CSC2047 Theory of Computation
Assignment 2 (v3)
Due: 29 March 2020 (Sunday), 23:55
Questions
Total Marks: 100

Rules:

- **Submission and late submission:**
  The solutions must be submitted in PDF only via Canvas. You may wish to complete your coursework electronically using Word (utilising Microsoft Equation Editor where appropriate), LaTeX, or other technology which facilitates mathematical typesetting. If you complete your coursework on paper you must create a digital copy of your work using a scanner. Photographs of paper based coursework will not be accepted. Late submissions will be deducted 5% marks per day of delay. No submission will be accepted more than 7 days later than the deadline.

- **No plagiarism allowed:**
  This is an individual assignment. No plagiarism is permitted: you should not copy your solutions from each other or any resource. If you need to refer to online sources/books (e.g. for some new definition), you must refer to them appropriately. If plagiarism is detected you may lose marks and/or face other action.

- **Collusion is not permitted:**
  The submitted work should be solely of your own completion in accordance of Section 2.5 of the Academic Offences guidelines. You are not permitted to work with other students or third parties e.g. using online forums or pay-for-solution websites.

- A guide to academic offences for students can be viewed:

  https://www.qub.ac.uk/directorates/AcademicStudentAffairs/AcademicAffairs/AppealsComplaintsandMisconduct/AcademicOffences/Student-Guide/

- **This is an open book and open resource assignment:**
  You are allowed to access books and online resources. However, you must attribute sources (see point 6) and the solutions must be in your own words.

- **Attribution:**
  If at all you need to cite any sources/books (standard definitions do not require a citation), have a separate references section at the end. All the references should be present using a single standardised reference style (e.g. IEEE, APA, Harvard etc.).

- **Show working out:**
  It is recommended that you show your working out throughout this assignment. This can be beneficial in case you make some mistake in which case partial marks may be awarded for showing the correct process.

1. Decide whether each of the following expressions are true or not. Answer yes or no.

Hint: Remember that e.g. $4n = \mathcal{O}(n^2)$ is true, even though $4n = \mathcal{O}(n)$ is also the case.

In any case where it is not true, perform an asymptotic analysis using the informal method discussed in the lecture so as to provide a correct $\mathcal{O}$-complexity (that is, do not provide an $\mathcal{O}$-complexity which is unnecessarily high; e.g. for $4n$, $\mathcal{O}(n)$ would be fine, however $\mathcal{O}(n^2)$ would not).

(a) $n! + 3n^6 + 2n^3 = \mathcal{O}(n^6)$ [1 mark]

(b) $2\sqrt{n} = \mathcal{O}(n!)$ [1 mark]

(c) $\log_3 n = \mathcal{O}(\log_2 n)$ [1 mark]

(d) $\log_2 n = \mathcal{O}(\log_3 n)$ [1 mark]

(e) $3 \cdot 2^n = \mathcal{O}(1.5^n)$ [1 mark]

(f) $3 \cdot 2^n = 2^{\mathcal{O}(n)}$ [1 mark]

(g) $3^n = 2^{\mathcal{O}(n)}$ [1 mark]

(h) $3n = \mathcal{O}((\sqrt{n})^2)$ [1 mark]

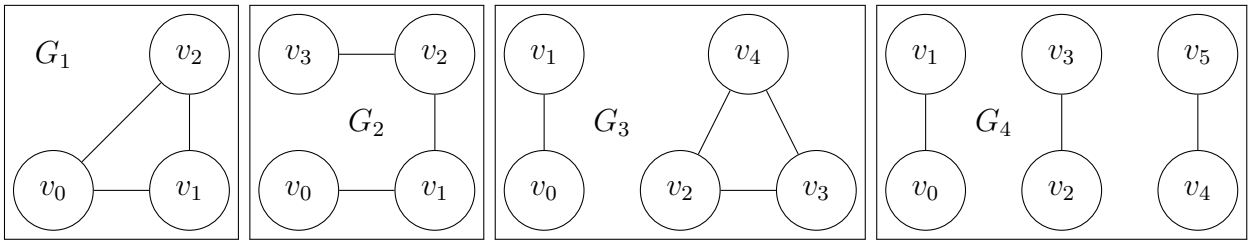
(i) $3! + 7 = \mathcal{O}(1)$ [1 mark]

(j) $2n + \sqrt{n} = \mathcal{O}(n \log n)$ [1 mark]

2. Consider the language

$$L = \{\langle G, n\rangle \mid G \text{ is an undirected graph with } n \text{ connected components}\}.$$

Consider the following undirected graphs:



(a) For each of the following statements, decide whether it holds.

(i) $\langle G_1, 1\rangle \in L$ [1 mark]

(ii) $\langle G_2, 2\rangle \in L$ [1 mark]

(iii) $\langle G_3, 1\rangle \in L$ [1 mark]

(iv) $\langle G_3, 4\rangle \in L$ [1 mark]

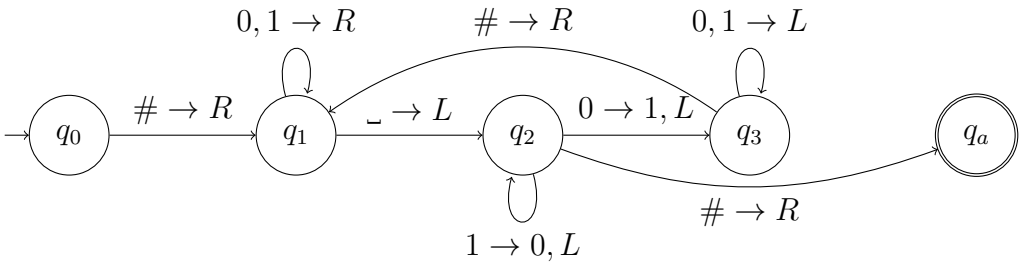
(v) $\langle G_4, 3\rangle \in L$ [1 mark]

(vi) $\langle G_4, 4\rangle \in L$ [1 mark]

(b) Prove that L is decidable by providing a high-level decider. (That is, an algorithm-like description in English, cf. the according lecture slides) Your implementation should require no more than polynomial time. [2 marks]

(c) Argue that your decider runs in polynomial time. Do so by reasoning about the runtime of its individual steps, the number of steps required, etc. as in the lecture. [2 marks]

3. Consider the following Turing machine M with input alphabet $\Sigma = \{0, 1, \#\}$:



(a) Let C_1 be the initial configuration for the input word $\#101$ and let C_2 be the configuration yielded by C_1. That is, C_2 is the configuration obtained from the Turing machine after one step when it is started on the word $\#101$.

(i) Provide C_1 in terms of a string [1 mark]

(ii) Provide C_2 in terms of a string [1 mark]

(b) Decide whether each of the following strings would be accepted by the Turing machine. Write down the computation the Turing machine performs for each of them in terms of a sequence of configurations. For the accepted ones, answer yes; otherwise, no.

(i) $\#$ [1 mark]

(ii) $\#\#$ [1 mark]

(iii) 0 [1 mark]

(iv) $\#0$ [1 mark]

(v) $\#1$ [1 mark]

(vi) $\#00$ [1 mark]

(vii) $\#01$ [1 mark]

(viii) $\#10$ [1 mark]

(ix) $\#11$ [1 mark]

(x) $\#101$ [1 mark]

(c) What is the language of accepted (input) words of this Turing machine? Describe the language using a regular expression. [1 mark]

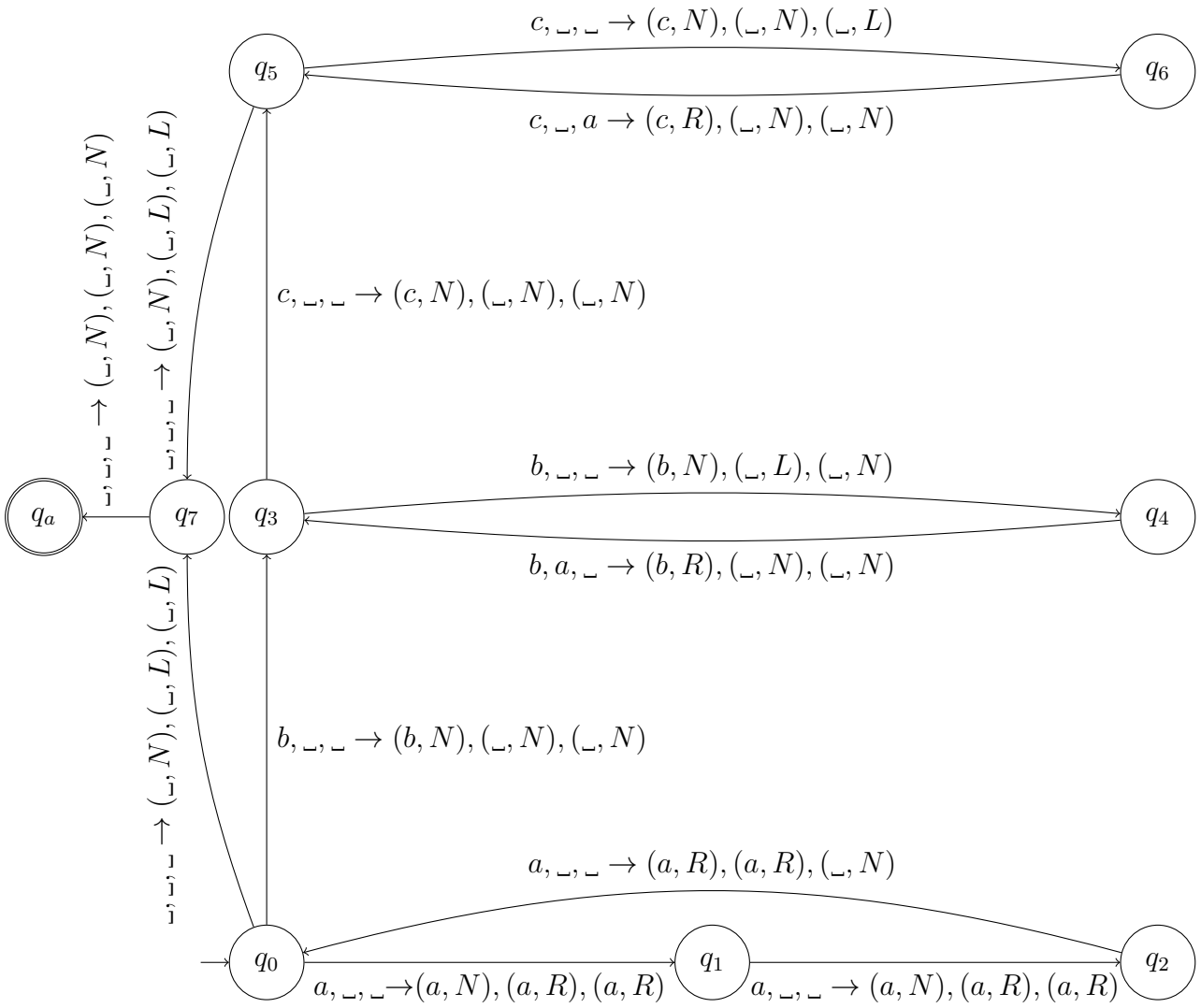
(d) What is the language of (output) words which may be on the tape at the moment the Turing machine has moved to the accepting state? Describe the language using a regular expression. [1 mark]

(e) What is the worst-case runtime $f(n)$ of this Turing machine for a word of length n? [2 marks]

(f) What is the best-case runtime $g(n)$ of this Turing machine for a word of length n? [2 marks]

(g) Formalise the graphical representation of the Turing machine above as a 7-tuple. [2 marks]

4. Consider the following three-tape Turing machine with input alphabet $\Sigma = \{a, b, c\}$:



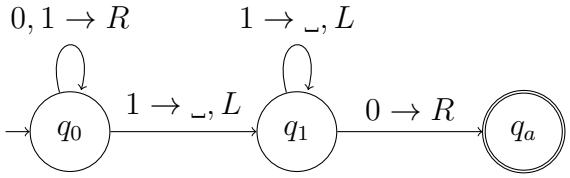
(a) Decide whether each of the following strings would be accepted by the Turing machine. Write down the computation the Turing machine performs for each of them in terms of a sequence of configurations. For the accepted ones, answer yes; otherwise, no.

(i) ε [1 mark] **(ii)** abc [1 mark] **(iii)** $abbbcc$ [1 mark] **(iv)** $abbb$ [1 mark] **(v)** bac [1 mark] **(vi)** $aabbbbbbcccc$ [1 mark]

(b) Which language does this machine recognise? Provide the language in set notation. [2 marks]

(c) Provide a two-tape Turing machine which recognises the same language and still runs in the same runtime order $\mathcal{O}(n)$. [2 marks]

5. Consider the following nondeterministic Turing machine with input alphabet $\Sigma = \{0, 1\}$:



(a) Decide whether each of the following strings would be accepted by the Turing machine. Write down a computation the Turing machine performs for each of them in terms of a sequence of configurations. For the accepted ones, the computation you write down should be an accepting computation. For the accepted ones, answer yes; otherwise, no.

(i) ε [1 mark] **(ii)** 0 [1 mark] **(iii)** 1 [1 mark] **(iv)** 01 [1 mark] **(v)** 10 [1 mark] **(vi)** 101 [1 mark] **(vii)** 11100 [1 mark] **(viii)** 10100 [1 mark]

(b) What is the language of accepted words of this Turing machine? Describe the language using a regular expression. [2 marks]

(c) What is the worst-case runtime $f(n)$ of this Turing machine for a word of length n? Remember that you have to consider the maximum over all possible runs for the word. [2 marks]

(d) What is the best-case runtime $g(n)$ of this Turing machine for a word of length n? Remember that you have to consider the maximum (not the minimum) over all possible runs for the word. [2 marks]

(e) What is the maximum number of different runs for a given word of length n? [1 mark]

(f) What is the minimum number of different runs for a given word of length $n > 0$? [1 mark]

(g) Provide a (deterministic) Turing machine which always halts which recognises the same language as this nondeterministic Turing machine. [2 marks]

(h) Formalise the graphical representation of the nondeterministic Turing machine above as a 7-tuple. [2 marks]

6. Consider the language

$$HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph with a Hamiltonian path from } s \text{ to } t\}$$

$HAMPATH$ can be reduced to SAT as follows: Consider an arbitrary directed graph $G = (V, E)$ with n vertices and m edges. We assume $V = \{v_1, \ldots, v_n\}$. We consider the set of variables

$$C = \{c_{i,j} \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n\}.$$

Consider formula ϕ consisting of the conjunction ($\wedge$) of the following set S of clauses:

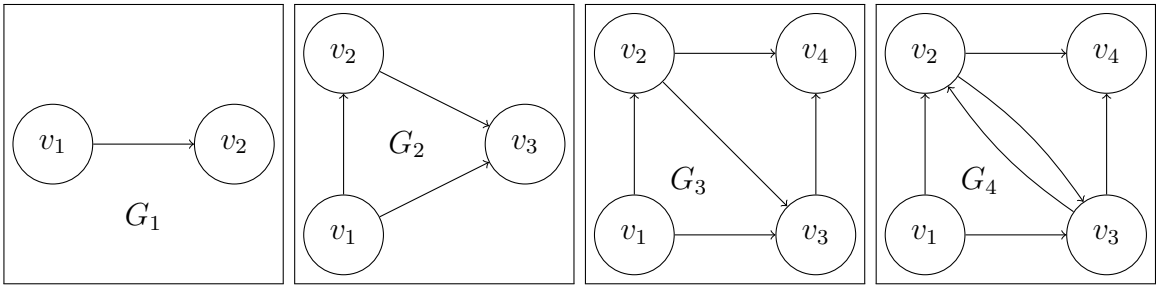
$$S = S_{\text{atleastonce}} \cup S_{\text{maxoneperstage}} \cup S_{\text{nottwice}} \cup S_{\text{first}} \cup S_{\text{last}} \cup S_{\text{path}}$$

where

- $S_{\text{atleastonce}} = \{(c_{i,1} \vee \ldots \vee c_{i,n}) \mid 1 \leq i \leq n\}$,
- $S_{\text{maxoneperstage}} = \{\overline{c_{i,k}} \vee \overline{c_{j,k}} \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq n \text{ and } i < j \text{ and } 1 \leq k \leq n\}$
- $S_{\text{nottwice}} = \{(\overline{c_{k,i}} \vee \overline{c_{k,j}}) \mid 1 \leq k \leq n \text{ and } 1 \leq i \leq n \text{ and } 1 \leq j \leq n \text{ and } i < j\}$,
- $S_{\text{first}} = \{c_{i,1}\}$ where $v_i = s$,
- $S_{\text{last}} = \{c_{i,n}\}$ where $v_i = t$,
- $S_{\text{path}} = \{\overline{c_{i,k}} \vee \overline{c_{j,k+1}} \mid (v_i, v_j) \notin E \text{ and } 1 \leq i \leq n \text{ and } 1 \leq j \leq n \text{ and } 1 \leq k \leq n-1 \text{ and } i \neq j\}$.

For instance, if we consider the set $A = \{a \vee b, \overline{c}, \overline{c} \vee a\}$, then the conjunction of the clauses of A would be $\phi = (a \vee b) \wedge \overline{c} \wedge (\overline{c} \vee a)$, where the order of clauses in ϕ is arbitrary. The way the encoding works is similar to how the nondeterministic decider N_1 from "Time Complexity - Part 3" recognises $HAMPATH$. A variable $c_{i,k}$ encodes the fact that that node v_i is chosen as the k-th number (p_k in the algorithm). $S_{\text{atleastonce}}$ ensures that each node is chosen at least once. Accordingly, S_{nottwice} ensures that each node is chosen no more than once: if v_k it is chosen as the i-th number, then it cannot be chosen again as later number j. S_{first} and S_{last} ensure that s and t are the first and last nodes of the path to be obtained. S_{path} encodes step 4. of the algorithm seen in the lecture: it disallows that a node v_i is followed by node v_j if there is no edge from the first to the second node.

(a) Consider the following directed graphs:



Encode the following problems into Boolean formulas as described above.

(i) $\langle G_1, v_1, v_2\rangle \in HAMPATH$ [1 mark]

(ii) $\langle G_2, v_1, v_3\rangle \in HAMPATH$ [1 mark]

(iii) $\langle G_3, v_1, v_4\rangle \in HAMPATH$ [1 mark]

(iv) $\langle G_4, v_1, v_4\rangle \in HAMPATH$ [1 mark]

(continued at next page)

(continued from previous page)

(b) *DIMACS* is a text-based format for Boolean formulas in CNF. Read the description at `http://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html`. Encode the following problems in *DIMACS*. Also state which $c_{i,j}$ is assigned which variable number. Afterwards, use the webpage `https://jgalenson.github.io/research.js/demos/minisat.html` to solve these *SAT* problems automatically. For each problem, provide the resulting output. If the problem is satisfiable, provide the Hamilton path which corresponds to the solution.

(i) $\langle G_1, v_1, v_2 \rangle \in HAMPATH$ [1 mark]

(ii) $\langle G_2, v_1, v_3 \rangle \in HAMPATH$ [1 mark]

(iii) $\langle G_3, v_1, v_4 \rangle \in HAMPATH$ [1 mark]

(iv) $\langle G_4, v_1, v_4 \rangle \in HAMPATH$ [1 mark]

(c) SAT problems can be solved in $\mathcal{O}(2^q \cdot p)$ where q is the number of variables and p the length of the formula (total number of literals). Let $\mathbf{EXPTIME} = \bigcup_k \mathbf{TIME}(2^{n^k})$ be the set of languages which are decidable in *exponential time* (Note that this is a larger class than $\mathbf{E} = \mathbf{TIME}(2^{\mathcal{O}(n)})$).

(i) Provide the complexity of deciding $HAMPATH$ via *SAT* using the $\mathcal{O}$-notation depending on the number of vertices n. [2 marks]

(ii) Is $HAMPATH$ in $\mathbf{EXPTIME}$? Justify your answer. [2 marks]

(iii) Note that $n!$ is asymptotically bounded by 2^{n^2}. Using the complexity of solving $HAMPATH$ by mapping it to *SAT* found above, is it possible to show that this method is asymptotically faster than the one from the lecture? Justify your answer. [2 marks]

(d) Describe an encoding of the following language to SAT:

$$ANYHAM = \{\langle G \rangle \mid G \text{ is a directed graph with a Hamiltonian path}\}$$

[2 marks]

(e) Describe an encoding of the following language to SAT:
$HCIRC = \{\langle G \rangle \mid G = (V, E)$ is a directed graph and it has a Hamiltonian path from some $s \in V$ to some $t \in V$ with $(t, s) \in E\}$ [2 marks]

(f) Describe how you could exclude a particular Hamiltonian path when constructing the SAT problem (e.g. because you have already found it and you want to find another one). [2 marks]

7. Consider the recurrences below defined for $n \geq 0$.
Note: The lecture discussing the methods necessary for solving this exercise is scheduled for Thursday, February 27th, 2020.

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ f(n-1) + 2n & \text{else} \end{cases}$$

$$g(n) = \begin{cases} 1 & \text{if } n = 0 \\ 2g(n-1) + 2n & \text{else} \end{cases}$$

(a) Give the values for the following items.

(i) $f(0)$ [1 mark]

(ii) $g(1)$ [1 mark]

(iii) $g(2)$ [1 mark]

(iv) $f(10)$ [1 mark]

(b) Give a closed-form solution to the recurrence for $f(n)$ and show your process for arriving at this solution. [2 marks]

(c) Prove the correctness of your solution in (b) by induction. [2 marks]

(d) Prove that $g(n) = 5 \cdot 2^n - 2n - 4$ by induction. [2 marks]