ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Spring Semester 2023 Prof. R. Wattenhofer

Computational Thinking

Friday, September 1, 2023, 14:00-16:00

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		19
3 - Cryptography		18
4 - Databases		19
5 - Machine Learning		15
6 - Neural Networks		18
7 - Computability		13
Total		**120**

1 Algorithms (18 points)

Disco Coins

Discoland uses coins of different integer denominations $D = \{d_1 = 1, d_2, \ldots, d_k\}$. You want to find the minimum number of coins needed to make a payment of an integer value $n \geq 0$. Note that coins can be used multiple times.

a) [6 points] Complete lines 3, 5 and 6 in the recursive algorithm below for finding the minimum number of coins to pay n.

```
1  def min_coins_R(D, n):
2      if n == 0:
3          return
4      else:
5          options = [min_coins_R(         ) for d in D if         ]
6          return
```

b) [4 points] Complete lines 6 and 10 in the dynamic programming implementation below.

```
1  import math
2  infinity = math.inf
3
4  def min_coins_DP(D, n):
5      table = [infinity for i in range(n + 1)]
6
7      for i in range(1, n + 1):
8          for d in D:
9              if (d <= i):
10                 rest_coins =
11                 if (rest_coins + 1 < table[i]):
12                     table[i] = rest_coins + 1
13     return table[n]
```

c) [2 points] What is the time complexity of a correct DP implementation?

d) [6 points] The **Greedy Algorithm** selects the largest denomination possible, subtracts it from the value and continues recursively with the remainder. For example, if the value is $n = 24$ and the denominations are $D = \{1, 2, 5, 10\}$, then the greedy solution will be 4 coins taken in the order $10, 10, 2, 2$.

Indicate whether Greedy is optimal in the following cases. Give a counterexample if not!

(i) $D = \{1, 2, 4, 8, 16\}$

☐ True ☐ False

Counterexample (if False):

(ii) $D = \{1, 2, 5, 10, 20\}$

☐ True ☐ False

Counterexample (if False):

(iii) $D = \{1, 4, 5, 10, 20\}$

☐ True ☐ False

Counterexample (if False):

Solution

a)
```python
def min_coins_R(D, n):
    if n == 0:
        return 0
    else:
        options = [min_coins_R(D, n-d) for d in D if d <= n]
        return min(options) + 1
```

b)
```python
import math
infinity = math.inf

def min_coins_DP(D, n):
    table = [infinity for i in range(n + 1)]
    table[0] = 0
    for i in range(1, n + 1):
        for d in D:
            if (d <= i):
                rest_coins = table[i - d]
                if (rest_coins + 1 < table[i]):
                    table[i] = rest_coins + 1
    return table[n]
```

c) $\mathcal{O}(nk)$

d) (i) True
(ii) True
(iii) False, e.g. $n = 8$

2 Complexity (19 points)

Multiple Choice

a) `[4 points]` For each of the following problems, decide whether they are in the complexity class P. Briefly explain your answer in one sentence.

Sorting an array of integers.
☐ yes ☐ no
Reason:

Given a graph $G = (V, E)$, an integer k, and a set $S \subseteq V$, determining whether S forms a vertex cover of size at most k.
☐ yes ☐ no
Reason:

b) `[4 points]` For each of the following statements about the complexity class NP, decide whether it is true or false. Briefly explain your answer in one sentence.

NP-complete problems are the most difficult problems in NP.
☐ yes ☐ no
Reason:

If we can find a polynomial time algorithm for some NP-hard problem, then P=NP.
☐ yes ☐ no
Reason:

Finding a Minimum Vertex Cover on a Tree

Let T be any tree with $n \geq 3$ vertices.

c) `[3 points]` Prove that T has a vertex cover of minimum size that does not contain any leaves (nodes with exactly one neighbor) of T.

d) `[8 points]` Develop a polynomial time algorithm that computes an minimum vertex cover for a given tree T. Describe the algorithm, and argue in 2 to 3 sentences why the solution obtained by your algorithm is a minimum vertex cover.

Solution

2.1 Multiple Choice

a) For each of the following problems, decide whether they are in the complexity class P. Briefly explain your answer in one sentence.

Sorting an array of integers.
☐ yes ☒ no
Reason: Sorting is not a decision problem

Given a graph $G = (V, E)$, an integer k, and a set $S \subseteq V$, determining whether S forms a vertex cover of size at most k.
☒ yes ☐ no
Reason: We can determine in polynomial time whether $|S| \leq k$ and whether S forms a vertex cover

b) For each of the following statements about the complexity class NP, decide whether it is true or false. Briefly explain your answer in one sentence.

NP-complete problems are the hardest problems in NP.
☒ yes ☐ no
Reason: NP-complete problems are in NP and every problem in NP polynomial time reduces to any NP-complete problem.

If we can find a polynomial time algorithm for an NP-hard problem, then P=NP.
☒ yes ☐ no
Reason: As every problem in NP polynomial time reduces to every NP-hard problem, a polynomial time algorithm to an NP-hard problem implies that we can solve every problem in NP in polynomial time.

2.2 Finding a Minimum Vertex Cover on a Tree

c) Consider a vertex cover S of optimal size for T. We can replace any leaf node v contained in S with its parent. As the degree of v is one, we know that v's parent covers all edges covered by v. Further, as $n \geq 3$, we know that v's parent is not a leaf node itself. The obtained set is a vertex cover for T, that has the same size as S.

d)

Algorithm 1 Vertex-Cover-Trees(T)

```
1: C ← ∅
2: while T has a node v of degree 1 do
3:     C ← C ∪ neighbor(v)
4:     remove neighbor(v) and all its incident edges from T
5: end while
6: Return C
```

To get full points for the argument students should argue as to why the solution obtained is (1) a valid vertex cover and why it (2) has minimum size, for example, as follows:

(1) Since the algorithm only removes edges covered by the set C and runs until all edges are removed, it returns a vertex cover.

(2) For every leaf in a tree, either the leaf or its neighbor need to be part of a vertex cover so that the edge to the leaf is covered. However, the parent always covers all edges the leaf does, and possibly more. So in order to find the smallest vertex cover, it is always "better" to choose the parent.

Below we give a full mathematical proof for the interested reader (which is not necessary for full points).

For the correctness proof, we establish two invariants on the while loop. Let $T(t)$ denote the graph after iteration t of the while loop, with $T(0)$ being the initial tree, and let $C(t)$ be the set C after iteration t, where $C(0) = \emptyset$.

Invariant 1: For every t and vertex cover X of $T(t)$, the set $X \cup C(t)$ forms a vertex cover for $T(0)$.

This easily follows from the fact, that in the while loop, we only remove edges from T for which one of the endpoints is in C. Hence if X covers all edges in $T(t)$, then $X \cup C(t)$ covers $T(0)$.

Invariant 2: For each $t \geq 0$ there exists an optimal solution with $C(t)$ as a subset.

This statement clearly holds for $t = 0$, as the emptyset is a subset of any optimal vertex cover. Suppose the statement holds for $C(t-1)$, and let v be the node of degree 1 we consider in round t. We know that either v or parent(v) must be in the solution. However, node v only covers a single edge, and this edge is also covered by parent(v). Therefore, in any solution that contains v we can replace v by parent(v) and still have a valid vertex cover.

Invariant 1 implies that the final solution must be a valid vertex cover, while invariant 2 implies optimality.

3 Cryptography (18 points)

Threshold Secret Sharing and MPC

When Oppenheimer invented the atomic bomb, he used a $(t = 2, n = 2)$-Shamir secret sharing scheme to split the bomb's detonation key $s \in \{0, ..., 18\}$ between Alice and Bob. Once this distribution was completed, Oppenheimer deleted the detonation key s.

Distribution:

$p(x) = s + a_1 x \mod 19$	random polynomial of Oppenheimer (with modulo computation)
$p(0) = s$	secret = detonation key
$p(1) = 16$	share sent to Alice
$p(2) = 6$	share sent to Bob

a) `[1 point]` What was Oppenheimer's primary concern that led him to take such security measures?

b) `[4 points]` Alice and Bob intend to include an additional share, $p(3)$, and send it to Charlie. Show that $p(3) = 2p(2) - p(1) \mod 19$, independent of which value of detonation key s and coefficient of the polynomial a_1 is chosen, and compute the numerical value of $p(3)$.

c) `[2 points]` What could be the reason for Alice and Bob to include an extra share, $p(3)$, in their arrangement with Charlie?

d) [3 points] After Alice and Bob jointly computed $p(3)$, Alice, having access to $p(3)$, wishes to independently detonate the bomb. Using only the values of $p(1)$ and $p(3)$, compute the numerical value of s to demonstrate Alice's ability to independently detonate the bomb.

e) [8 points] How can Alice and Bob securely compute and send $p(3)$ to Charlie without risking individual access to detonate the bomb? Design a protocol to achieve this. Show that the protocol is correct and the shares of Alice, Bob, and Charlie remain secret.
Hint. Use the fact from **c)**: $p(3) = 2p(2) - p(1) \mod 19$

```
1  def Secure_Share_Charlie():
2
3
4
5
6
7
8
9
10
```

(i) Correctness

(ii) Security

Solution

a) He was concerned about entrusting the sole authority of detonation to a single party.

b) Note that

$$p(3) = s + 3a_1 \tag{1}$$

Given

$$p(2) = s + 2a_1 = 6 \mod 19$$
$$p(1) = s + a_1 = 16 \mod 19$$

we show that $p(3) = 2p(2) - p(1)$ by multiplying the first equation by 2

$$2p(2) = 2s + 4a_1 = 12 \mod 19$$
$$p(1) = s + a_1 = 16 \mod 19$$

and subtracting them

$$2p(2) - p(1) = s + 3a_1 \overset{(1)}{=} p(3) \mod 19 \tag{2}$$

Hence, the value of $p(3) = 2p(2) - p(1) = 12 - 16 = -4 = 15 \mod 19$.

c) With only two shares, if one of the parties loses their share, they cannot reconstruct the detonation key anymore. With an additional share, even if one of the parties loses their share, the remaining two can still reconstruct the secret.

d) Alice can compute the secret since she has two evaluation points of the linear function. Alice can compute the secret s given,

$$p(1) = s + a_1 = 16 \mod 19$$
$$p(3) = s + 3a_1 = 15 \mod 19$$

by multiplying the first line by 3

$$3p(1) = 3s + 3a_1 = 48 \mod 19$$
$$p(3) = s + 3a_1 = 15 \mod 19$$

and subtracting them

$$3p(1) - p(3) = 2s = 48 - 15 = 33 = 14 \mod 19$$

Hence, $s = 14/2 = 7 \mod 19$.

e) From **c)** we know that $p(3) = 2p(2) - p(1)$, independent of which secret (detonation key) and coefficient of the polynomial is chosen. Hence, $p(3)$ is a linear combination of the shares of Alice and Bob. This is a typical problem for MPC. Different methods exist, here we provide a solution that is similar to `Sum_MPC` protocol from the lecture.

```
def Secure_Share_Charlie():
    Alice picks a random number r
    Alice sends m_1 = r - p(1) mod 19 to Bob and r to Charlie
    Bob sends m_2 = 2p(2) + m_1 mod 19 to Charlie
    Charlie computes p(3) = m_2 - r mod 19
```

(i) Correctness: Charlie computes $m_2 - r = 2p(2) + m_1 - r = 2p(2) + r - p(1) - r = 2p(2) - p(1) = p(3) \mod 19$

(ii) Security: Alice only sends messages so cannot learn any other share. Bob receives $m_1 = r - p(1) \mod 19$ with a random r, hence no information is revealed concerning Alice's share. Charlie receives the r and $m_2 = p(3) + r \mod 19$ and hence can only compute $p(3)$. Note that although all parties are aware of the relationship $p(3) = 2p(2) - p(1)$, no individual information about $p(1)$, $p(2)$, or $p(3)$ is disclosed. This is because multiple solutions exist for each share, and any of these solutions is equally likely.

4 Databases (19 points)

Multiple Choice

a) `[4 points]` When looking at data structures to use for a dynamic dictionary, which of the following statements are correct:

☐ yes	☐ no	An unsorted list can in principle be used as a data structure for a dynamic dictionary.
☐ yes	☐ no	Special measures need to be taken to ensure that binary search trees stay efficient during operation.
☐ yes	☐ no	Hash tables allow for the efficient search of all keys in a certain range.
☐ yes	☐ no	With a badly designed hash function, a hash table can be less efficient than a binary search tree.

University Database

We consider the SQL database of a university. The database contains information about the students, courses, enrollments and departments (primary keys are underlined):

students(<u>student_id</u>, first_name, last_name)
courses(<u>course_id</u>, course_name, department_id, credits)
enrollments(<u>enrollment_id</u>, student_id, course_id, semester, grade)
departments(<u>department_id</u>, department_name, department_head)

b) `[6 points]` While students enrolled into the "HS23" semester, a bug in the system led to some messed up values in the course_id field not corresponding to any course_id in the courses table. Write a SQL query that outputs the names of all affected students.

c) [3 points] What SQL commands need to be executed to avoid the issue of unmatched rows?

d) [6 points] Write a SQL query that outputs the first and last name of the student with the highest average grade for courses with more than 3 credits. You can assume that there is only one student with the highest average grade.

Solution

a)
- Yes: It will be very inefficient for lookups, but in principle it is possible.
- Yes: The trees need to stay balanced, otherwise a binary search tree could end up as a linked list.
- No: Due to the properties of the hash function, keys close to each before hashing will most likely not be close to each other after hashing
- Yes, if the hash function introduces a lot of collisions, the lookups will become very inefficient.

b)
```
SELECT first_name, last_name, course_name FROM students
JOIN enrollments ON enrollments.student_id = students.student_id
LEFT OUTER JOIN courses ON enrollments.course_id = courses.course_id
WHERE course_name IS null and semester = "HS23"
GROUP BY students.student_id
```

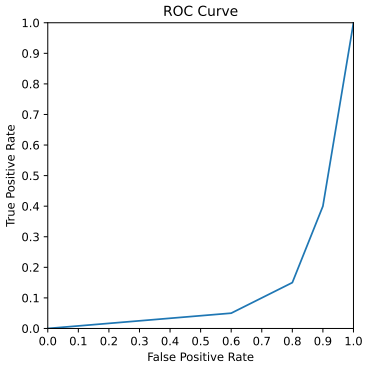
c) (i)
```
ALTER TABLE courses ADD FOREIGN KEY department_id REFERENCES departments
```
(ii)
```
ALTER TABLE enrollments ADD FOREIGN KEY student_id REFERENCES students
```
(iii)
```
ALTER TABLE enrollments ADD FOREIGN KEY course_id REFERENCES courses
```

d)
```
SELECT first_name, last_name from students
JOIN enrollments ON students.student_id = enrollments.student_id
JOIN (
      SELECT course_id FROM courses
      WHERE credits > 3
) AS c ON c.course_id = enrollments.course_id
GROUP BY students.student_id
ORDER BY AVG(grade) DESC
LIMIT 1
```

5 Machine Learning (15 points)

The Lord of the Evaluation

In *The Lord of the Rings*, Gandalf the wizard has an internal classifier to know when Gollum, the wretched fiend is lurking nearby. Analyzing Gandalf's classifier we find the following ROC curve:



a) `[2 points]` What is the problem with this classifier?

b) `[2 points]` What is the simplest solution to improve the classifier?

Having helped Gandalf, he re-calibrates his mind and tells us the raw probabilities he is now predicting, and if Gollum is indeed nearby.

The following set $\mathcal{P} = \mathcal{P}_0 \cup \mathcal{P}_1$ contains Gandalf's predictions:

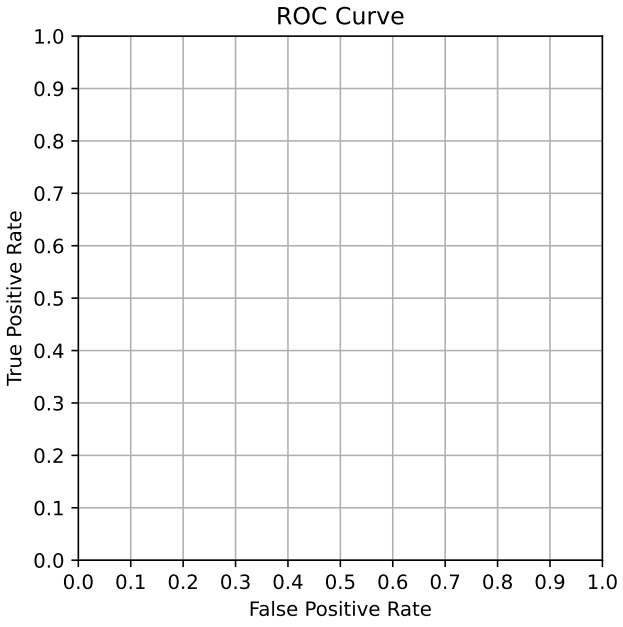
Gollum **not nearby**: $\mathcal{P}_0 = \{0.04, 0.1, 0.24, 0.29, 0.31, 0.33, 0.49, 0.52, 0.61, 0.85\}$

Gollum **nearby**: $\mathcal{P}_1 = \{0.39, 0.67, 0.73, 0.81, 0.9\}$

c) `[4 points]` To turn Gandalf's predictions listed in $\mathcal{P}$ into a usable classifier we need to apply a threshold. Which of the following thresholds T lead to a minimum number of misclassifications? Mark all that apply.

☐ 0.1 ☐ 0.2 ☐ 0.3 ☐ 0.4 ☐ 0.5 ☐ 0.6 ☐ 0.7 ☐ 0.8 ☐ 0.9

d) `[7 points]` Gandalf wants to draw an approximate ROC curve to better understand the performance of his new classifier, based on the predictions from $\mathcal{P}$. Help him by drawing the ROC curve by first plotting the points for thresholds $T = 0.3$, $T = 0.5$, $T = 0.8$, and linearly interpolating between them in the plot below.



Solution

The Lord of the Evaluation

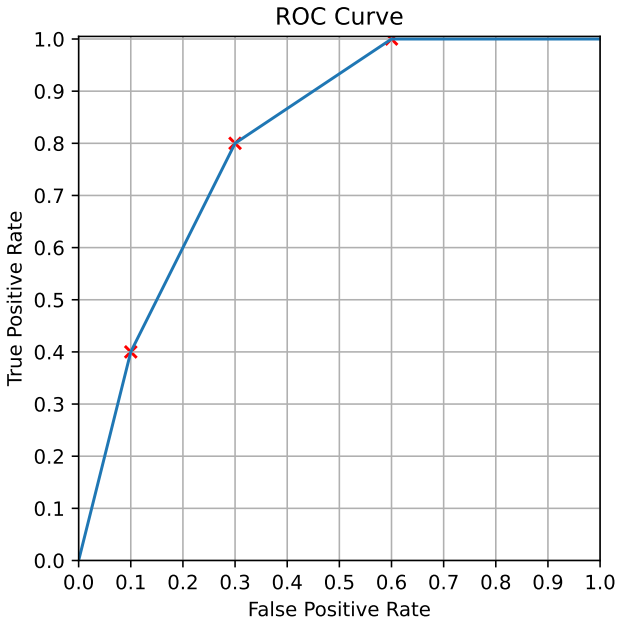
a) The classifier predicts more labels wrong than correct.

b) The easiest solution is to swap the predicted labels.

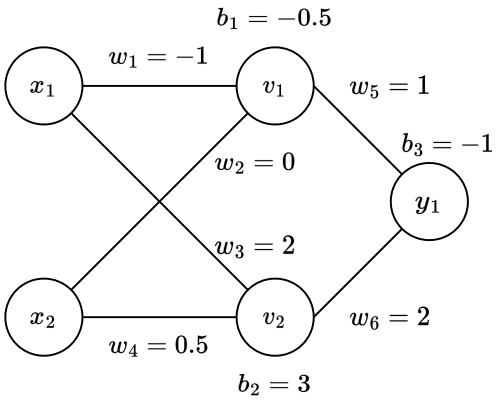
c) Ideal thresholds are 0.6 and 0.7 with 3 misclassifications each.

d) The x and y coordinates for a specific threshold are given by the TPR and FPR values:

	$T = 0.3$	$T = 0.5$	$T = 0.8$
True Positive Rate	5/5	4/5	2/5
False Positive Rate	6/10	3/10	1/10



6 Neural Networks (18 points)



a) `[4 points]` Consider the simple neural network computing the function f as given above, with all nodes using the ReLU activation. You are given two datapoints $d_1 = (1, 3)$ and $d_2 = (-2, 0)$. Biases are given explicitly as values at the nodes. What is the output of the network for these two values?

$f(d_1) =$ $\square$ $\quad$ $f(d_2) =$ $\square$

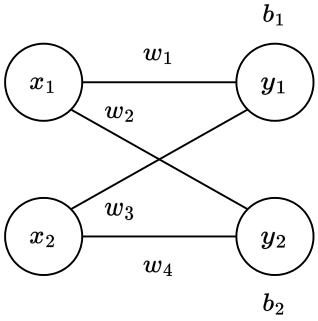
b) `[2 points]` The ground truth values for two new datapoints d_3 and d_4 are $\hat{f}(d_3) = 11$ and $\hat{f}(d_4) = 2$, while the predictions are $f(d_3) = 14$ and $f(d_4) = 0$. What is the MSE loss for the dataset $\{d_3, d_4\}$?

$L_{MSE} =$ $\square$

c) [6 points] Now assume that we only work with datapoint $d_5 = (1, 4)$ with $\hat{f}(d_5) = 14$ and that MSE loss is used. Compute the gradients on this datapoint for w_4, w_5 and w_6.

$\frac{\partial L_{MSE}}{\partial w_5} =$ $\frac{\partial L_{MSE}}{\partial w_6} =$ $\frac{\partial L_{MSE}}{\partial w_4} =$

d) [6 points] You are given the following network with two input numbers x_1 and x_2 and $x_1 \neq x_2$. Give weights and biases such that the output (y_1, y_2) is $(1, 0)$ iff $x_1 > x_2$ and $(0, 1)$ otherwise. You can choose the activation function freely (even non-differentiable functions are allowed, but the same activation function has to be used for all nodes).



$w_1 =$ $w_2 =$ $w_3 =$ $w_4 =$ $b_1 =$ $b_2 =$

Activation function:

Solution

a) $f(d_1) = 12, f(d_2) = 0.5$

b) $L_{MSE} = \frac{1}{2} \sum_{d \in \{d_1, d_2\}} (f(d) - \hat{f}(d))^2 = 6.5$

c) $\frac{\partial L_{MSE}}{\partial w_6} = \frac{\partial L_{MSE}}{\partial y_1} \cdot \frac{\partial y_1}{\partial w_6} = 2(y_1 - \hat{f}(d_5)) \cdot v_2 = -2 \cdot 7 = -14$

$\frac{\partial L_{MSE}}{\partial w_5} = 0$, because the gradient of the ReLU function at v_1 is 0.

$\frac{\partial L_{MSE}}{\partial w_4} = \frac{\partial L_{MSE}}{\partial y_1} \cdot \frac{\partial y_1}{\partial v_2} \cdot \frac{\partial v_2}{\partial w_4} = -2 \cdot w_6 \cdot x_2 = -2 \cdot 2 \cdot 4 = -16$

d) For example (other values are possible):

$w_1 = 1, w_2 = -1, w_3 = -1, w_4 = 1, b_1 = 0, b_2 = 0$

The activation function is the step function $\sigma(z)$ with $\sigma(z) = 0$ for $z < 0$ and $\sigma(z) = 1$ otherwise.

7 Computability (13 points)

Post Correspondence Problem (PCP)

a) [3 points] Consider the following PCP variant. For each domino (α, β) we have $|\alpha| = 2 \cdot |\beta|$. Is this variant of the PCP problem decidable or undecidable? Explain your answer.

b) [5 points] Consider the following domino set:

$$\left[\frac{ba}{a}\right], \quad \left[\frac{bb}{a}\right], \quad \left[\frac{d}{bbdc}\right], \quad \left[\frac{cc}{d}\right], \quad \left[\frac{aca}{ac}\right]$$

Is the corresponding PCP instance solvable? Explain your answer.

c) [5 points] Consider the following domino set:

$$\left[\frac{ca}{a}\right], \quad \left[\frac{bb}{a}\right], \quad \left[\frac{d}{bbdc}\right], \quad \left[\frac{cc}{d}\right], \quad \left[\frac{aca}{ac}\right]$$

Is the corresponding PCP instance solvable? Explain your answer.

Solution

Post Correspondence Problem (PCP)

a) *Decidable* – Since the top of the domino is always longer than the bottom of the domino, no solution is possible.

b) *Not solvable* – The last domino has to be domino 1, as it is the only domino that ends with the same letter. However, no domino can come before domino 1. Thus, the PCP is not solvable.

c) *Solvable* – The following sequence is valid:

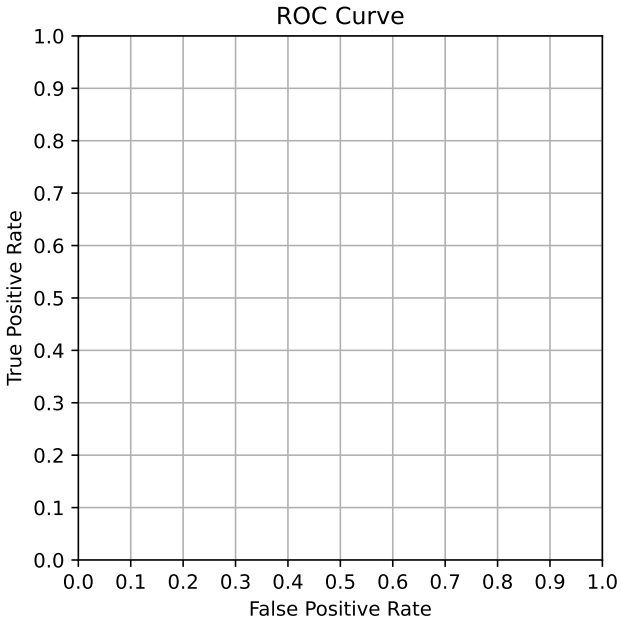
$$\left[\frac{aca}{ac}\right], \left[\frac{bb}{a}\right], \left[\frac{d}{bbdc}\right], \left[\frac{ca}{a}\right]$$

Thus, the PCP is solvable.

Replacements

Here you can find replacement templates for the fill in tasks, in case your original solution becomes too messy. If you use these, please **clearly** indicate which one we should consider.

Question 5



Use this page if you need extra space.