CSC2047 Theory of Computation
Assignment 2 (v4)
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.

This version contains the sample solution.

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]

No.
$n! + 3n^6 + 2n^3 = \mathcal{O}(n!)$

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

Yes.

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

Yes.
Comment: As seen in the lecture, we do not have to care about the base of logarithms.

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

Yes.
Comment: As seen in the lecture, we do not have to care about the base of logarithms.

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

No.
$3 \cdot 2^n = \mathcal{O}(2^n)$

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

Yes.

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

Yes.

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

Yes.

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

Yes.

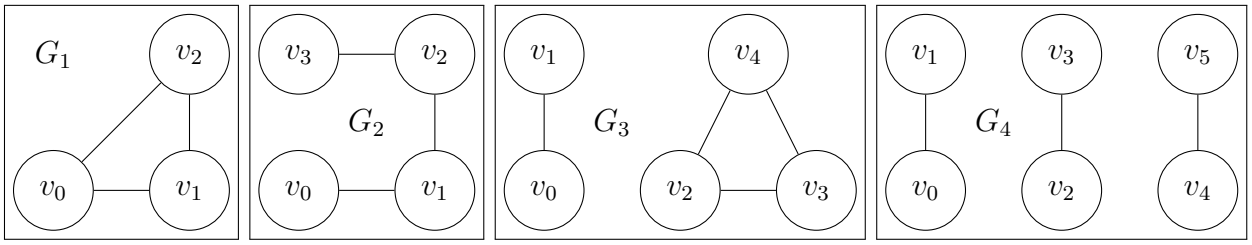
(j) $2n + \sqrt{n} = \mathcal{O}(n \log n)$ [1 mark]

Yes.

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]

Yes.

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

No.

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

No.

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

No.

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

Yes.

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

No.

(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]

M = "On input $\langle G, n \rangle$ where G is an undirected graph and n is an integer do:
1. Let $i = 0$.
2. While there are unmarked nodes in G repeat the following:
ww2.1. Select the first unmarked node of G and mark it.
ww2.2. Repeat the following stage until no new nodes are marked:
wwww2.2.1 For each node in G,
wwww2.2.1 mark it if it is attached by an edge to a node that is already marked.
ww2.3 Increase i by 1.
3. If $i = n$ accept, otherwise reject."

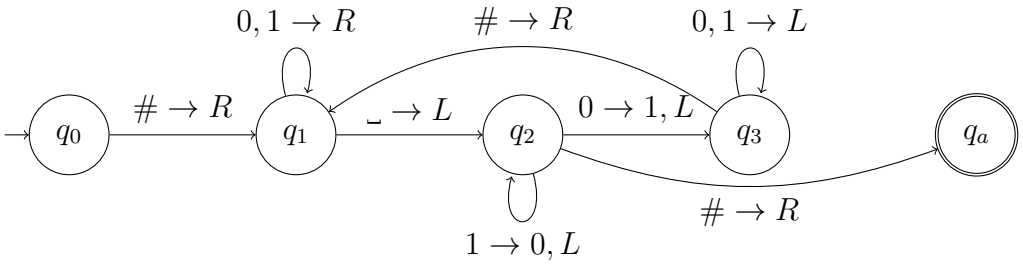
(c) Argue that your decider runs in polynomial time. Do so by reasoning about the runtime of its individual steps, the number of steps required, etc. as in the lecture. [2 marks]

Let m be the total size of the input encoding. Line 1. can be implemented in constant time $\mathcal{O}(1)$ or in the size of the input $\mathcal{O}(m)$ if we want to e.g. reserve the same amount of space as for n. For line 2., the check whether there are nodes remaining can be implemented in $\mathcal{O}(m)$, as we can search over the graph representation plus some auxiliary data to check whether a node has been marked. Also, the check cannot be executed more than m times. Thus, the total time for all all such checks is in $\mathcal{O}(m^2)$. A single execution of line 2.1. may take $\mathcal{O}(m)$ for scanning over the whole graph and auxiliary data structures while the marking of a node then takes $\mathcal{O}(1)$ time. Thus, the total time for line 2.1. is in $\mathcal{O}(m^2)$ as well. For line 2.2. the check can again be implemented in $\mathcal{O}(m)$. In total, in each iteration this check may take a maximum of $\mathcal{O}(m^2)$, and thus (because line 2. may be executed no more than $\mathcal{O}(m)$ times) in total no more than $\mathcal{O}(m^3)$. A single execution of 2.2.1 may take $\mathcal{O}(m^2))$ time: we can go over all nodes ($\mathcal{O}(m)$) which are checked and then go over all edges for this mode ($\mathcal{O}(m)$, and if the edge belongs to this node also mark the other node. As 2.2.1 be be executed $\mathcal{O}(m^2)$ times and thus in total takes time no more than $\mathcal{O}(m^4)$. 2.3 can be performed in no more than linear time, and thus in a total time of $\mathcal{O}(m^2)$. Line 3. requires no more than $\mathcal{O}(m)$ time.
Thus, the total required runtime of M is no more than

$$\underbrace{\mathcal{O}(m)}_{\text{line 1.}} + \underbrace{\mathcal{O}(m^2)}_{\text{check line 2.}} + \underbrace{\mathcal{O}(m^2)}_{\text{line 2.1}} + \underbrace{\mathcal{O}(m^3)}_{\text{check line 2.2.}} + \underbrace{\mathcal{O}(m^4)}_{\text{line 2.2.1.}} + \underbrace{\mathcal{O}(m^2)}_{\text{line 2.3.}} + \underbrace{\mathcal{O}(m)}_{\text{line 3.}} = \mathcal{O}(m^4).$$

Therefore, the complexity of this decider is polynomial, and thus it is in $\mathbf{P}$.

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]

$C_1 = q_0\#101$

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

$C_2 = \#q_1101$

(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]

Yes. $q_0\#$ $\#q_1$ $q_2\#$ $\#q_a$

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

No. $q_0\#\#$ $\#q_1\#$ $\#\#q_{\text{reject}}$

(iii) 0 [1 mark]

No. q_00 $0q_{\text{reject}}$

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

Yes. $\#q_10$ $\#q_20$ $\#q_11$ $\#q_21$ $\#q_a0$
$q_0\#0$ $\#0q_1$ $q_3\#1$ $\#1q_1$ $q_2\#0$

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

Yes. $q_0\#1$ $\#q_1 1$ $\#1q_1$ $\#q_2 1$ $q_2\#0$ $\#q_a 0$

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

Yes. $\#00q_1$ $\#q_1 01$ $\#q_2 00$ $\#10q_1$ $\#q_1 11$ $\#q_2 10$
$q_0\#00$ $\#0q_2 0$ $\#0q_1 1$ $q_3\#10$ $\#1q_2 0$ $\#1q_1 1$ $q_2\#00$
$\#q_1 00$ $\#q_3 01$ $\#01q_1$ $\#q_1 10$ $\#q_3 11$ $\#11q_1$ $\#q_a 00$
$\#0q_1 0$ $q_3\#01$ $\#0q_2 1$ $\#1q_1 0$ $q_3\#11$ $\#1q_2 1$

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

Yes. $q_0\#01$ $\#q_1 01$ $\#q_a 01$

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

Yes. $\#q_1 10$ $\#10q_1$ $\#q_3 11$ $\#q_1 11$ $\#11q_1$ $\#q_2 10$ $\#q_a 00$
$q_0\#10$ $\#1q_1 0$ $\#1q_2 0$ $q_3\#11$ $\#1q_1 1$ $\#1q_2 1$ $q_2\#00$

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

Yes. $q_0\#11$ $\#q_1 11$ $\#1q_1 1$ $\#11q_1$ $\#1q_2 1$ $\#q_2 10$ $q_2\#00$ $\#q_a 00$

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

Yes. $\#10q_1 1$ $\#q_3 110$ $\#11q_1 0$ $\#q_3 111$ $\#11q_1 1$ $q_2\#100$
$q_0\#101$ $\#101q_1$ $q_3\#110$ $\#110q_1$ $q_3\#111$ $\#111q_1$ $\#q_a 100$
$\#q_1 101$ $\#10q_2 1$ $\#q_1 110$ $\#11q_2 0$ $\#q_1 111$ $\#11q_2 1$
$\#1q_1 01$ $\#1q_2 00$ $\#1q_1 10$ $\#1q_3 11$ $\#1q_1 11$ $\#1q_2 10$

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

$$\#(0 \cup 1)^*$$

Comment: In the transition from q_0 to q_1, the machine makes sure that the first character is an $\#$. Then, in q_1, only 0, 1, and the blank character lead to non-rejecting states, such that after the initial $\#$ only a sequence of 0s and 1s can follows.

(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]

$$\#0^*$$

Comment: See the following exercise part for an explanation of what the Turing machine does. In the end, the automaton has changed all characters which previously were 0 or 1 to 0, such that the output word has the form given above.

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

$$f(n) = n2^n + 1$$

Comment: Consider the words of length n. The worst-case occurs for words of the form $w = \# \underbrace{0 \ldots 0}_{n-1 \text{ times}}$, that is, for a hash symbol followed by $n-1$ zeros. The Turing machine interprets the right part of the word after $\#$ as a binary number. The Turing machine increments this number by 1 until the number is $\underbrace{1 \ldots 1}_{n-1 \text{ times}}$ plus one further time such that an overflow occurs and the number wraps to $\underbrace{0 \ldots 0}_{n-1 \text{ times}}$. For the case of a word of the form w, this way in total the number is incremented 2^{n-1} times.
To increment the number by 1, the Turing machine first changes q_3 (or q_0 for the first round) to q_1 while moving the read/write head over the initial $\#$ symbol. Afterwards, in q_1 it moves the head further to the right while reading over the 0 and 1 of the number. Moving from the left to the right takes $\underbrace{1}_{\text{move over } \#} + \underbrace{(n-1)}_{\text{move over the 0s and 1s of the word}} =$ n steps. Eventually, the machine reaches a blank symbol and changes to q_2 while moving to the left. In q_2, the machine keeps replacing 1 by 0 while moving to the left. Once the machine sees a 0 on the tape, it replaces it by a 1 and changes to q_3. In q_3, the machine continues moving to the left skipping over the remaining 0 and 1 until it has reached hash symbol $\#$ on the leftmost position. On reaching this symbol, it moves to q_1 for the next increment.
Thus, for each increment, the Turing machine first moves the head to the right in n steps and then back to the left in another n steps. For a word of the form w, this happens 2^{n-1} times. Another step is required to acceptt the word when reaching the $\#$ already in q_2 rather than in q_3. Thus, for a word in form w, the total number of steps is

$$\underbrace{(\underbrace{n}_{\text{steps moving left to right}} + \underbrace{n}_{\text{steps moving right to left}})}_{\text{number of increments for } w} \cdot \underbrace{2^{n-1}}_{} + \underbrace{1}_{\text{final step for acceptance}}$$
$$= n2^n + 1$$

Note that the state q_3 would not really be required to increment the number repeatedlhy. The automaton could as well move back to q_1 directly after reading a 0 in q_2. However, then number of steps required for each increment would vary which would make the total number of steps harder to count.

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

$$g(n) = 1$$

Comment: This case occurs for words which do not start with #, because they are immediately rejected after one step.

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

$$(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_{\text{reject}})$$

where

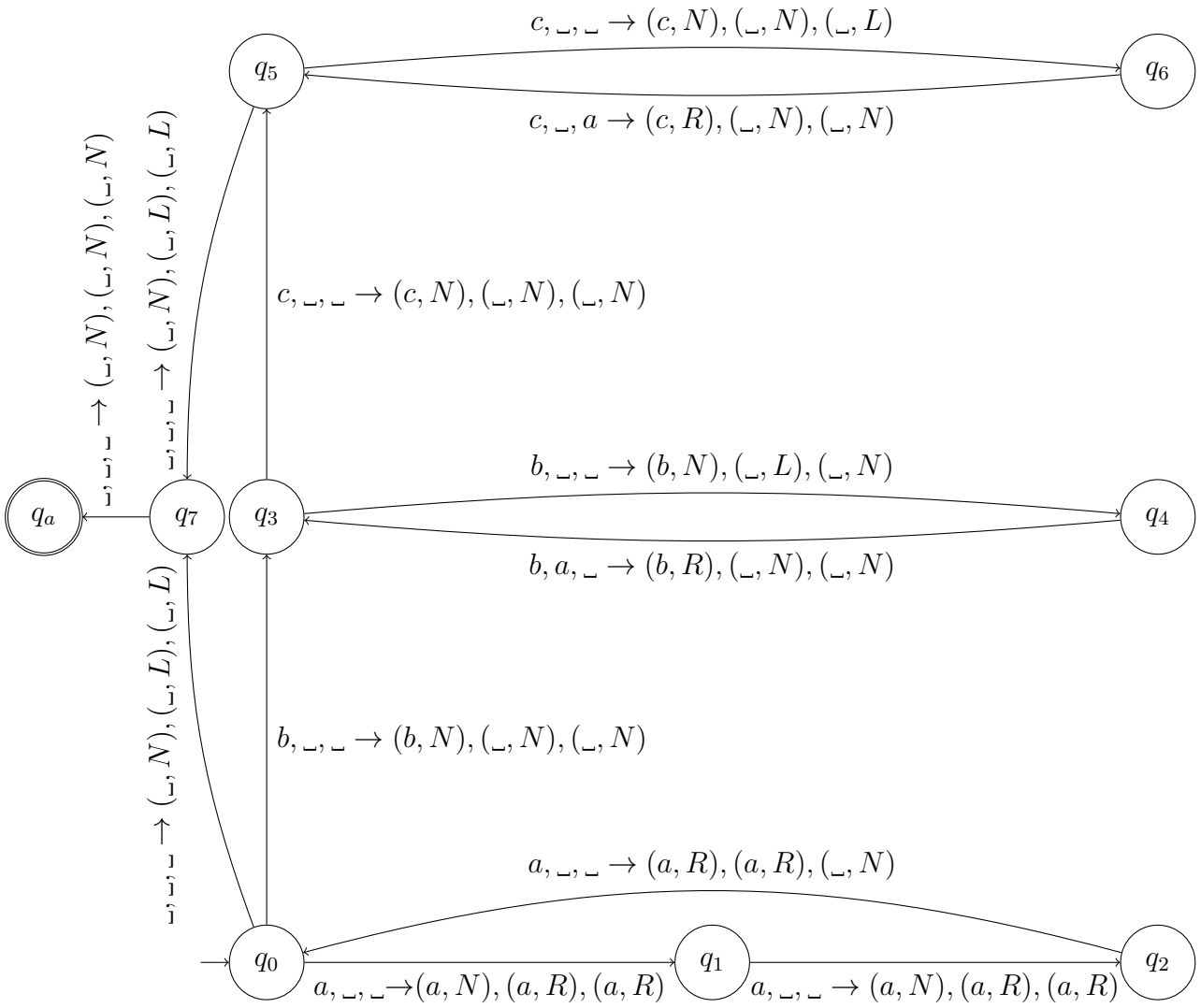
- $Q = \{q_0, q_1, q_2, q_3, q_a, q_{\text{reject}}\}$,
- $\Sigma = \{\#, 0, 1\}$,
- $\Gamma = \{\#, 0, 1, \text{␣}\}$,

and $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is given by the following table:

Q/Γ	$\#$	0	1	␣
q_0	$(q_0, \#, R)$	-	-	-
q_1	$-$	$(q_1, 0, R)$	$(q_1, 1, R)$	$(q_2, \text{␣}, L)$
q_2	$(q_a, \#, R)$	$(q_3, 1, L)$	$(q_2, 0, L)$	-
q_3	$(q_1, \#, R)$	$(q_3, 0, L)$	$(q_3, 1, L)$	-

Entries "-" correspond to moving to the rejecting state.

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]

Yes.	q_7; q_7; q_7
q_0; q_0; q_0	q_a; q_a; q_7

(ii) abc [1 mark]

No.
q_0abc; q_0; q_0
q_1abc; aq_1; aq_1
q_2abc; aaq_2; aaq_2
aq_0bc; $aaaq_0$; aaq_0
aq_3bc; $aaaq_3$; aaq_3
aq_4bc; aaq_4a; aaq_4
abq_3c; aaq_3; aaq_3
abq_5c; aaq_5; aaq_5
abq_6c; aaq_6; aq_6a
$abcq_5$; aaq_5; aq_5
$abcq_7$; aq_7a; q_7a
$abcq_{\text{reject}}$; $aq_{\text{reject}}a$; $q_{\text{reject}}a$

(iii) $abbbcc$ [1 mark]

Yes.
$q_0abbbcc$; q_0; q_0
$q_1abbbcc$; aq_1a; aq_1
$q_2abbbcc$; aaq_2; aaq_2
aq_0bbbcc; $aaaq_0$; aaq_0
aq_3bbbcc; $aaaq_3$; aaq_3
aq_4bbbcc; aaq_4a; aaq_4
abq_3bbcc; aaq_3; aaq_3
abq_4bbcc; aq_4a; aaq_4
$abbq_3bcc$; aq_3; aaq_3
$abbq_4bcc$; q_4a; aaq_4
$abbbq_3cc$; q_3; aaq_3
$abbbq_5cc$; q_5; aaq_5
$abbbq_6cc$; q_6; aq_6a
$abbbcq_5c$; q_5; aq_5
$abbbcq_6c$; q_6; q_6a
$abbbccq_5$; q_5; q_5
$abbbccq_7$; q_7; q_7
$abbbccq_a$; q_a; q_a

(iv) $abbb$ [1 mark]

No.
q_0abbb; q_0; q_0
q_1abbb; aq_1; aq_1
q_2abbb; aaq_2; aaq_2
aq_0bbb; $aaaq_0$; aaq_0
aq_3bbb; $aaaq_3$; aaq_3
aq_4bbb; aaq_4a; aaq_4
abq_3bb; aaq_3; aaq_3
abq_4bb; aq_4a; aaq_4
$abbq_3b$; aq_3; aaq_3
$abbq_4b$; q_4a; aaq_4
$abbbq_3$; q_3; aaq_3
$abbbq_7$; q_7; aq_7a
$abbbq_{\text{reject}}$; q_{reject}; $aq_{\text{reject}}a$

(v) bac [1 mark]

No.
q_0bac; q_0; q_0
q_3bac; q_3; q_3
q_3bac; q_4; q_4
$q_{\text{reject}}bac$; q_{reject}; q_{reject}

(vi) $aabbbbbbcccc$ [1 mark]

Yes.

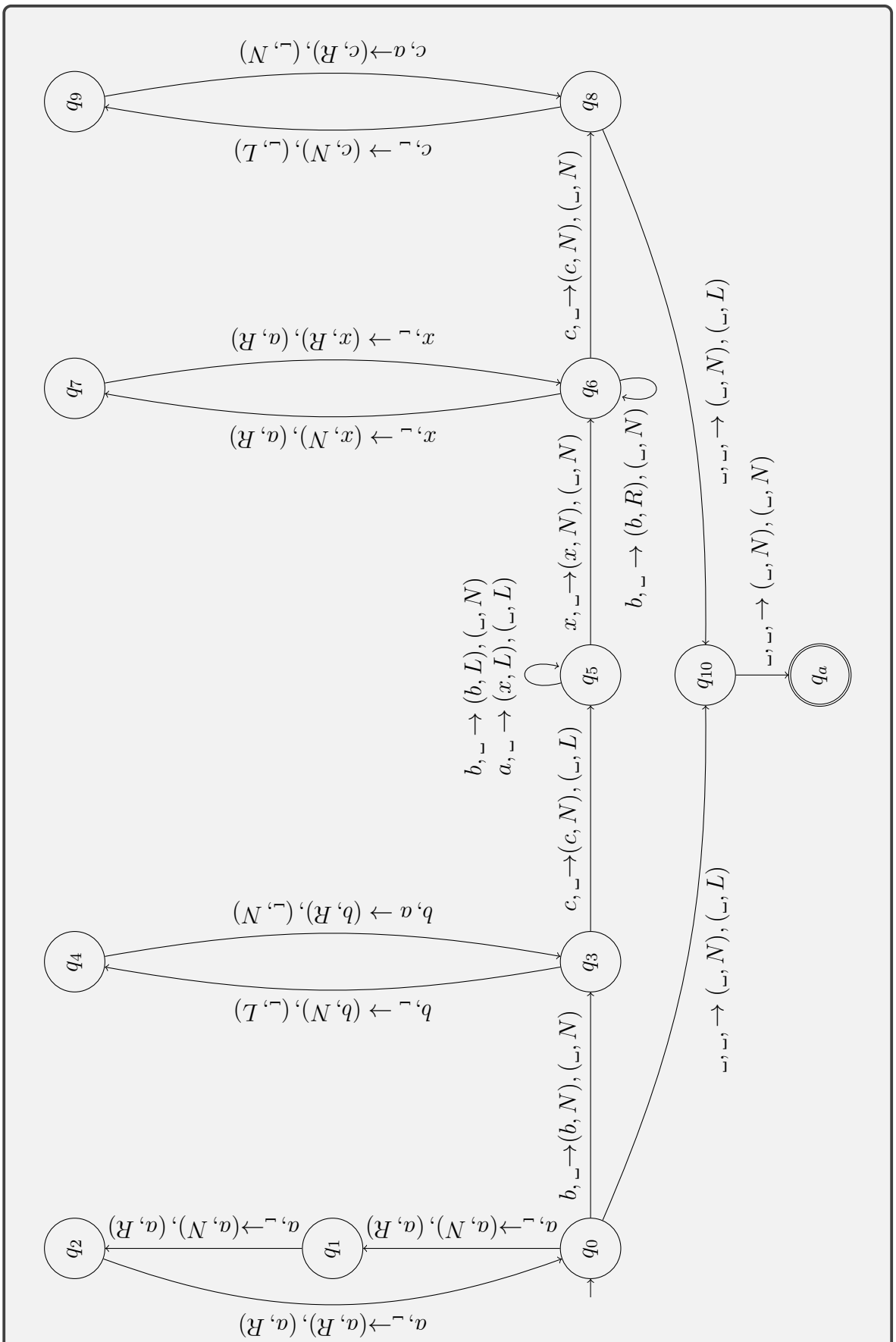
$q_0aabbbbbbcccc$; q_0; q_0
$q_1aabbbbbbcccc$; aq_1; aq_1
$q_2aabbbbbbcccc$; aaq_2; aaq_2
$aq_0abbbbbbcccc$; $aaaq_0$; aaq_0
$aq_1abbbbbbcccc$; $aaaaq_1$; $aaaq_1$
$aq_2abbbbbbcccc$; $aaaaaq_2$; $aaaaq_2$
$aaq_0bbbbbbcccc$; $aaaaaaq_0$; $aaaaq_0$
$aaq_3bbbbbbcccc$; $aaaaaaq_3$; $aaaaq_3$
$aaq_4bbbbbbcccc$; $aaaaaq_4a$; $aaaaq_4$
$aabq_3bbbbbcccc$; $aaaaaq_3$; $aaaaq_3$
$aabq_4bbbbbcccc$; $aaaaq_4a$; $aaaaq_4$
$aabbq_3bbbbcccc$; $aaaaq_3$; $aaaaq_3$
$aabbq_4bbbbcccc$; $aaaq_4a$; $aaaaq_4$
$aabbbq_3bbbcccc$; $aaaq_3$; $aaaaq_3$
$aabbbq_4bbbcccc$; aaq_4a; $aaaaq_4$
$aabbbbq_3bbcccc$; aaq_3; $aaaaq_3$
$aabbbbq_4bbcccc$; aq_4a; $aaaaq_4$
$aabbbbbq_3bcccc$; aq_3; $aaaaq_3$
$aabbbbbq_4bcccc$; q_4a; $aaaaq_4$
$aabbbbbbq_3cccc$; q_3; $aaaaq_3$
$aabbbbbbq_5cccc$; q_5; $aaaaq_5$
$aabbbbbbq_6cccc$; q_6; $aaaq_6a$
$aabbbbbbcq_5ccc$; q_5; $aaaq_5$
$aabbbbbbcq_6ccc$; q_6; aaq_6a
$aabbbbbbccq_5cc$; q_5; aaq_5
$aabbbbbbccq_6cc$; q_6; aq_6a
$aabbbbbbcccq_5c$; q_5; aq_5
$aabbbbbbcccq_6c$; q_6; q_6a
$aabbbbbbccccq_5$; q_5; q_5
$aabbbbbbccccq_7$; q_7; q_7
$aabbbbbbccccq_a$; q_a; q_a

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

$$\{a^i b^{3i} c^{2i} \mid i \geq 0\}$$

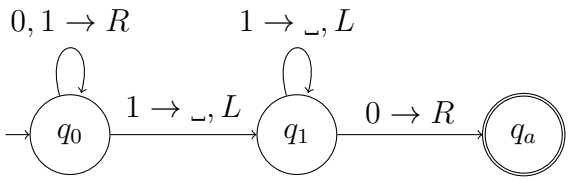
Comment: In states q_0, q_1, and q_2, the Turing machine reads over the initial sequence of as. Doing so, it writes a number of 3 times the number of as to tape 2 and writes a number of 2 times the number of as on tape 3. In states q_3 and q_4, it reads the bs, using the content of tape 2 to make sure that there are no less than 3 times the number of bs than as. States q_5 and q_6 have an according functionality for the cs. State q_7 is entered once the whole input word has been read. Its functionality is to make sure that there are not too many bs or cs, in which case there tapes 2 or 3 would be nonempty.

(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]



Comment: The Turing machine works similar to the three-tape machine. The difference is that the two-tape machine has to read the sequence of as twice to verify the correctness of the number of bs and then of cs. However, the runtime is still linear.

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]

No. q_0 $\_q_{\text{reject}}$

(ii) 0 [1 mark]

No. q_00 $0q_0$ 0_q_{reject}

(iii) 1 [1 mark]

No. q_01 $1q_0$ 1_q_{reject}

(iv) 01 [1 mark]

Yes. q_001 $0q_01$ q_10 $0q_a$

(v) 10 [1 mark]

No. q_010 q_1_0 $\_q_{\text{reject}}0$

(vi) 101 [1 mark]

Yes. q_0101 $1q_001$ $1q_001$ $10q_01$ $1q_10$ $10q_a$

(vii) 11100 [1 mark]

No. $1q_01100$ $111q_000$ $11100q_0$
q_011100 $11q_0100$ $1110q_00$ 11100_q_{reject}

(viii) 10100 [1 mark]

Yes.	$q_0 10100$	$1q_0 0100$	$10q_0 100$	$1q_1 0\_00$	$10q_a \_00$

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

$$(0 \cup 1)^* 0 (0 \cup 1)^* 1 (0 \cup 1)^*$$

Comment: In q_0, the automaton moves from the left to the right. At some point, if it sees a 1, it can decide whether to move to q_1. In q_1, it moves to the left as long as it sees only 1. Once it sees a 0, it moves to the accepting state. Therefore, in the word there must at some point be a 1, but before this 1, there has to be a 0 in the word.

(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]

$$f(n) = 2n$$

Comment: The worst case occurs in case we have a word consisting only of 1s. In this case, the automaton may move its read/write head until it is at the rightmost 1. This takes $n-1$ steps. Then, it may take the transition from q_0 to q_1, which takes 1 further step. Afterwards, the head is at the previous penultimate 1. Now, it deletes all $n-1$ remaining 1 still on the tape, which takes $n-1$ steps. Afterwards, the tape is all blank and the head is at the leftmost position. With 1 further step, the automaton then moves to the rejecting state. Thus, the total number of steps required is

$$\underbrace{n-1}_{\text{move to rightmost character}} + \underbrace{1}_{q_0 \text{ to } q_1} + \underbrace{n-1}_{\text{delete all 1s}} + \underbrace{1}_{\text{reject}} = 2n.$$

(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]

$$g(n) = n + 1$$

Comment: The best-case occurs in case we have a word consisting only of 0s. In this case, the Turing machine can only stay in q_0 and move to the right until it has reached over the word and the head is at a blank. This takes n steps. In 1 further step, the Turing machine rejects the word. Thus, the total number of steps required is

$$\underbrace{n}_{\text{read over word}} + \underbrace{1}_{\text{reject}}$$

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

$$n+1$$

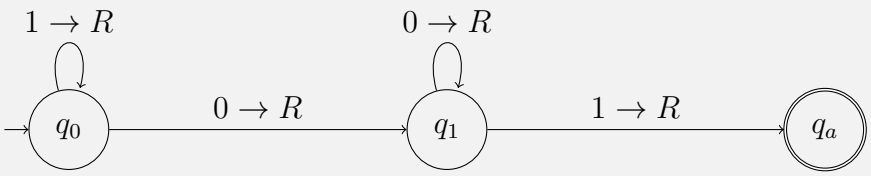
Comment: There is only one nondeterministic choice, in q_1 in which if the head is at a 1 the machine can either stay in q_0 or move to q_1. We have a maximum number of runs for a word which consists only of 1s. Thus, we have one run where the automaton stays in q_0 until it rejects. In addition, for each 1 of the word the Turing machine we have one run where it moves to q_1 instead.

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

$$1$$

Comment: We have a minimum number of runs for a word which consists only of 0s. In this case, the automaton can only stay in q_0, such that there is only 1 run in total.

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



Comment: The Turing machine stays in q_0 until it sees a 0. Then, it stays in q_1 until it sees a 1, after which it accepts.

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

$$(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_{\text{reject}})$$

where

- $Q = \{q_0, q_1, a_a, q_{\text{reject}}\}$,
- $\Sigma = \{0, 1\}$,
- $\Gamma = \{0, 1, \sqcup\}$,

and $\delta \colon Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$ is given by the following table:

Q/Γ	0	1	$\sqcup$
q_0	$\{(0, q_0, R)\}$	$\{(1, q_0, R), (\sqcup, q_2, L)\}$	-
q_1	$\{(0, q_a, R)\}$	$\{(\sqcup, q_1, L)\}$	-

Entries "-" correspond to moving to the rejecting state.

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:

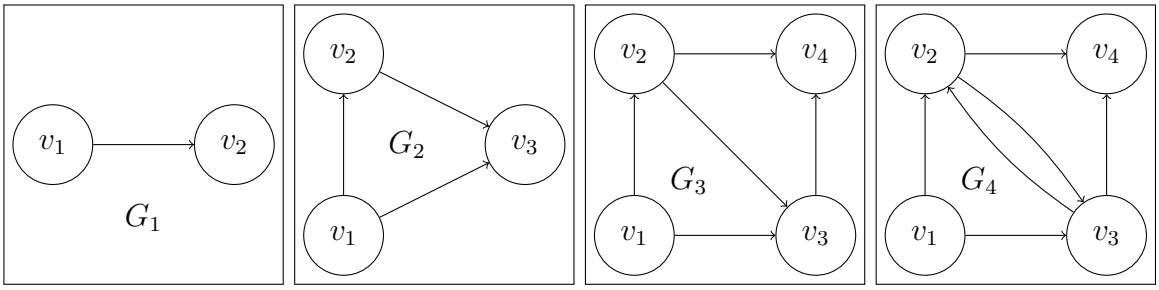
$$S = S_{\text{atleastonce}} \cup S_{\text{maxoneperstage}} \cup S_{\text{nottwice}} \cup S_{\text{first}} \cup S_{\text{last}} \cup S_{\text{path}}$$

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]

Variables: $C = \{c_{1,1}, c_{1,2}, c_{2,1}, c_{2,2}\}$

$$
\begin{aligned}
\phi = \quad & \\
& // \ S_{\text{atleastonce}} \\
& (c_{1,1} \vee c_{1,2}) \\
& \wedge (c_{2,1} \vee c_{2,2})
\end{aligned}
\qquad
\begin{aligned}
& // \ S_{\text{maxoneperstage}} \\
& \wedge (\overline{c_{1,1}} \vee \overline{c_{2,1}}) \\
& \wedge (\overline{c_{1,2}} \vee \overline{c_{2,2}}) \\
& // \ S_{\text{nottwice}} \\
& \wedge (\overline{c_{1,1}} \vee \overline{c_{1,2}}) \\
& \wedge (\overline{c_{2,1}} \vee \overline{c_{2,2}})
\end{aligned}
\qquad
\begin{aligned}
& // \ S_{\text{first}} \\
& \wedge c_{1,1} \\
& // \ S_{\text{last}} \\
& \wedge c_{2,2} \\
& // \ S_{\text{path}} \\
& \wedge \text{true}
\end{aligned}
$$

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

Variables: $C = \{c_{1,1}, c_{1,2}, c_{1,3}, c_{2,1}, c_{2,2}, c_{2,3}, c_{3,1}, c_{3,2}, c_{3,3}\}$

$$
\begin{aligned}
\phi = \quad & \\
& // \ S_{\text{atleastonce}} \\
& (c_{1,1} \vee c_{1,2} \vee c_{1,3}) \\
& \wedge (c_{2,1} \vee c_{2,2} \vee c_{2,3}) \\
& \wedge (c_{3,1} \vee c_{3,2} \vee c_{3,3}) \\
& // \ S_{\text{maxoneperstage}} \\
& \wedge (\overline{c_{1,1}} \vee \overline{c_{2,1}}) \\
& \wedge (\overline{c_{1,1}} \vee \overline{c_{3,1}}) \\
& \wedge (\overline{c_{2,1}} \vee \overline{c_{3,1}}) \\
& \wedge (\overline{c_{1,2}} \vee \overline{c_{2,2}}) \\
& \wedge (\overline{c_{1,2}} \vee \overline{c_{3,2}}) \\
& \wedge (\overline{c_{2,2}} \vee \overline{c_{3,2}}) \\
& \wedge (\overline{c_{1,3}} \vee \overline{c_{2,3}}) \\
& \wedge (\overline{c_{1,3}} \vee \overline{c_{3,3}}) \\
& \wedge (\overline{c_{2,3}} \vee \overline{c_{3,3}})
\end{aligned}
\qquad
\begin{aligned}
& // \ S_{\text{nottwice}} \\
& \wedge (\overline{c_{1,1}} \vee \overline{c_{1,2}}) \\
& \wedge (\overline{c_{1,1}} \vee \overline{c_{1,3}}) \\
& \wedge (\overline{c_{1,2}} \vee \overline{c_{1,3}}) \\
& \wedge (\overline{c_{2,1}} \vee \overline{c_{2,2}}) \\
& \wedge (\overline{c_{2,1}} \vee \overline{c_{2,3}}) \\
& \wedge (\overline{c_{2,2}} \vee \overline{c_{2,3}}) \\
& \wedge (\overline{c_{3,1}} \vee \overline{c_{3,2}}) \\
& \wedge (\overline{c_{3,1}} \vee \overline{c_{3,3}}) \\
& \wedge (\overline{c_{3,2}} \vee \overline{c_{3,3}})
\end{aligned}
\qquad
\begin{aligned}
& // \ S_{\text{first}} \\
& \wedge c_{1,1} \\
& // \ S_{\text{last}} \\
& \wedge c_{3,3} \\
& // \ S_{\text{path}} \\
& \wedge (\overline{c_{2,1}} \vee \overline{c_{1,2}}) \\
& \wedge (\overline{c_{2,2}} \vee \overline{c_{1,3}}) \\
& \wedge (\overline{c_{3,1}} \vee \overline{c_{1,2}}) \\
& \wedge (\overline{c_{3,2}} \vee \overline{c_{1,3}}) \\
& \wedge (\overline{c_{3,1}} \vee \overline{c_{2,2}}) \\
& \wedge (\overline{c_{3,2}} \vee \overline{c_{2,3}})
\end{aligned}
$$

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

Variables: $C = \{c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, c_{2,1}, c_{2,2}, c_{2,3}, c_{2,4}, c_{3,1}, c_{3,2}, c_{3,3}, c_{3,4}, c_{4,1}, c_{4,2}, c_{4,3}, c_{4,4}\}$

$$
\begin{aligned}
\phi = \\
&\text{// } S_{\text{atleastonce}} \\
&(c_{1,1} \vee c_{1,2} \vee c_{1,3} \vee c_{1,4}) \\
&\wedge (c_{2,1} \vee c_{2,2} \vee c_{2,3} \vee c_{2,4}) \\
&\wedge (c_{3,1} \vee c_{3,2} \vee c_{3,3} \vee c_{3,4}) \\
&\wedge (c_{4,1} \vee c_{4,2} \vee c_{4,3} \vee c_{4,4}) \\
&\quad \text{// } S_{\text{maxoneperstage}} \\
&\quad\wedge (\overline{c_{1,1}} \vee \overline{c_{2,1}}) \\
&\quad\wedge (\overline{c_{1,1}} \vee \overline{c_{3,1}}) \\
&\quad\wedge (\overline{c_{1,1}} \vee \overline{c_{4,1}}) \\
&\quad\wedge (\overline{c_{2,1}} \vee \overline{c_{3,1}}) \\
&\quad\wedge (\overline{c_{2,1}} \vee \overline{c_{4,1}}) \\
&\quad\wedge (\overline{c_{3,1}} \vee \overline{c_{4,1}}) \\
&\quad\wedge (\overline{c_{1,2}} \vee \overline{c_{2,2}}) \\
&\quad\wedge (\overline{c_{1,2}} \vee \overline{c_{3,2}}) \\
&\quad\wedge (\overline{c_{1,2}} \vee \overline{c_{4,2}}) \\
&\quad\wedge (\overline{c_{2,2}} \vee \overline{c_{3,2}}) \\
&\quad\wedge (\overline{c_{2,2}} \vee \overline{c_{4,2}}) \\
&\quad\wedge (\overline{c_{3,2}} \vee \overline{c_{4,2}}) \\
&\quad\wedge (\overline{c_{1,3}} \vee \overline{c_{2,3}}) \\
&\quad\wedge (\overline{c_{1,3}} \vee \overline{c_{3,3}}) \\
&\quad\wedge (\overline{c_{1,3}} \vee \overline{c_{4,3}}) \\
&\quad\wedge (\overline{c_{2,3}} \vee \overline{c_{3,3}}) \\
&\quad\wedge (\overline{c_{2,3}} \vee \overline{c_{4,3}}) \\
&\quad\wedge (\overline{c_{3,3}} \vee \overline{c_{4,3}}) \\
&\quad\wedge (\overline{c_{1,4}} \vee \overline{c_{2,4}}) \\
&\quad\wedge (\overline{c_{1,4}} \vee \overline{c_{3,4}}) \\
&\quad\wedge (\overline{c_{1,4}} \vee \overline{c_{4,4}}) \\
&\quad\wedge (\overline{c_{2,4}} \vee \overline{c_{3,4}}) \\
&\quad\wedge (\overline{c_{2,4}} \vee \overline{c_{4,4}}) \\
&\quad\wedge (\overline{c_{3,4}} \vee \overline{c_{4,4}}) \\
&\text{// } S_{\text{nottwice}} \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{1,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{2,2}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{2,3}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{3,2}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{3,3}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{4,2}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{4,4}}) \\
&\quad\text{// } S_{\text{first}} \\
&\quad\wedge c_{1,1} \\
&\quad\text{// } S_{\text{last}} \\
&\quad\wedge c_{4,4} \\
&\text{// } S_{\text{path}} \wedge (\overline{c_{1,1}} \vee \overline{c_{4,2}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{1,3}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{2,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{3,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{2,2}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{3,3}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{2,2}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{3,2}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{3,4}})
\end{aligned}
$$

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

Variables: $C = \{c_{1,1}, c_{1,2}, c_{1,3}, c_{1,4}, c_{2,1}, c_{2,2}, c_{2,3}, c_{2,4}, c_{3,1}, c_{3,2}, c_{3,3}, c_{3,4}, c_{4,1}, c_{4,2}, c_{4,3}, c_{4,4}\}$

$$
\begin{aligned}
\phi = \\
&\text{// } S_{\text{atleastonce}} \\
&(c_{1,1} \vee c_{1,2} \vee c_{1,3} \vee c_{1,4}) \\
&\wedge (c_{2,1} \vee c_{2,2} \vee c_{2,3} \vee c_{2,4}) \\
&\wedge (c_{3,1} \vee c_{3,2} \vee c_{3,3} \vee c_{3,4}) \\
&\wedge (c_{4,1} \vee c_{4,2} \vee c_{4,3} \vee c_{4,4}) \\
&\text{// } S_{\text{maxoneperstage}} \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{2,1}}) \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{3,1}}) \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{4,1}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{3,1}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{4,1}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{4,1}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{2,2}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{3,2}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{4,2}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{3,2}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{4,2}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{4,2}}) \\
&\wedge (\overline{c_{1,3}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{1,3}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{1,3}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{2,3}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{2,3}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{3,3}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{1,4}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{1,4}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{1,4}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{2,4}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{2,4}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{3,4}} \vee \overline{c_{4,4}}) \\
&\text{// } S_{\text{nottwice}} \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{1,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{2,2}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{2,3}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{3,2}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{3,3}} \vee \overline{c_{3,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{4,2}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{4,4}}) \\
&\text{// } S_{\text{first}} \\
&\wedge c_{1,1} \\
&\text{// } S_{\text{last}} \\
&\wedge c_{4,4} \\
&\text{// } S_{\text{path}} \\
&\wedge (\overline{c_{1,1}} \vee \overline{c_{4,2}}) \\
&\wedge (\overline{c_{1,2}} \vee \overline{c_{4,3}}) \\
&\wedge (\overline{c_{1,3}} \vee \overline{c_{4,4}}) \\
&\wedge (\overline{c_{2,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{2,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{2,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{3,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{3,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{3,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{1,2}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{1,3}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{1,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{2,2}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{2,3}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{2,4}}) \\
&\wedge (\overline{c_{4,1}} \vee \overline{c_{3,2}}) \\
&\wedge (\overline{c_{4,2}} \vee \overline{c_{3,3}}) \\
&\wedge (\overline{c_{4,3}} \vee \overline{c_{3,4}})
\end{aligned}
$$

(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]

Mapping of variables:

Variable	Number
$c_{1,1}$	1
$c_{1,2}$	2
$c_{2,1}$	3
$c_{2,2}$	4

Encoding into DIMACS:

```
p cnf 4 6
c S_atleastonce
1 2 0
3 4 0
c S_nottwice
-1 -2 0
-3 -4 0
c S_first
1 0
c S_last
4 0
c S_path
```

Output:

```
CPU time: 0.113s
SAT 1 -2 -3 4
```

Hamilton path:

$v_1 \to v_2$

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

Mapping of variables:

Variable	Number
$c_{1,1}$	1
$c_{1,2}$	2
$c_{1,3}$	3
$c_{2,1}$	4
$c_{2,2}$	5
$c_{2,3}$	6
$c_{3,1}$	7
$c_{3,2}$	8
$c_{3,3}$	9

Encoding into DIMACS:

```
p cnf 9 20
c S_atleastonce
1 2 3 0
4 5 6 0
7 8 9 0
c S_nottwice
-1 -2 0
-1 -3 0
-2 -3 0
-4 -5 0
-4 -6 0
-5 -6 0
-7 -8 0
-7 -9 0
-8 -9 0
c S_first
1 0
c S_last
9 0
c S_path
-4 -2 0
-5 -3 0
-7 -2 0
-8 -3 0
-7 -5 0
-8 -6 0
```

Output:

```
CPU time: 0.116s
SAT 1 -2 -3 4 -5 -6 -7 -8 9
```

Hamilton path:

$v_1 \rightarrow v_2 \rightarrow v_3$

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

Mapping of variables:

Variable	Number
$c_{1,1}$	1
$c_{1,2}$	2
$c_{1,3}$	3
$c_{1,4}$	4
$c_{2,1}$	5
$c_{2,2}$	6
$c_{2,3}$	7
$c_{2,4}$	8
$c_{3,1}$	9
$c_{3,2}$	10
$c_{3,3}$	11
$c_{3,4}$	12
$c_{4,1}$	13
$c_{4,2}$	14
$c_{4,3}$	15
$c_{4,4}$	16

Encoding into DIMACS:

```
p cnf 16 74
c S_atleastonce
1 2 3 4 0
5 6 7 8 0
9 10 11 12 0
13 14 15 16 0
c S_maxoneperstage
-1 -5 0
-1 -9 0
-1 -13 0
-5 -9 0
-5 -13 0
-9 -13 0
-2 -6 0
-2 -10 0
-2 -14 0
-6 -10 0
-6 -14 0
-10 -14 0
-3 -7 0
-3 -11 0
-3 -15 0
-7 -11 0
-7 -15 0
-11 -15 0
-4 -8 0
-4 -12 0
-8 -12 0
-8 -16 0
-12 -16 0
c S_nottwice
-1 -2 0
-1 -3 0
-1 -4 0
-2 -3 0
-2 -4 0
-3 -4 0
-5 -6 0
-5 -7 0
-5 -8 0
-6 -7 0
-6 -8 0
-7 -8 0
-9 -10 0
-9 -11 0
-9 -12 0
-10 -11 0
-10 -12 0
-11 -12 0
-13 -14 0
-13 -15 0
-13 -16 0
-14 -15 0
-14 -16 0
-15 -16 0
c S_first
1 0
c S_last
16 0
c S_path
-1 -14 0
-2 -15 0
-3 -15 0
-5 -2 0
-6 -3 0
-7 -4 0
-9 -2 0
-10 -3 0
-11 -4 0
-9 -6 0
-10 -7 0
-11 -8 0
-13 -2 0
-14 -3 0
-15 -4 0
-13 -6 0
-14 -7 0
-15 -8 0
-13 -10 0
-14 -11 0
-15 -12 0
```

Output:

```
 CPU time: 0.001s
SAT 1 -2 -3 -4 -5 6
-7 -8 -9 -10 11 -12
-13 -14 -15 16
```

Hamilton path:

$v_1 \to v_2 \to v_3 \to v_4$

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

Mapping of variables:

Variable	Number
$c_{1,1}$	1
$c_{1,2}$	2
$c_{1,3}$	3
$c_{1,4}$	4
$c_{2,1}$	5
$c_{2,2}$	6
$c_{2,3}$	7
$c_{2,4}$	8
$c_{3,1}$	9
$c_{3,2}$	10
$c_{3,3}$	11
$c_{3,4}$	12
$c_{4,1}$	13
$c_{4,2}$	14
$c_{4,3}$	15
$c_{4,4}$	16

Encoding into DIMACS:

```
p cnf 16 71
c S_atleastonce
1 2 3 4 0
5 6 7 8 0
9 10 11 12 0
13 14 15 16 0
c S_maxoneperstage
-1 -5 0
-1 -9 0
-1 -13 0
-5 -9 0
-5 -13 0
-9 -13 0
-2 -6 0
-2 -10 0
-2 -14 0
-6 -10 0
-6 -14 0
-10 -14 0
-3 -7 0
-3 -11 0
-3 -15 0
-7 -11 0
-7 -15 0
-11 -15 0
-4 -8 0
-4 -12 0
-8 -12 0
-8 -16 0
-12 -16 0
c S_nottwice
-1 -2 0
-1 -3 0
-1 -4 0
-2 -3 0
-2 -4 0
-3 -4 0
-5 -6 0
-5 -7 0
-5 -8 0
-6 -7 0
-6 -8 0
-7 -8 0
-9 -10 0
-9 -11 0
-9 -12 0
-10 -11 0
-10 -12 0
-11 -12 0
-13 -14 0
-13 -15 0
-13 -16 0
-14 -15 0
-14 -16 0
-15 -16 0
c S_first
1 0
c S_last
16 0
c S_path
-1 -14 0
-2 -15 0
-3 -15 0
-5 -2 0
-6 -3 0
-7 -4 0
-9 -2 0
-10 -3 0
-11 -4 0
-13 -2 0
-14 -3 0
-15 -4 0
-13 -6 0
-14 -7 0
-15 -8 0
-13 -10 0
-14 -11 0
-15 -12 0
```

Output:

```
 CPU time: 0.116s
SAT 1 -2 -3 -4 -5 6
-7 -8 -9 -10 11 -12
-13 -14 -15 16
```

Hamilton path:

$v_1 \to v_2 \to v_3 \to v_4$

(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]

We consider an arbitrary directed graph $G = (V, E)$ with n vertices and m edges. We assume $V = \{v_1, \ldots, v_n\}$. By definition of the transformation, we have $q = |C| = n^2$ variables in the resulting *SAT* problem.
We now consider the length p of the formula in terms of literals. We have

- for $S_{\text{atleastone}}$: exactly n^2 literals
- for $S_{\text{maxoneperstage}}$: $\mathcal{O}(n^2)$ literals - we do not nneed the exact number here, though it would be easy to compute
- for S_{nottwice}: $\mathcal{O}(n^2)$ literals
- for S_{first}: $\mathcal{O}(1)$ literals
- for S_{last}: $\mathcal{O}(1)$ literals
- S_{path}: $\mathcal{O}(n^2)$ literals

Thus, we have

$$p = \underbrace{n^2}_{S_{\text{atleastone}}} + \underbrace{\mathcal{O}(n^2)}_{S_{\text{maxoneperstage}}} + \underbrace{\mathcal{O}(n^2)}_{S_{\text{nottwice}}} + \underbrace{\mathcal{O}(1)}_{S_{\text{first}}} + \underbrace{\mathcal{O}(1)}_{S_{\text{last}}} + \underbrace{\mathcal{O}(n^2)}_{S_{\text{path}}} = \mathcal{O}(n^2)$$

Therefore, we have

$$\mathcal{O}(2^{n^2} \cdot \mathcal{O}(n^2)) = \mathcal{O}(2^{n^2} \cdot n^2).$$

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

We can further bound the complexity from the previous exercise part by e.g.

$$\mathcal{O}(2^{n^2} \cdot n^2) = \mathcal{O}(2^{n^3}).$$

Now, the number of variables n cannot be larger than the input size. Therefore, by solving *HAMPATH* via *SAT* we have found a method which solves this problem in $\mathcal{O}(2^{n^3})$. Thus, *HAMPATH* is decidable in exponential time and thus in $\mathbf{EXPTIME}$.

(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]

No. Because $n!$ is bounded by 2^{n^2}, the method from the lecture is in $\mathcal{O}(2^{n^2})$. This method is in $\mathcal{O}(2^{n^3})$, such that we cannot show that it is asymptotically faster.

(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]

We can use the same variables and subformulas $S_{\cdot}$ as in the original encoding and

then let

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

that is, we modify the original encoding by removing the requirements about a specific start and end node.

(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]

We can use the same variables and subformulas $S.$ as in the original encoding and then let

$$S = S_{\text{atleastonce}} \cup S_{\text{maxoneperstage}} \cup S_{\text{nottwice}} \cup \{\varphi_{\text{circ}}\} \cup S_{\text{path}}$$

that is, we modify the original encoding by removing the requirements about a specific start and end node and adding a new clause φ_{circ} with

$$\varphi_{\text{circ}} = \bigvee_{(v_j, v_i) \in E} c_{i,1} \wedge c_{j,n}.$$

which encodes the additional requirement that the Hamiltonian path is from some node s to some node t for which we have an edge from t to s.

(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]

Assume that we have found the Hamiltonian path $v_{i_1} \to v_{i_2} \to \ldots \to v_{i_n}$. We can exclude this path by adding the clause

$$\phi_{\text{exclude}} = \bigvee_{1 \leq j \leq n} \overline{c_{i_j, j}}$$

to S. This effect this has is follows: With $\overline{c_{i_j,j}}$ we disallow v_{i_j} as the jth node in the new path to be found. By building the disjunction over all j with $1 \leq j \leq n$, we ensure that at least one of the nodes is different from the one in the already known path. Therefore, adding ϕ_{exclude} we exclude that the same path is found twice. In the same way, one can continue excluding paths until all have been found and the problem is unsatisfiable.

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]

$$f(0) = 1$$

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

$$\begin{aligned} & g(1) \\ & = 2g(1-1) + 2 \cdot 1 \\ & = 2g(0) + 2 \\ & = 2 + 2 \\ & = 4 \end{aligned}$$

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

$$\begin{aligned} & g(2) \\ & = 2g(2-1) + 2 \cdot 2 \\ & = 2g(1) + 4 \\ & = 2 \cdot 4 + 4 \\ & = 12 \end{aligned}$$

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

$$
\begin{aligned}
& f(10) \\
&= f(9) + 20 \\
&= f(8) + 38 \\
&= f(7) + 54 \\
&= f(6) + 68 \\
&= f(5) + 80 \\
&= f(4) + 90 \\
&= f(3) + 98 \\
&= f(2) + 104 \\
&= f(1) + 108 \\
&= f(0) + 110 \\
&= 111
\end{aligned}
$$

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

$$
\begin{aligned}
& f(n) \\
&= f(n-1) + 2n \\
&= f(n-2) + 2n + 2(n-1) \\
&\ldots \\
&= f(n-n) + \underbrace{2n + 2(n-1) + \ldots + 2(n-(n-1))}_{n \text{ terms}} \\
&= f(0) + 2(\underbrace{n + (n-1) + \ldots + (n-(n-1))}_{n \text{ terms}}) \\
&= f(0) + 2(\underbrace{1 + 2 + \ldots + n}_{n \text{ terms}}) \\
&= 1 + 2 \underbrace{\sum_{i=1}^{n} i}_{=\frac{1}{2}n(n+1)} \\
&= 1 + 2\left(\frac{1}{2}n(n+1)\right) \\
&= n^2 + n + 1
\end{aligned}
$$

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

Induction start: $n = 0$:
We have by definition $f(0) = 1$ and also $0^2 + 0 + 1 = 1$. Thus, the induction start holds.
Induction hypothesis:

Assume that $f(n-1) = (n-1)^2 + (n-1) + 1$ for $n > 0$.
Induction step:
Then we have

$$\begin{aligned} & f(n) \\ & = f(n-1) + 2n \text{(by definition)} \\ & = ((n-1)^2 + (n-1) + 1) + 2n \text{(by induction hypothesis)} \\ & = n^2 - 2n + 1 + n - 1 + 1 + 2n \\ & = n^2 + n + 1 \end{aligned}$$

Thus, by mathematical induction, $f(n) = n^2 + n + 1$ for all $n \geq 0$.

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

Induction start: $n = 0$:
We have by definition that $g(0) = 1$ and also $5 \cdot 0^n - 2 \cdot 0 - 4 = 5 - 0 - 4 = 1$.
Thus, the induction start holds.
Induction hypothesis:
Assume that $g(n-1) = 5 \cdot 2^{n-1} - 2(n-1) - 4$ for $n > 0$.
Induction step:
Then we have

$$\begin{aligned} & g(n) \\ & = 2g(n-1) + 2n \text{(by definition)} \\ & = 2(5 \cdot 2^{n-1} - 2(n-1) - 4) + 2n \text{(by induction hypothesis)} \\ & = 2 \cdot 5 \cdot 2^{n-1} - 2 \cdot 2(n-1) - 2 \cdot 4 + 2n \\ & = 5 \cdot 2^n - 2 \cdot 2(n-1) - 2 \cdot 4 + 2n \\ & = 5 \cdot 2^n - 4(n-1) - 2 \cdot 4 + 2n \\ & = 5 \cdot 2^n - 4(n-1) - 8 + 2n \\ & = 5 \cdot 2^n - 4n + 4 - 8 + 2n \\ & = 5 \cdot 2^n - 2n - 4 \end{aligned}$$

Thus, by mathematical induction, $g(n) = 5 \cdot 2^n - 2n - 4$ for all $n \geq 0$.