ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Spring Semester 2024 Prof. R. Wattenhofer

Computational Thinking

Thursday, August 15, 2024, 10:00-12:00

Do not turn over until instructed to do so

This examination lasts for 120 minutes and comprises 120 points. There is one block of questions for each lecture chapter.

You may answer in German, English, or combine German and English.

Unless explicitly stated, you do **not** have to justify your answers. Writing down your thoughts (math, text, or annotated sketches), however, might help with the understanding of your approach. This may then result in points being awarded even if your answer is not correct. Please write legibly. Unreadable answers will not be graded.

Some questions will ask you to fill in answers in a template. If you decide to start over you will find fill-in replacements at the end of the examination booklet.

Please write your name and student number on every additional sheet. Please write your name and student number in the following fields on this cover sheet.

Family Name	**First Name**	**Student Number**

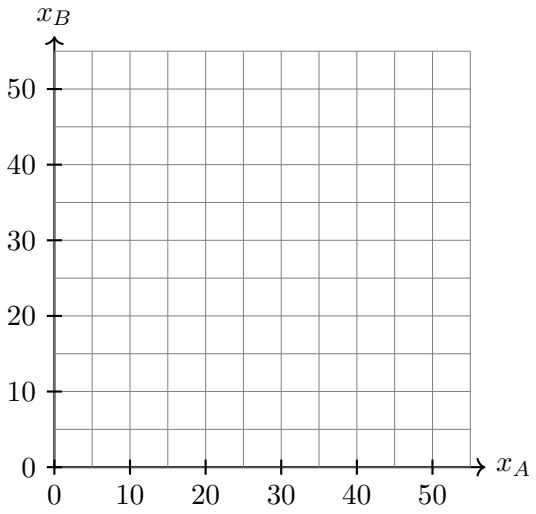
Task	**Achieved Points**	**Maximum Points**
1 - Algorithms		18
2 - Complexity		16
3 - Cryptography		19
4 - Databases		16
5 - Machine Learning		17
6 - Neural Networks		16
7 - Computability		18
Total		**120**

1 Algorithms (18 points)

You spent the entire summer preparing for the Computational Thinking exam. As you confidently submit your exam sheet, you suddenly realize that you completely neglected your two other courses: Advanced Swiss German (Course A) and Basic Rocket Science (Course B). Exam A is in exactly 4 days, and exam B in exactly 5. You can study at most 10 hours per day, and the probability of passing an exam is given by the product between the time, in hours, you will spend studying for that exam (x_A for A and x_B for B) and your learning rate for that course (l_A for A and l_B for B). If this product (i.e., $l_A \cdot x_A$ or $l_B \cdot x_B$) is at least 1, success in that exam is guaranteed.

a) `[6 points]` Given any positive l_A and l_B, write a linear program to minimize the total study time while **ensuring** success in both exams. Your LP must be expressed in the standard form $\max\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$ with $\mathbf{x} = [x_A, x_B]^T$.

b) `[4 points]` Given $l_A = \frac{1}{15}$ and $l_B = \frac{1}{5}$, draw below the feasible polytope of your linear program. If the polytope is non-empty, find the optimal solution.



c) [4 points] Now suppose that your learning rates l_A and l_B are such that you cannot ensure success in both exams. Therefore, instead of minimizing study time, you now want to maximize the joint probability of success of both exams. Can this new problem be formulated as a LP **in the same standard form as in a)** (also with $\mathbf{x} = [x_A, x_B]^T$)? Justify your answer.

d) [4 points] In general, what is the worst case time complexity of the simplex algorithm to solve an LP whose polytope has exactly k nodes and m edges. Justify your response.

Solution

a) The corresponding linear program (expressed as a minimization problem) is

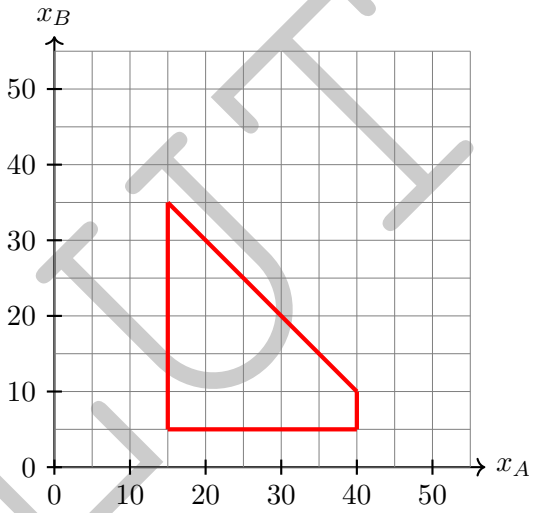
$$\begin{aligned} \text{Minimize} \quad & x_A + x_B \\ \text{subject to} \quad & l_A \cdot x_A \geq 1 \\ & l_B \cdot x_B \geq 1 \\ & x_A + x_B \leq 50 \\ & x_A \leq 40 \end{aligned}$$

This can be transformed in the standard form $\max\{\mathbf{c}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$ by setting

$$\mathbf{c}^T = [-1\ -1], \qquad \mathbf{A} = \begin{bmatrix} -l_A & 0 \\ 0 & -l_B \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} -1 \\ -1 \\ 50 \\ 40 \end{bmatrix}.$$

Note that $\mathbf{x} \geq 0$ is not needed due to the positiveness of l_A and l_B.

b) The feasible polytope is drawn below. The optimal solution is $x_A = 15$ and $x_B = 5$.



c) The joint probability of success in both exams is given by $\min\{1, l_A \cdot x_A\} \cdot \min\{1, l_B \cdot x_B\}$. This new problem can therefore be formulated as follows:

$$\begin{aligned} \text{Maximize} \quad & \min\{1, l_A \cdot x_A\} \cdot \min\{1, l_B \cdot x_B\} \\ \text{subject to} \quad & x_A + x_B \leq 50 \\ & x_A \leq 40 \\ & x_A, x_B \geq 0 \end{aligned}$$

The objective is non-linear in $[x_A, x_B]^T$, and therefore it is not a linear program.

d) The worst case occurs when the simplex algorithm visits all the nodes before finding the optimal one. At each step, to find the next node, the simplex algorithm must try each incident edge. The worst-time complexity is therefore $\mathcal{O}(m)$.

2 Complexity (16 points)

Recall the boolean satisfiability problem (SAT), where the goal is to decide whether a given formula in conjunctive normal form (CNF) admits a satisfying assignment. Here is an example of a CNF formula: $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y) \wedge (\neg y \vee z) \wedge (y)$. In this question, we investigate two variants:

- Positive-SAT: same as SAT, but no negations occur in the formula.
- Pretty-Positive-SAT: same as SAT, but in each clause, there can be at most one negated literal. In Pretty-Positive-SAT, for example $(x \vee y \vee \neg z)$ is a valid clause, while $(\neg x \vee \neg y)$ is not.

In answering the following, you may assume P $\neq$ NP.

a) `[2 points]` Is Positive-SAT in NP? ○ Yes ○ No
Reason (1 sentence):

b) `[4 points]` Is Positive-SAT in P? ○ Yes ○ No
Reason (1-2 sentences):

c) `[2 points]` Is Pretty-Positive-SAT in NP? ○ Yes ○ No
Reason (1 sentence):

d) `[8 points]` Is Pretty-Positive-SAT In P? ○ Yes ○ No
Reason (3-4 sentences):

Solution

- For Positive-SAT, the answer is always yes: make all variables true, so it's in **b)** P and hence also in **a)** NP (since P $\subseteq$ NP).
- For Pretty-Positive-SAT, the answer is always yes provided no clauses are of size 1. The algorithm is hence: while there is a clause of size 1, force the value of that variable and simplify the formula. If this ever leads to a clause that evaluates to false, then there is no solution. Otherwise, we can assign true to all remaining variables. Therefore, Pretty-Positive-SAT is in **d)** P and hence also in **c)** NP (since P $\subseteq$ NP).

3 Cryptography (19 points)

Encryption

Let $\mathsf{AES}(m, k)$ be the deterministic AES (Advanced Encryption Standard) block encryption function, which securely encrypts any 128-bit block message block m into a 128-bit ciphertext block using any 128-bit key k. We use AES to construct $\mathsf{Enc}(m, k)$, which can encrypt any message m (whose length is divisible by 128 bits) using any 128-bit key k.

```
1  def Enc(m, k):
2      Split m into r 128-bit blocks m_1, m_2, ..., m_r
3      IV = Rand(0, 2**128 - 1)   #IV = a uniformly random 128-bit block
4      c_0 = AES(IV, k)
5      for i = 1, 2, ..., r:
6          c_i = AES(m_i, k ⊕ IV)
7      return c_0; c_1; ...; c_r   #with ';' we denote concatenation
```

a) [3 points] Every day, Alice encrypts the message "Good day!" with the key k, and sends the ciphertext to Bob. Does Alice always send Bob the same ciphertext? ○ Yes ○ No
Reason (1-2 sentences):

b) [3 points] Suppose we modify line 7 of $\mathsf{Enc}(\mathsf{m}, \mathsf{k})$ so that it returns $c_1; \ldots; c_r$, without the c_0. Would Enc still work correctly as an encryption function? ○ Yes ○ No
Reason (1-2 sentences):

c) [5 points] Is Enc secure? That is, is it always computationally hard for an adversary who knows the encryptions $c_1, \ldots, c_\ell$ (where ℓ is not too large) of some secret messages $m_1, \ldots, m_\ell$ under some uniformly random secret key k to learn anything about $m_1, \ldots, m_\ell$ beyond their lengths? ○ Yes ○ No
Reason (3-4 sentences):

Zero-Knowledge Proofs

Peggy and Victor have a common connected graph $G = (V, E)$ with the vertices $1, 2, \ldots, n$. Peggy knows a 3-coloring of the graph, which means that she knows some $\mathsf{color}(v) \in \{1, 2, 3\}$ for each vertex v such that $\mathsf{color}(x) \neq \mathsf{color}(y)$ for every edge $\{x, y\} \in E$. Peggy wants to prove to Victor that G is 3-colorable (that a 3-coloring exists for G) without letting Victor learn her coloring. For this, Peggy and Victor run the following complete and sound in-person scheme.

1: **for** $i \in \{1, \ldots, \lambda|E|\}$ where $\lambda \geq 1$ is a statistical security parameter **do**
2: Victor goes outside the room.
3: For each vertex v, Peggy writes $\mathsf{color}(v)$ on a piece of paper, and puts it in the box v.
4: Peggy closes all the boxes $1, \ldots, n$ so that none of the colors are visible.
5: Peggy invites Victor inside, and Victor comes inside the room.
6: Victor asks Peggy to open the boxes x and y for some random edge $\{x, y\} \in E$.
7: Peggy opens the boxes x and y.
8: Victor checks that the boxes x and y contain distinct colors in $\{1, 2, 3\}$.
9: **end for**
10: Victor accepts the proof if and only if his checks were successful in every iteration.

---

d) `[8 points]` Peggy puts her vertex colors in boxes because she does not want to reveal her coloring to Victor. Unfortunately, this is not enough to make the scheme zero-knowledge. In a few sentences, explain why the scheme is not zero-knowledge, propose a modification to make it zero-knowledge, and explain how your proposal makes the scheme zero-knowledge.

Solution

a) No, Alice does not send Bob the same ciphertext every day. She randomizes her ciphertexts by choosing IV uniformly at random. She only sends the same ciphertext if she chooses the same IV, which happens with negligible probability.

b) No, Enc would no longer work correctly as an encryption function. Without c_0, even given the key k one cannot obtain IV, which is required to compute $k \oplus \text{IV}$ and decrypt $c_1, \ldots, c_r$.

c) No, Enc is not secure. Suppose we use it with some key k to encrypt any 256-bit message m whose first 128 bits are the same as its last 128 bits. Then, $\mathsf{Enc}(m, k)$ returns a 384-bit ciphertext c whose second 128-bit block equals its third 128-bit block. One can then tell based on c alone that the encrypted message m consists of some 128-bit block repeated twice. This is because the second and third 128-bit blocks of c being identical indicates that c is a valid encryption of not just m but also m with its first and second 128-bit blocks swapped, which means that the first and second 128-bit blocks of m must be equal for c to be decryptable into a single message.

d) In each iteration, Peggy reveals the colors of the endpoints of a single edge. Over $|E|$ iterations, Victor can learn the colors of the endpoints of all edges, and thus learn Peggy's coloring. To prevent this, Peggy can randomly permute her coloring each iteration. That is, in each iteration, she can choose a uniformly random permutation $\sigma : \{1, 2, 3\} \to \{1, 2, 3\}$, and for each $v \in V$ put the color $\sigma(\mathsf{color}(v))$ in box v. Then, in each iteration, the revealed edge endpoints will have freshly randomized distinct uniformly random colors, and thus Victor will not learn anything new.

4 Databases (16 points)

You are in charge of helping a texting app to manage and design its database. They use a table for users as follows:

email	first name	surname	birthday	favorite color
john.doe@example.com	John	Doe	1990-01-15	Blue
jane.smith@example.com	John	Smith	1985-06-23	Green
alice.jones@example.com	Alice	Jones	1992-12-11	Red
		⋮		

Table 1: Users table

a) `[2 points]` Among the already existing columns, which should be chosen as the primary key and why?

Due to a poor hiring decision, the current message storage schema is questionable. The previous data engineer created multiple tables to store conversation messages. There are n tables. For $i \in \{1, ..., n\}$, the table `msgi` contains all the i-th messages of conversations and has three attributes:

- `id` - the message ID,
- `content` - the message content,
- `next` - a foreign key pointing to the (i+1)-th message in the next table, `msg(i+1)`.

If a message is the last one in a conversation, its `next` attribute is `NULL`. We define the conversation id as the id of its first message. For example, if there are 2 conversations, conversation 1 being "Hi"→"Hallo" and conversation 2 being "Bonjour"→"Salut"→"Bye", the database would look like this:

msg1

id	content	next
1	Hi	4
2	Bonjour	3

msg2

id	content	next
3	Salut	5
4	Hallo	NULL

msg3

id	content	next
5	Bye	NULL

b) `[3 points]` Give a SQL query that counts the number of conversations that have exactly 3 messages.

c) [4 points] Assume $n \geq 500$, use SQL to count the number of conversations that have strictly less than 241 messages (you can compute it from the results of a few queries).

d) [5 points] Assume $n = 3$ only in this question (meaning a conversation has at most 3 messages). Provide a SQL query to retrieve all messages from conversation 1. The returned table must contain 4 columns: id, m1, m2, m3 with id the id of the conversation and mi the content of the i-th message, i ranging from 1 to 3.

e) [2 points] Suggest a better modeling scheme allowing to retrieve messages from a conversation more easily. Your scheme must maintain the same level of information as the current one.

Solution

a) Column *email* should be used because it is the only column that is unique for each user.

b) `SELECT COUNT(*) FROM msg3 WHERE next IS NULL;`

c) The idea is to first compute the number of conversations that have at least 241 messages. Then we can derive our target by computing the total number of conversations minus the number of conversations with more than 241 messages. We obtain the desired result by subtracting the result of query `SELECT COUNT(*) FROM msg1;` by the result of query `SELECT COUNT(*) FROM msg241;`.
Note: we can also directly obtain the result with the one-liner: `SELECT (SELECT COUNT(*) FROM msg1) - (SELECT COUNT(*) FROM msg241);`.

d)
```
SELECT msg1.id as id, msg1.content as m1, msg2.content as m2, msg3.content
as m3 FROM
msg1 LEFT OUTER JOIN msg2 ON msg1.next=msg2.id
LEFT OUTER JOIN msg3 ON msg2.next = msg3.id
WHERE msg1.id=1;
```

e) One possible answer is to use one single table `Messages` with the following attributes:

- `convid` - the conversation id,
- `msgid` - the id of the message within the conversation (eg. 4th message has msgid=4),
- `content` - the content of the message.

The tuple (convid, msgid) can work as the primary key.

5 Machine Learning (17 points)

Stochastic Gradient Descent (SGD) is defined as:

$$x_{t+1} = x_t - \eta \cdot \nabla f(x_t),$$

where $f()$ is the loss function, x is our parameter and η is the learning rate. Stochastic Gradient Descent with Momentum (SGDM) is defined as:

$$m_{t+1} = \beta \cdot m_t + (1 - \beta) \cdot \nabla f(x_t)$$
$$x_{t+1} = x_t - \eta \cdot m_{t+1},$$

where m_t is the momentum, β is a parameter, and m is initialized with $m_0 = 0$.

a) `[2 points]` For what β does SGDM reduce to SGD?

b) `[3 points]` For a typical deep neural network training, switching from SGD to SGDM, the per sample...

○ inference compute cost doubles. ○ training compute cost doubles.
○ training compute cost decreases. ○ compute cost stays approximately the same.

Reason (1-2 sentences):

c) `[3 points]` For a typical deep neural network training, switching from SGD to SGDM, the memory footprint...

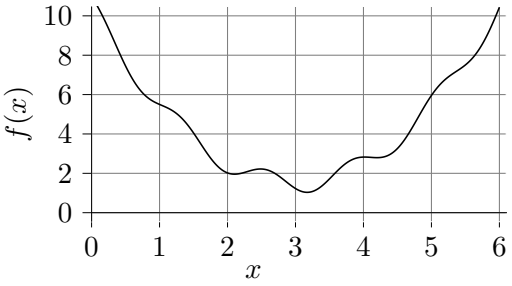
○ increases by the size of the model weights. ○ increases by the size of the activations.
○ decreases by the size of the activations. ○ stays approximately the same.

Reason (1-2 sentences):

Consider the function:

$$f(x) = (x-3)^2 + 0.5 \cdot \sin(5 \cdot x - 5) + 1.5$$

We use SGD and SGDM to minimize $f(x)$ with respect to x. The value of $f(x_{100})$ is illustrated in Figure 1. For SGDM $\beta = 0.9$ is used.



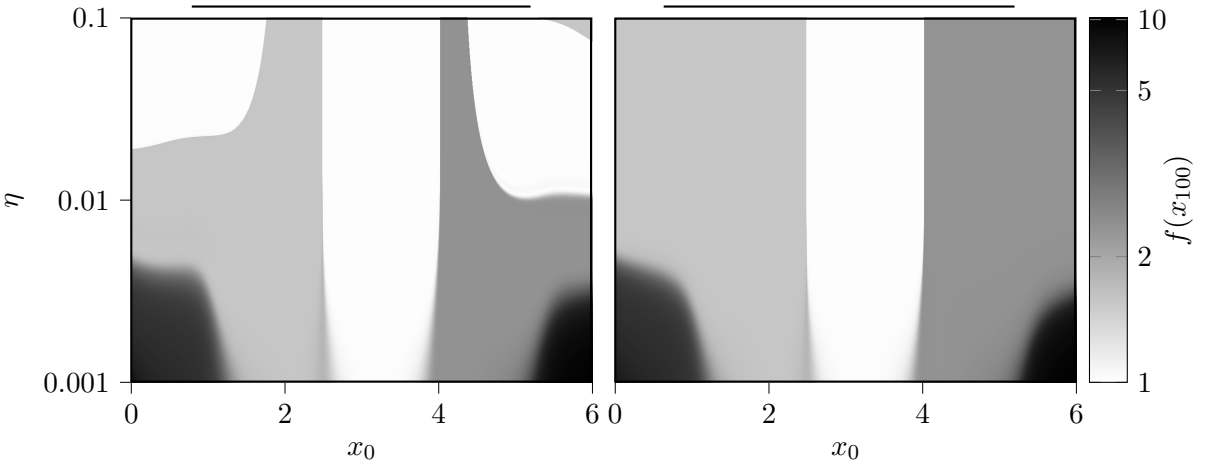


Figure 1: Optimizing $f(x)$ for 100 iterations with the two optimization algorithms.

d) `[5 points]` Identify the graphs in Figure 1 corresponding to SGD and SGDM. Write the algorithm above the graphs on the line. Explain your answer.

e) `[4 points]` In the left graph of Figure 1, why is it not converging to the global minimum for $x_0 = 6$ and $\eta = 0.1$, but converging for $x_0 = 5$ and $\eta = 0.1$?

Solution

a) $\beta = 0$

b) D. For a typical deep neural network training, the momentum update takes significantly less time than the gradient calculation.

c) A. The momentum must be stored for each weight doubling the model memory footprint.

d) Left: SGDM. Right: SGD. SGDM can escape the local minimum at $x \approx 2.1$ and $x \approx 4.2$ due to the added momentum term. Assuming a sufficiently high learning rate and distance from the local minimum, the momentum can enable SGDM to bypass the local minimum and move towards the global minimum.

e) For a sufficiently large learning rate and a starting point far from the local minimum, the SGDM can gather enough momentum to overshoot the global minimum. This can result in the algorithm getting stuck in a local minimum on the opposite side of the valley. For $x_0 = 5$ it does not gather enough momentum.

6 Neural Networks (16 points)

a) Fundamentals of Neural Networks

(i) `[2 points]` ReLU is non-differentiable and can therefore **not** be used as an activation function in neural networks.

○ True ○ False

Reason (1-2 sentences):

(ii) `[2 points]` If we want to learn how to classify images of digits (input image, output one of the 10 digits), a CNN will learn faster than an MLP.

○ True ○ False

Reason (1-2 sentences):

(iii) `[2 points]` Using an autoencoder with a much smaller latent dimension size than input/output dimension size (i.e. $m \ll n$) always incurs a reconstruction loss.

○ True ○ False

Reason (1-2 sentences):

We are now training a very simple MLP with a one dimensional hidden layer and activation functions f_1, and f_2. That is, given an input x, the output, $\hat{y}$, can be calculated as,

$$h_1 = f_1(\alpha_1 x + \beta_1) = f_1(y_1)$$
$$\hat{y} = f_2(\alpha_2 h_1 + \beta_2) = f_2(y_2).$$

We use the mean squared error as the loss function, so the loss given input x and output y is,

$$L(y, \hat{y}) = (\hat{y} - y)^2.$$

Write f_i' as the derivative of the activation functions with respect to their input.

b) (i) `[6 points]` What is the gradient of the loss function with respect to the weights $(\alpha_1, \beta_1, \alpha_2, \beta_2)$?

(ii) `[4 points]` Assume all weights are as follows: $\alpha_1 > 0, \alpha_2 > 0, \beta_1 = 0, \beta_2 = 0$. Also assume all inputs x are negative, and ReLU is used as one of the activation functions. Can the network learn? Justify your answer using the answer to the previous question (2 sentences).

Solution

a) Fundamentals of Neural Networks

(i) False

It suffices that subderivatives exist for a function to be useable for gradient optimization (cf. remarks for Definition 6.13).

(ii) True

Using weight sharing, a single filter can learn to pick up certain shapes (features) regardless of location, while an MLP would be required to learn the filter for every possible location.

(iii) False

If the data exists on a manifold/in a subspace of size at most m, then the model can perfectly learn to compress the data.

b) Gradients

(i) We can calculate the gradients using backpropagation

$$\begin{aligned}
\frac{\mathrm{d}L}{\mathrm{d}\alpha_2} &= 2(\hat{y}-y)f_2'(y_2)h_1 \\
\frac{\mathrm{d}L}{\mathrm{d}\beta_2} &= 2(\hat{y}-y)f_2'(y_2) \\
\frac{\mathrm{d}L}{\mathrm{d}\alpha_1} &= 2(\hat{y}-y)f_2'(y_2)\alpha_2 f_1'(y_1)x \\
\frac{\mathrm{d}L}{\mathrm{d}\beta_1} &= 2(\hat{y}-y)f_2'(y_2)\alpha_2 f_1'(y_1)
\end{aligned}$$

(ii) No. The gradient of ReLU is zero for negative inputs, so the gradients are zero for all parameters. Hence, the parameters will never be updated.

7 Computability (18 points)

Recall the Post Correspondence Problem (PCP): We are given a finite set of dominoes $S = \{D_1, \ldots, D_n\}$. Each domino D_i is of the form $\left[\frac{x_i}{y_i}\right]$, where the *top string* x_i and *bottom string* y_i are finite strings over some finite alphabet. A solution to PCP is a nonempty sequence of dominoes (possibly with repetitions) such that the texts formed by the concatenated top and bottom strings are equal.

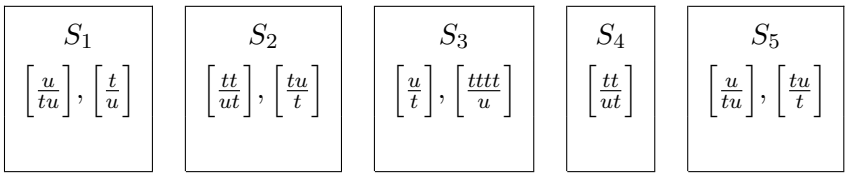
a) `[4 points]` Consider the PCP instance below. Explain in high level sentences (at most 2) why it does not have a solution.

$$\left[\frac{abb}{ba}\right], \left[\frac{aaa}{b}\right], \left[\frac{bb}{a}\right], \left[\frac{aa}{b}\right]$$

b) `[4 points]` Give a solution to the PCP instance with these two dominoes: $\left[\frac{bbbbb}{}\right], \left[\frac{b}{bbbb}\right]$. Please write your answer as the number of times each domino appears in the sequence, using the empty boxes in the exponents:

$$\left[\frac{bbbbb}{}\right]^{\square}, \qquad \left[\frac{b}{bbbb}\right]^{\square}$$

Consider a variant of the PCP, called *Set-PCP*. In this variant, dominoes come from a constant number $k \in \mathbb{N}$ of sets $S_1, \ldots, S_k$. The number of elements in each set is bounded by a constant. An example of a Set-PCP instance is shown below:



In Set-PCP, a sequence is only valid if it is formed by dominoes from sets in a **strictly increasing** order. This means that we must have $s_1 < s_2 < \cdots < s_\ell$, where s_j denotes the index of the set where the jth domino is taken, and ℓ is the length of the sequence.

c) `[4 points]` Give a solution to the Set-PCP instance above.

d) `[6 points]` Determining if an instance of Set-PCP has a solution is:

○ Solvable in polynomial time ○ Decidable but NP-hard ○ Undecidable

Reason (2-4 sentences):

Solution

a) There are many alternative ways to show this:

- There are no dominoes with the first letter of the top and bottom strings equal, which is a requirement for the first domino.
- There is no domino that can end the sequence, as there are no dominoes with the last character of the top and bottom strings equal.
- The top strings are longer than the bottom strings for all dominoes. We need dominoes with the bottom string longer to balance out the sequence.

b) One solution is $\left[\frac{bbbbb}{}\right]^3, \left[\frac{b}{bbbb}\right]^5$.

Reasoning (not required for the answer): using the fact that all the letters are the same, a general solution is of the form 5x - 3y = 0, where 5 and -3 are the differences between the number of letters in the top and bottom strings.

c) A valid solution is: $\left[\frac{tu}{t}\right], \left[\frac{tt}{ut}\right], \left[\frac{u}{tu}\right]$.

Reasoning (not required in the answer):

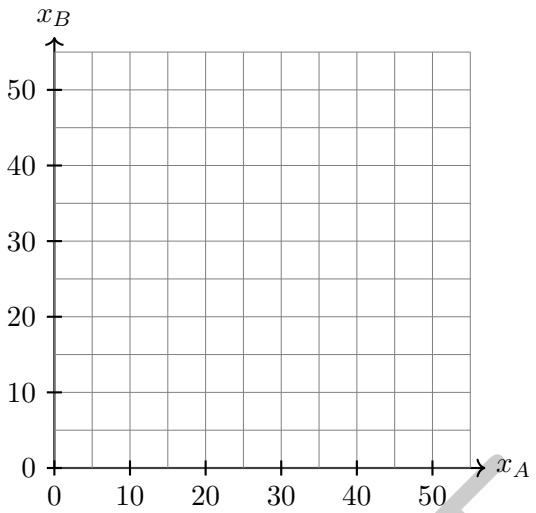
(i) For the first domino the first letters need to match. Possible choices are $\left[\frac{tu}{t}\right] \in S_2$ and $\left[\frac{tu}{t}\right] \in S_5$. The latter cannot be used, because we would have nothing to continue with.

(ii) The next domino must be from S_3, S_4 or S_5. The bottom text needs an **u**, so we could choose $\left[\frac{tt}{ut}\right] \in S_4$ or $\left[\frac{tttt}{u}\right] \in S_3$. The latter cannot be used, because the remaining dominoes don't have enough **t**s for the bottom text.

(iii) Since the previous domino was from S_4, we can only use dominoes from S_5. $\left[\frac{u}{tu}\right]$ completes the sequence.

d) Set-PCP is decidable in polynomial time. The number of possible sequences is bounded by $(|S_1| + 1) \cdot (|S_2| + 1) \cdot \cdots \cdot (|S_k| + 1)$ due to the fact that the index of the set in the sequence must be strictly increasing. Note that k is constant. Hence we can manually go through all the possible sequences in polynomial time.

Replacements



Use this page if you need extra space.