Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 1: Turing machines

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

1. Show that each of the following "generalisations" of the Turing machine model can be simulated efficiently by a standard Turing machine.

   (a) A Turing machine defined in the same way as normal, but with a *two-way* infinite tape. The head starts at position 0 and can move arbitrarily far in either direction (by at most one position at each step).

   (b) A Turing machine with a 2-dimensional, one-way infinite "tape". The head starts at position $(0, 0)$ and can move arbitrarily far down or right. At each step, the head can move at most one position up, down, left or right.

2. For the purposes of this question, say that a variant of the Turing machine is *equivalent* to the standard Turing machine if the set of languages $\mathcal{L} \subseteq \{0,1\}^*$ that can be decided by the variant model is the same as the set of languages that can be decided by the standard model. Is each of the following variants of the Turing machine equivalent to the standard Turing machine?

   (a) A Turing machine which operates on a two-way infinite tape, and can move arbitrarily far in each direction. That is, the transition function is of the form $\delta : K \times \Sigma \to K \times \Sigma \times \mathbb{Z}$, where the last integer specifies how far to move to the right (if positive), or left (if negative).

   (b) A Turing machine operating on the infinite alphabet $\Sigma = \mathbb{N}$.

   (c) A Turing machine which cannot stay still. That is, the transition function is of the form $\delta : K \times \Sigma \to K \times \Sigma \times \{\leftarrow, \rightarrow\}$.

   (d) A Turing machine which cannot move left. That is, the transition function is of the form $\delta : K \times \Sigma \to K \times \Sigma \times \{-, \rightarrow\}$.

3. Let $\mathcal{L}$ be a language such that $|\mathcal{L}|$ is finite. Show that $\mathcal{L}$ is decidable.

4. Let $\mathcal{L}_1$, $\mathcal{L}_2$ be decidable (resp. recognisable) languages. Are the languages $\mathcal{L}_1 \cup \mathcal{L}_2$ and $\mathcal{L}_1 \cap \mathcal{L}_2$ necessarily decidable (resp. recognisable)?

5. Is HALT recognisable? Let $\overline{\text{HALT}}$ be the language defined by

   $$\overline{\text{HALT}} = \{(\underline{M}, x) : M \text{ is a Turing machine that does not halt on input } x\ \}.$$

   Is $\overline{\text{HALT}}$ recognisable?

6. For any Turing machine M, let $L(M)$ denote the language recognised by M. Let S be a set of languages, and let $\mathcal{L}_S$ be the language

$$\mathcal{L}_S = \{\underline{M} : L(M) \in S\}.$$

That is, $\mathcal{L}_S$ is the set of Turing machines that recognise languages in S. Show that $\mathcal{L}_S$ is either trivial or undecidable. Conclude that the language

$$\mathcal{L}_k = \{\underline{M} : |L(M)| = k\}$$

is undecidable for any k. In other words, for any k it is undecidable whether, given a Turing machine M, M accepts exactly k inputs.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 2: Time complexity

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

1. To get some practice with big-O notation, prove or disprove the following assertions.

   (a) If $f(n) = n^k$ and $g(n) = c^n$, for some constants $c > 1$, $k \geq 0$, then $f(n) = o(g(n))$.

   (b) If $f(n) = \Theta(g(n))$, then $2^{f(n)} = \Theta(2^{g(n)})$.

   (c) $f(n) = 2^{\Theta(\log n)}$ if and only if $f(n) = \text{poly}(n)$.

   (d) If $f(n) = O(n)$ and $g(n) = \Theta(n)$, then:

   i. $f(n) + g(n) = O(n)$;

   ii. $f(n)g(n) = O(n^2)$;

   iii. $f(n) - g(n) = O(1)$;

   iv. $\frac{f(n)}{g(n)} = O(1)$.

2. Let $T(n)$ be defined recursively by $T(1) = 1$, and $T(n) = cT(\lceil n/2 \rceil) + d$ for some constants $c, d > 1$. Show that $T(n) = \Theta(n^{\log_2 c})$.

3. Give examples of functions f, g such that neither $f(n) = O(g(n))$, nor $f(n) = \Omega(g(n))$.

4. Prove that $T(n) = 2^n$ is time-constructible.

5. Give an example (not necessarily explicit) of a function $T : \mathbb{N} \to \mathbb{N}$ such that $T(n) \geq n$ and $T(n)$ is not time-constructible.

6. Prove the claim in the notes that polynomial-time reductions compose: if $\mathcal{A} \leq_{\mathsf{P}} \mathcal{B}$, and $\mathcal{B} \leq_{\mathsf{P}} \mathcal{C}$, then $\mathcal{A} \leq_{\mathsf{P}} \mathcal{C}$.

7. Given two strings $a, b \in \Sigma^*$, a common subsequence of a and b is a sequence of symbols which is a subsequence of both a and b. For example, *abara* is a common subsequence of *abracadabra* and *barbarian*. Give an algorithm which outputs the maximal length of a common subsequence of two strings in polynomial time.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 3: NP

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

1. Given languages $\mathcal{L}_1, \mathcal{L}_2 \in \mathsf{NP}$, prove that $\mathcal{L}_1 \cap \mathcal{L}_2 \in \mathsf{NP}$, $\mathcal{L}_1 \cup \mathcal{L}_2 \in \mathsf{NP}$.

2. Given languages $\mathcal{L}_1 \subseteq \mathcal{L}_2$ such that $\mathcal{L}_1 \in \mathsf{NP}$, $\mathcal{L}_2 \in \mathsf{NP}$, is $\mathcal{L}_2 \backslash \mathcal{L}_1 \in \mathsf{NP}$?

3. Show that, if $\mathsf{P} = \mathsf{NP}$, then any non-trivial language $\mathcal{L} \in \mathsf{P}$ is NP-complete.

4. Let Unary Primes be the following language.

$$\text{Unary Primes} = \{1^n : n \text{ is prime}\}.$$

   Prove that, if Unary Primes is NP-complete, then $\mathsf{P} = \mathsf{NP}$.

5. Prove that $\emptyset$ is not NP-complete.

6. Let DNF-SAT be the variant of SAT where the input is required to be a boolean formula in DNF, rather than CNF. Prove that DNF-SAT $\in \mathsf{P}$, and consider the following claim: As CNF formulae can be converted into DNF formulae using De Morgan's laws, $\mathsf{P} = \mathsf{NP}$. Is this claim correct?

7. Prove that Halt is NP-hard. Is it NP-complete?

8. Prove that, for any $\epsilon > 0$, there exists an NP-complete language which can be decided in time $O(2^{n^\epsilon})$. ("There exist almost arbitrarily easy NP-complete problems.")

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 4: NP-completeness

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

1. Prove that each of the following problems is NP-complete.

   (a) 0-1 Integer Programming. Given a list of linear inequalities over n variables with integer coefficients, is there an assignment of 0's and 1's to the variables that satisfies all the inequalities? For example, the inequalities

   $$\begin{aligned} 3x_1 - x_2 + x_3 &\geq 1 \\ -x_1 + 2x_3 &\leq 2 \end{aligned}$$

   are satisfied by setting $x_1 = 1$, $x_2 = 0$, $x_3 = 1$.

   (b) Hamiltonian Cycle. Given a directed graph G, is there a cycle which includes each vertex of G exactly once?

   (c) 3-colouring. Given an undirected graph G, does there exist an assignment of at most 3 colours to the vertices of G, such that adjacent vertices are assigned different colours?

2. Consider the following variants of the SAT problem. In each case, either show that the problem is in P or that it is NP-complete, as appropriate.

   (a) The special case of 3-SAT where each variable appears at most three times in the input formula.

   (b) The special case of 3-SAT where each variable appears at most twice in the formula.

   (c) The variant of SAT where a clause is *unsatisfied* if and only if the literals in the clause either all evaluate to true, or all evaluate to false. For example, the clause $(x_1, \neg x_2)$ is satisfied by the assignment $x_1 = 1$, $x_2 = 1$ and the assignment $x_1 = 0$, $x_2 = 0$.

   (d) The variant of 3-SAT where each clause is satisfied if and only if an odd number of literals in the clause evaluate to true. For example, the clause $(x_1, \neg x_2, x_3)$ is satisfied by $x \in \{000, 011, 110, 101\}$.

3. Is each of the following special cases of NP-complete problems in P or NP-complete?

   (a) Undirected Hamiltonian Path. Given an undirected graph G, does there exist a path which visits each vertex exactly once?

   (b) 2-colouring. Given an undirected graph G, does there exist an assignment of at most 2 colours to the vertices of G, such that adjacent vertices are assigned different colours?

   (c) Clique of size 100. Given an undirected graph G, does it contain a clique of size at most 100?

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 5: P vs. NP and beyond

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Starred questions are likely to be more challenging/interesting.

1. Prove that $\mathsf{P} \subseteq \mathsf{NP} \cap \mathsf{co\text{-}NP}$, and that if $\mathsf{P} = \mathsf{NP}$, $\mathsf{NP} = \mathsf{co\text{-}NP}$.

2. Prove that $\mathsf{co\text{-}NP} \subseteq \mathsf{P}^{\mathrm{SAT}}$.

3. Prove that, if $\mathsf{P} = \mathsf{NP}$, $\mathsf{EXP} = \mathsf{NEXP}$.

4. Prove that, if $\mathsf{P} = \mathsf{NP}$, then $\mathsf{PH} = \mathsf{P}$ – "the polynomial hierarchy collapses".

5. Let Smallest Formula be the following problem. Given a boolean formula ϕ in CNF, and an integer k, does there exist a CNF formula ϕ' such that ϕ' contains at most k symbols, and ϕ' computes the same function as ϕ? Prove that Smallest Formula $\in \Sigma_2$.

6. ($\star$) Give an *explicit* algorithm for SAT which, given a boolean formula ϕ, outputs a satisfying assignment for ϕ in polynomial time, assuming that $\mathsf{P} = \mathsf{NP}$. If there is no satisfying assignment, the algorithm can behave arbitrarily.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 6: Space complexity

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Starred questions are likely to be more challenging/interesting.

1. Show that, if $\mathcal{A} \in \mathsf{NL}$ and $\mathcal{B}$ is any non-trivial language, $\mathcal{A} \leq_{\mathsf{P}} \mathcal{B}$. That is, almost every language $\mathcal{B}$ is NL-hard under polynomial-time reductions.

2. Prove composability of log-space reductions: i.e. that, if $\mathcal{A} \leq_{\mathsf{L}} \mathcal{B}$ and $\mathcal{B} \leq_{\mathsf{L}} \mathcal{C}$, $\mathcal{A} \leq_{\mathsf{L}} \mathcal{C}$.

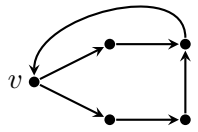
3. Let $\mathcal{L}$ be the language of properly nested parentheses. For example, (()) and (()(()())) are in $\mathcal{L}$ but)(is not. Show that $\mathcal{L} \in \mathsf{L}$.

4. Prove the claim in the lecture notes that, if the read-once restriction is removed from the definition of NL, the resulting class is equal to NP.

5. Assuming the Immerman-Szelepcsényi Theorem, prove that 2-SAT $\in \mathsf{NL}$.

6. Prove that $\mathsf{P} \neq \mathsf{SPACE}(n)$. [Hint: you need not show that either class contains the other.]

7. ($\star$) For any directed graph G, consider the following two-player game. Starting at a given node v and with player 1, players take it in turns to pick an arc from the current node to a node which has not yet been visited. If there is no arc from the current node to an unvisited node, the current player loses. For example, on the following graph, starting with node v, player 1 can always win.



Show that the following problem is PSPACE-complete: given a graph G and node v, determine whether player 1 can win when starting from node v.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 7: Randomised algorithms

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Starred questions are likely to be more challenging/interesting.

1. Prove that $\mathsf{PP} \subseteq \mathsf{PSPACE}$.

2. Imagine we generalise the definition of BPP to a class $\mathsf{BPP}_{\epsilon,\delta}$ defined as follows. Letting M be a PTM as in the standard definition, we say that $\mathcal{L} \in \mathsf{BPP}_{\epsilon,\delta}$ if there exists an M such that:

   - For all $x \in \mathcal{L}$, $\Pr[M(x) = 0] \leq \epsilon$;
   - For all $x \notin \mathcal{L}$, $\Pr[M(x) = 1] \leq \delta$.

   Thus $\mathsf{BPP} = \mathsf{BPP}_{\frac{1}{3},\frac{1}{3}}$.

   (a) What are the classes $\mathsf{BPP}_{0,0}$ and $\mathsf{BPP}_{\frac{1}{2},\frac{1}{2}}$?

   (b) Show that $\mathsf{BPP}_{\frac{1}{2},0} \subseteq \mathsf{NP}$.

   (c) Show that $\mathsf{BPP}_{2^{-|x|},2^{-|x|}} = \mathsf{BPP}$.

   (d) What is the class $\mathsf{BPP}_{2-2^{|x|},2-2^{|x|}}$?

3. Imagine we generalise the definition of BPP by allowing the PTM to run in *expected* time polynomial in the input size. That is, the expected running time of the PTM is polynomial on every input (but sometimes, depending on its internal randomness, it might run for much longer). Does this change the definition of BPP?

4. ($\star$) Modify the randomised algorithm for k-SAT to give a randomised algorithm which determines with worst-case success probability at least 0.99 whether a graph G with n vertices can be properly 3-coloured. Prove that your algorithm runs in time $O((3/2)^n \operatorname{poly}(n))$, beating the naïve algorithm of exhaustively trying every possible colouring, which takes time $\Omega(3^n)$.

5. ($\star$) Give alternative non-trivial randomised and deterministic algorithms for 3-colouring. Possible examples include a different randomised algorithm for 3-colouring which uses time $O((3/2)^n \operatorname{poly}(n))$, and a deterministic algorithm which uses time $O(1.94^n \operatorname{poly}(n))$.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 8: Counting complexity

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Starred questions are likely to be more challenging/interesting.

1. Give a parsimonious reduction from SAT to 3-SAT.

2. Let #DNF denote the problem of counting the number of satisfying assignments to a boolean formula in disjunctive normal form. Prove that #DNF is #P-complete.

3. ($\star$) Prove that counting the number of matchings (not necessarily perfect) in a bipartite graph is #P-complete.

4. Assuming (3), prove that #2-SAT is #P-complete.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 9: Circuit complexity

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Starred questions are likely to be more challenging/interesting.

1. Prove that $\mathsf{AC}^d \subseteq \mathsf{NC}^{d+1}$ for any $d \geq 0$.

2. Prove that $\mathsf{NC}^1 \subseteq \mathsf{L}$.

3. Given two $n \times n$ boolean matrices A and B (i.e. matrices which contain only 0's and 1's), define the boolean matrix product

$$(A \circ B)_{ij} = \bigvee_{k=1}^{n} (A_{ik} \wedge B_{kj}).$$

Also define the problem "BMM" as follows: given two $n \times n$ boolean matrices A and B, and integers s and t such that $1 \leq s, t \leq n$, determine whether $(A \circ B)_{st} = 1$.

   (a) Show that BMM $\in \mathsf{AC}^0$.

   (b) Show that, given an $n \times n$ boolean matrix A, and any integer $k = O(\log n)$, the problem of determining whether $(A^{\circ 2^k})_{st} = 1$ is in AC^1. Here $A^{\circ 2^k}$ is the 2^k-fold product $A \circ A \circ \cdots \circ A$.

   (c) Conclude that $\mathsf{NL} \subseteq \mathsf{AC}^1$.

4. ($\star$) Let 1UN-SAT be the special case of SAT where every clause contains at most one unnegated variable (for example, $x_1 \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2)$ is a valid 1UN-SAT instance). Prove that 1UN-SAT is P-complete.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 10: Decision trees

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Starred questions are likely to be more challenging/interesting.

1. Assume that $f : \{0,1\}^n \to \{0,1\}$ depends only on the Hamming weight of the input: $f(x) = g(\mathrm{wt}(x))$ for some $g : \{0,\ldots,n\} \to \{0,1\}$. Show that either $D(f) = 0$ or $D(f) = n$.

2. Given a function $f : \{0,1\}^n \to \mathbb{R}$, describe how to write down a representation of f as a multilinear polynomial. Show that this polynomial is unique.

3. Consider the function $f : \{0,1\}^n \to \{0,1\}$ defined by

$$f(x) = \begin{cases} 1 & \text{if } x \text{ contains two consecutive 1's} \\ 0 & \text{otherwise.} \end{cases}$$

   Show that $D(f) = n - 1$ if $n \equiv 1 \bmod 3$, and $D(f) = n$ otherwise.

4. Consider the function $\mathrm{MAJ} : \{0,1\}^3 \to \{0,1\}$ defined by

$$\mathrm{MAJ}(x) = \begin{cases} 1 & \text{if } \mathrm{wt}(x) \geq 2 \\ 0 & \text{if } \mathrm{wt}(x) \leq 1. \end{cases}$$

   For arbitrary integer $k > 1$, we define the function $\mathrm{MAJ}^{(k)} : \{0,1\}^{3^k} \to \{0,1\}$ recursively by $\mathrm{MAJ}^{(k)}(x) = \mathrm{MAJ}(\mathrm{MAJ}^{(k-1)}(x^1), \mathrm{MAJ}^{(k-1)}(x^2), \mathrm{MAJ}^{(k-1)}(x^3))$, where the input x is partitioned into three consecutive blocks x^1, x^2, x^3 of 3^{k-1} bits each, and $\mathrm{MAJ}^{(1)}(x) = \mathrm{MAJ}(x)$.

   Prove that $R(\mathrm{MAJ}^{(k)}) = O((8/3)^k)$. Optional: Improve this bound.

5. ($\star$) Prove Theorem 13.10 by giving an $O(n)$-query algorithm to determine whether an input graph is a scorpion graph.

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 11: Communication complexity and interactive proofs

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

Starred questions are likely to be more challenging/interesting.

1. Let $\mathrm{GT} : \{1, \ldots, 2^n\} \times \{1, \ldots, 2^n\} \to \{0, 1\}$ be the greater than function: $\mathrm{GT}(x, y) = 1$ if and only if $x < y$. Show that $D_{cc}(\mathrm{GT}) = n$ but $R_{cc}(\mathrm{GT}) = O(\log^2 n)$.

2. Let $\mathrm{DISJ} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be the set disjointness function, $\mathrm{DISJ}(x, y) = 0$ if and only if there exists i such that $x_i = y_i = 1$. Prove that $D_{cc}(\mathrm{DISJ}) = n$.

3. Prove that any multiple-tape Turing machine which decides the language of palindromes must use space $\Omega(\log n)$.

4. ($\star$) Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be picked uniformly at random. Show that for $n \geq 4$, with probability > 0.999, $D_{cc}(f) = n$.

5. Prove that any language with a multiple-prover interactive proof can be decided by an NDTM running in exponential time ($\mathsf{MIP} \subseteq \mathsf{NEXP}$).

6. ($\star$) Let p be prime. An integer y is said to be a quadratic residue modulo p if there exists x such that $y \equiv x^2 \pmod{p}$. Prove (directly) that the following language is in IP.

$$\mathrm{QNR} = \{(y, p) : y \text{ is not a quadratic residue modulo } p\}.$$

Part III **Michaelmas 2012**

COMPUTATIONAL COMPLEXITY

EXERCISE SHEET 12: Hardness of approximation and PCP

Ashley Montanaro, DAMTP Cambridge
am994@cam.ac.uk

1. Let MAX-2SAT be the following problem. Given a CNF formula ϕ with at most 2 variables per clause, output the largest number of clauses in ϕ which can be satisfied by any assignment x.

   (a) (Optional.) Show that MAX-2SAT is NP-hard.

   (b) Give a polynomial-time algorithm which approximates MAX-2SAT to within a factor of 3/4.

2. The following strengthening of the PCP Theorem is known to hold: for any $\epsilon > 0$, $\mathsf{NP} \subseteq \mathsf{PCP}_{1,1/2+\epsilon}(O(\log n), 3)$. That is, any language in NP has a probabilistically checkable proof with perfect completeness where the verifier reads only 3 bits of the proof! Show that the 3 cannot be replaced with a 2 unless $\mathsf{P} = \mathsf{NP}$.

3. Consider the following "δ-biased" linearity test in the ± 1 picture. Given access to a function $f : \{0,1\}^n \to \{\pm 1\}$, pick $x, y \in \{0,1\}^n$ uniformly at random. Also pick $z \in \{0,1\}^n$ at random, but with the biased probability distribution $\Pr[z_i = 1] = \delta$, $\Pr[z_i = 0] = 1 - \delta$, for some $0 \leq \delta \leq 1$. Compute $f(x)$, $f(y)$ and $f(x \oplus y \oplus z)$, and accept if $f(x)f(y)f(x \oplus y \oplus z) = 1$; otherwise reject. For $\delta = 0$, this is just the standard linearity test.

   (a) Show that the probability that the δ-biased linearity test accepts is

   $$\frac{1}{2} + \frac{1}{2} \sum_{S \subseteq [n]} (1 - 2\delta)^{|S|} \hat{f}(S)^3.$$

   (b) Let $f : \{0,1\}^n \to \{\pm 1\}$ be a function that depends on at most one of its n input variables. Show that if f is constant, then $\hat{f}(S) = \pm 1$ for $S = \emptyset$ and $\hat{f}(S) = 0$ for all other $S \subseteq [n]$; and if f depends on exactly one of its input variables, then $\hat{f}(S) = \pm 1$ for some $S \subseteq [n]$ such that $|S| = 1$, and $\hat{f}(S) = 0$ for all other $S \subseteq [n]$.

   (c) Show that the δ-biased linearity test can be used to give a test which, for any ϵ, uses $O(1/\epsilon^2)$ queries to f and distinguishes, with constant success probability, between two cases: f depends on at most one of its input variables, or f is ϵ-far from depending on at most one of its input variables.