conversely, suppose that $f(G)$ has a Ham cycle. Without loss of generality we can suppose that this starts at x and is of the form $x, y, \ldots, z, x$. We can omit node x and get $y, \ldots, z$, which is a Ham path for G.
We have shown HAMPATH $\leq$ HAMCYCLE.

2. Given a number n, we can guess a non-trivial factorisation m_1, m_2, and check in p-time that $m_1 * m_2 = n$ (multiplication is p-time). Note that the guesses m_1, m_2 are bounded in the input size. Hence COMPOSITE $\in$ NP.
Here we decided to guess m_1 and m_2. Alternatively we could simply guess a single possible factor m and check whether m divides n.
VER-COMPOSITE(n, m_1, m_2) iff $n = m_1 * m_2$ and $m_1, m_2 > 1$.
Then COMPOSITE(n) iff $\exists m_1, m_2.$ VER-COMPOSITE(n, m_1, m_2).
Remark: In fact it can be shown that COMPOSITE $\in$ P.

3. Given G_1, G_2 we simply guess the potential isomorphism, which is a pair of mappings (f, g) for the nodes and the arcs. We then check in p-time that (f, g) is an isomorphism between the two graphs: f, g are bijections on nodes, resp. arcs, and if a has endpoints x, y in G_1 then endpoints of $g(a)$ are $f(x), f(y)$ in g_2. Note that f and g can be p-bounded in the sizes of the input graphs, since $f \subseteq \text{nodes}(G_1) \times \text{nodes}(G_2)$ and $g \subseteq \text{arcs}(G_1) \times \text{arcs}(G_2)$. Hence GRAPHISOM $\in$ NP.
VER-GRAPHISOM(G_1, G_2, f, g) iff (f, g) is an isomorphism between G_1 and G_2.
Remark: It is unknown whether GRAPHISOM $\in$ P.

4. (a) Let D be a decision problem. Then $D \leq D$ with the identity function as reduction function: We have $D(x)$ iff $D(f(x))$ with $f(x) = x$. Clearly the identity function is p-time computable.
(b) Let D, D', D'' be problems with $D \leq D' \leq D''$. We have p-time computable f, g such that

$$D(x) \text{ iff } D'(f(x))$$
$$D'(y) \text{ iff } D''(g(y))$$

Clearly $D(x)$ iff $D''(g(f(x))$ (substituting $f(x)$ for y). The function $g \circ f$ is p-time computable by Proposition 3.1.10 (lecture notes page 75). Hence $D \leq D''$ as required.

5. Suppose D is NP-hard and $D \leq D'$. Let D'' be any problem in NP. Since D is NP-hard, we have $D'' \leq D$. But $D \leq D'$. Hence $D'' \leq D'$ using transitivity of $\leq$ (Question 4). Hence for any NP problem D'' we have $D'' \leq D'$, showing that D' is NP-hard.

6. Since both $\mathrm{TSP(D)}$ and $\mathrm{VRPC(D)}$ are NP-complete, they are both in NP and both NP-hard. So for all $D \in$ NP we have $D \leq \mathrm{TSP(D)}$ and $D \leq \mathrm{VRPC(D)}$. So $\mathrm{VRPC(D)} \leq \mathrm{TSP(D)}$, $\mathrm{TSP(D)} \leq \mathrm{VRPC(D)}$. Hence $\mathrm{TSP(D)} \sim \mathrm{VRPC(D)}$.
Note that this argument works in general for any two NP-complete problems.

7. (a) A naive method

```
**Algorithm** Exponentiation:
procedure slowexp($m,n$):
  if $n = 0$:
    return 1
  else:
    return $m*$slowexp($m, n-1$)
```

would have n multiplications, which is exponential in the input size $\log m + \log n$.
Here is a program using repeated squaring:

```
**Algorithm** Exponentiation:
procedure exp($m,n$):
  if $n = 0$:
    return 1
  else:
    if $n$ is even:
      return exp($m * m, n/2$)
    else:
      # $n$ is odd
      return $m*$exp($m * m, (n-1)/2$)
```

For a given exponent n, the number of calls to the procedure is the number of times we can divide n by two (with an adjustment if n is odd). Thus the algorithm performs $O(\log n)$ multiplications and divisions (in fact only division by two is needed). Thus we use only polynomially many (in the input size $\log m + \log n$) multiplications and divisions.

(b) Exponentiation cannot be p-time, since the size of the output is $\log(m^n) = n \log m$, which is exponential in the input size. No matter how good an algorithm we have, we do not have time to write down the answer in p-time. If we look at the multiplications carried out when performing repeated squaring, we see that the input size keeps on doubling.
Quite often exponentiation is carried out in modular arithmetic (dividing by the modulus and keeping the remainder). In that case we do get a p-time operation.

8. Let us define the relation of being a valid timetable: $V(E, \{L_e : e \in E\}, T, f)$ iff $f : E \to T$ is a timetable for $(E, \{L_e : e \in E\}, T)$ with no clashes. Then clearly $(E, \{L_e : e \in E\}, T)$ has a timetable iff $\exists f . V(E, \{L_e : e \in E\}, T, f)$. We therefore need to show that $V \in \mathsf{P}$. But this means checking: for all pairs of exams e_1, e_2, if $e_1 \neq e_2$ and $f(e_1) = f(e_2)$ then $L_{e_1} \cap L_{e_2} = \emptyset$. Let the input size be $n = |E| + |T| + \sum_{e \in E} |L_e|$. Checking whether $L_{e_1} \cap L_{e_2} = \emptyset$ is $O(n^2)$. This has to be checked for each pair of exams, which gives us two outer for-loops. We conclude that V is $O(n^4)$, and so $V \in \mathsf{P}$. Finally we need to observe that f is p-bounded in n. But $|f| \leq |E| \times |T| \leq n^2$. Hence we have shown that the exam timetable problem is in NP. It is in fact NP-complete, which can be shown by a reduction from the problem of colouring a graph.
Note that (at the expense of extra space usage) we could reduce the polynomial from $O(n^4)$ to $O(n^2)$ by first computing in time $O(n^2)$ a two-dimensional Boolean array telling us whether two exam share a candidate. This can then be used inside the two outer for-loops. This might be a good thing to do if we are interested in efficiently checking a proposed timetable. However it makes no difference if we are simply interested in whether checking is p-time or not.
Other answers are possible for the size of the input or the order of the polynomial, depending on what computational model is used. However, by the polynomial invariance thesis, this makes no difference to whether checking the timetable is p-time.