State Final Examination (Sample Questions)

2026-09-14

1 Shortlex order and NFA (shared topics)

Let $\Sigma = \{a_1 < a_2 < \ldots < a_k\}$ be a linearly ordered finite alphabet. On the set Σ^*, define the *shortlex* order as follows: for words $u, v \in \Sigma^*$, put $u \preceq v$ if either $|u| < |v|$, or $|u| = |v|$ and $u \leq_{\text{lex}} v$ in the lexicographic order induced by the given alphabet. (In other words, we first prefer shorter words, and if the lengths are equal, we compare words lexicographically.)

1. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic finite automaton (NFA). Give a full formal definition of the language $L(A)$ recognized by A.
2. Suppose that an NFA A is given on the input, where the alphabet Σ is given as a list of symbols, which also determines their alphabetical order. Describe (in pseudocode) an algorithm, as efficient as possible, that returns the smallest word in the language $L(A)$ with respect to the *shortlex* order, or reports that the language is empty.
3. Analyze the time complexity of your algorithm in terms of the number of states and the number of transitions of the automaton A. Consider the alphabet size to be constant.

If you use a standard graph search algorithm (such as DFS, BFS), you do not need to explain it in detail or prove its time complexity.

Solution sketch

1. A nondeterministic finite automaton is a tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is a finite *alphabet*, $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the *transition function*, $q_0 \in Q$ is the *initial state*, and $F \subseteq Q$ is the set of *accepting states*. Extend δ to words by $\delta^*(q, \varepsilon) = \{q\}$ and $\delta^*(q, wa) = \bigcup_{p \in \delta^*(q,w)} \delta(p, a)$. Then

$$L(A) = \big\{ w \in \Sigma^* \;\big|\; \delta^*(q_0, w) \cap F \neq \emptyset \big\}.$$

2. Consider the state diagram of A as a labelled directed graph on vertex set Q, with an edge $q \to p$ (labelled by a) whenever $p \in \delta(q, a)$ for some $a \in \Sigma$. First, compute for every $q \in Q$ the length $d(q)$ of the shortest word u such that $\delta^*(q, u) \cap F \neq \emptyset$. This can be done by a multi-source BFS on the *reverse* of the state diagram, starting from the set F of all accepting states. If q_0 is not reached, report that no such word exists and stop. Otherwise, let $\ell = d(q_0)$ be the length of the shortest word in $L(A)$.

   To construct the lexicographically minimal accepted word w of length ℓ, maintain a set $S \subseteq Q$ of current states and the remaining length r. Initialize $S := \{q_0\}$, $r := \ell$, and $w := \varepsilon$. In each round ($r > 0$), scan all transitions (q, a, p) with $q \in S$ and $d(p) = r - 1$. Among these, let $a_{\min}$ be the alphabetically smallest label. Append $a_{\min}$ to w, set $S := \{p \in Q \mid \exists q \in S :\ p \in \delta(q, a_{\min}) \ \wedge\ d(p) = r - 1\}$, and decrement r. When $r = 0$, output w.
3. Let $n = |Q|$ and $m = \sum_{q \in Q,\, a \in \Sigma} |\delta(q, a)|$. Computing the values $d(q)$ by reverse multi-source BFS takes $O(n+m)$ time. In the construction phase, all states in S satisfy $d(p) = r-1$, so each state can enter S at most once. Consequently, each transition is examined at most once overall. This phase can be computed in $O(m)$ time if we first discard transitions (q, a, p) with $d(p) > d(q) - 1$ and group states according to the value of $d(q)$. Hence the whole algorithm runs in $O(n + m)$ time.

2 Exception-safe file processing (shared topics)

Choose one of C++, C#, or Java and use the customary resource-management idioms of the selected language.

Implement a function

```
transformFile(inputFilePath, outputFilePath, operation)
```

which incrementally reads all records from the file at `inputFilePath`, transforms each record exactly once using the user-supplied behavior `operation`, and writes the results in the same order to the file at `outputFilePath`. Represent `operation` in the customary way for a "callable object" in the selected language. The input may be too large to fit into memory in its entirety.

The following language-neutral API is available. `Record` is a supplied type representing one record; its internal structure is irrelevant to this question. The `resource` designation denotes an object that must be released deterministically, including in the presence of exceptions. Adapt the pseudocode to the syntax and conventions of the selected language.

```
// An opaque supplied type representing one record.
type Record

resource RecordReader {
    // Returns the next record or indicates the end of input.
    // May throw an exception.
    read() -> Record or no record
}

resource TransactionalRecordWriter {
    // Adds a record to the pending output.
    // The data is not yet visible in the target file.
    // May throw an exception.
    write(record: Record)

    // Atomically replaces the target file with all records written so far.
    // May throw an exception.
    commit()
}
```

The pseudocode does not prescribe how a `RecordReader` for `inputFilePath` or a `TransactionalRecordWriter` for `outputFilePath` is obtained. In the selected language, this may, for example, use a constructor or factory function. Creating either resource may throw an exception.

A `TransactionalRecordWriter` additionally has the following semantics:

- `commit` may be performed at most once;
- if `commit` fails, the target file remains unchanged;
- releasing an uncommitted writer discards all pending data and releases its resources;
- releasing a committed writer only releases its resources.

For the purposes of this task, assume that releasing a resource completes without throwing an exception. The `operation` behavior may throw an exception, as may the operations marked above.

Complete the following tasks:

1. Rewrite the declarations of the `RecordReader` and `TransactionalRecordWriter` types, the transformation behavior `operation`, and the `transformFile` function in the selected language. Add a suitable way to obtain both resources from the file paths. Do not implement the bodies of the supplied operations. Briefly state:
   - how the selected language expresses or documents that individual operations may throw exceptions;
   - which language mechanism ensures deterministic resource release, including in the presence of exceptions.
2. Implement `transformFile`. The implementation must satisfy the following requirements:
   - every record read is transformed exactly once;
   - records are processed incrementally without constructing a collection containing the complete input;
   - all resources that are created are always released;
   - `commit` is performed exactly once and only after all records have been successfully read, transformed, and written;

– the original target file remains unchanged after any failure;

– the exception is propagated to the caller.

Solution sketch 1. Declarations, exceptions, and resource management

Java

```
class RecordReader implements AutoCloseable {
    RecordReader(String inputFilePath) throws Exception {
        // Supplied implementation.
    }

    Record read() throws Exception {
        // Supplied implementation.
    }

    @Override
    public void close() {
        // Supplied implementation.
    }
}

class TransactionalRecordWriter implements AutoCloseable {
    TransactionalRecordWriter(String outputFilePath) throws Exception {
        // Supplied implementation.
    }

    void write(Record record) throws Exception {
        // Supplied implementation.
    }

    void commit() throws Exception {
        // Supplied implementation.
    }

    @Override
    public void close() {
        // Supplied implementation.
    }
}

@FunctionalInterface
interface RecordOperation {
    Record apply(Record record) throws Exception;
}

static void transformFile(
        String inputFilePath,
        String outputFilePath,
        RecordOperation operation) throws Exception;
```

Java lists checked exceptions in declarations using `throws`. According to the specification, the `close` operations do not throw. Deterministic release is provided by try-with-resources together with the `AutoCloseable` interface; `close` is called after normal completion and when the block is left because of an exception.

C#

```
class RecordReader : IDisposable
{
```

```
    public RecordReader(string inputFilePath)
    {
        // Supplied implementation.
    }

    public Record? Read()
    {
        // Supplied implementation.
    }

    public void Dispose()
    {
        // Supplied implementation.
    }
}

class TransactionalRecordWriter : IDisposable
{
    public TransactionalRecordWriter(string outputFilePath)
    {
        // Supplied implementation.
    }

    public void Write(Record record)
    {
        // Supplied implementation.
    }

    public void Commit()
    {
        // Supplied implementation.
    }

    public void Dispose()
    {
        // Supplied implementation.
    }
}

static void TransformFile(
    string inputFilePath,
    string outputFilePath,
    Func<Record, Record> operation);
```

C# does not list thrown exceptions in method signatures; this behavior is described in the contract documentation. Deterministic release is provided by the `using` statement together with the `IDisposable` interface; `Dispose` is called after normal completion and when the block is left because of an exception.

C++

```
#include <filesystem>
#include <functional>
#include <optional>

class RecordReader {
public:
    explicit RecordReader(
        const std::filesystem::path& inputFilePath);
    std::optional<Record> read();
```

```
    ~RecordReader() noexcept;
};

class TransactionalRecordWriter {
public:
    explicit TransactionalRecordWriter(
        const std::filesystem::path& outputFilePath);
    void write(const Record& record);
    void commit();
    ~TransactionalRecordWriter() noexcept;
};

using RecordOperation = std::function<Record(const Record&)>;

void transformFile(
    const std::filesystem::path& inputFilePath,
    const std::filesystem::path& outputFilePath,
    const RecordOperation& operation);
```

C++ does not normally list thrown exceptions in method signatures. Operations that do not throw according to their contract can be marked `noexcept`, as the destructors are here. Deterministic release is provided by RAII: a resource is tied to an object's lifetime and released by its destructor, which is invoked automatically when a scope is left or an exception unwinds the stack. Obtaining resources through a factory function or owning them through suitable smart pointers is also acceptable.

2. Implementations

Java

```
static void transformFile(
        String inputFilePath,
        String outputFilePath,
        RecordOperation operation) throws Exception {

    try (RecordReader reader = new RecordReader(inputFilePath);
         TransactionalRecordWriter writer =
                 new TransactionalRecordWriter(outputFilePath)) {

        Record record;
        while ((record = reader.read()) != null) {
            writer.write(operation.apply(record));
        }

        writer.commit();
    }
}
```

If creating the writer fails, try-with-resources closes the already created `reader`. If any later operation fails, the uncommitted `writer` is closed first, which discards the pending data, and then `reader` is closed.

C#

```
static void TransformFile(
    string inputFilePath,
    string outputFilePath,
    Func<Record, Record> operation)
{
    using (RecordReader reader = new(inputFilePath))
    using (TransactionalRecordWriter writer = new(outputFilePath))
```

```
    {
        Record? record;

        while ((record = reader.Read()) is not null)
        {
            writer.Write(operation(record));
        }

        writer.Commit();
    }
}
```

The nested `using` statements ensure that an already created `reader` is released if the writer constructor fails. If a later operation throws, the uncommitted `writer` is released first, which discards the pending data, and then `reader` is released.

C++

```
void transformFile(
    const std::filesystem::path& inputFilePath,
    const std::filesystem::path& outputFilePath,
    const RecordOperation& operation)
{
    RecordReader reader(inputFilePath);
    TransactionalRecordWriter writer(outputFilePath);

    while (std::optional<Record> record = reader.read()) {
        writer.write(operation(*record));
    }

    writer.commit();
}
```

Local objects are destroyed in the reverse order of their creation, so `writer` is destroyed before `reader`. If the writer constructor throws, the already created `reader` is destroyed automatically. The same destruction occurs if reading, transformation, writing, or committing throws.

In all the solutions shown, `commit` is called explicitly after the loop, which is reached only after all records have been successfully read, transformed, and written. If any of these operations fails, only automatic release of the uncommitted writer takes place and the original target file remains unchanged. Try-with-resources, `using`, and RAII do not call `commit` themselves.

3 Sequences and series (shared topics)

Let α be a real number. Let us define the sequence $(p_n)_{n=1}^{\infty}$ by

$$p_n = \frac{2n + \sin(\alpha)}{n^{\alpha}}.$$

1. Determine the limit of the sequence $(p_n)_{n=1}^{\infty}$, as a function of the parameter α.
2. State formally the definition of the sum of an infinite series.
3. For which values of α does the series $\sum_{n=1}^{\infty} p_n$ converge to a finite sum?

Solution sketch

1. From the simple bounds $n \leq 2n + \sin(\alpha) \leq 3n$ we obtain $n^{1-\alpha} < p_n < 3n^{1-\alpha}$, which implies that for $\alpha > 1$ we have $\lim_{n\to\infty} p_n = 0$ and for $\alpha < 1$ we get $\lim_{n\to\infty} p_n = +\infty$. For $\alpha = 1$, direct calculation yields $\lim_{n\to\infty} p_n = 2$.

2. For a sequence of real numbers $(a_n)_{n=1}^{\infty}$, the sum of the series $\sum_{n=1}^{\infty} a_n$ is defined as the limit of the sequence of partial sums $(s_n)_{n=1}^{\infty}$, where $s_n = a_1 + a_2 + \cdots + a_n$.
3. We know (or we easily deduce from the integral criterion), that $\sum_{n=1}^{\infty} \frac{1}{n^{\alpha-1}}$ converges if and only if $\alpha > 2$. Using again the bounds $n \leq 2n + \sin(\alpha) \leq 3n$, we see that $\sum_{n=1}^{\infty} p_n$ converges if and only if $\sum_{n=1}^{\infty} \frac{1}{n^{\alpha-1}}$ converges, i.e. if and only if $\alpha > 2$.

4 Trees (shared topics)

Let T be a tree with at least two vertices, and let L denote the number of its leaves.

1. State the number of edges of a tree on n vertices.
2. Prove that
$$L = 2 + \sum_{\substack{v \in V(T) \\ \deg(v) \geq 3}} (\deg(v) - 2).$$
3. Suppose that T has exactly 7 leaves, exactly one vertex of degree 4, and every other vertex has degree 2 or 3. How many vertices of degree 3 does T have?

Solution sketch

1. A tree on n vertices has exactly $n - 1$ edges.
2. Let $V_1 = \{v \in V(T) : \deg(v) = 1\}$ be the set of leaves. Thus $|V_1| = L$.

   By the handshaking lemma,
   $$\sum_{v \in V(T)} \deg(v) = 2|E(T)|.$$
   Since T is a tree, $|E(T)| = |V(T)| - 1$, and therefore
   $$\sum_{v \in V(T)} \deg(v) = 2|V(T)| - 2.$$
   It gives
   $$\sum_{v \in V(T)} (\deg(v) - 2) = -2.$$
   We now split the sum according to the degree of a vertex. A leaf contributes $1-2 = -1$, a vertex of degree 2 contributes 0, and a vertex of degree at least 3 contributes $\deg(v) - 2$. Hence
   $$-L + \sum_{\substack{v \in V(T) \\ \deg(v) \geq 3}} (\deg(v) - 2) = -2.$$
   Rearranging,
   $$L = 2 + \sum_{\substack{v \in V(T) \\ \deg(v) \geq 3}} (\deg(v) - 2).$$
3. Let x be the number of vertices of degree 3. By the formula from the previous part,
   $$7 = 2 + (4 - 2) + x(3 - 2).$$
   Therefore $7 = 2 + 2 + x$, so $x = 3$.

5 Multi-model Databases (specialization WDOP)

Consider an application that stores information about customers, orders, products, and relationships between customers. In a multi-model database, the data is stored as follows:

- relational tables

  `Customer(id, name)`, `Order(id, customer_id)`, `OrderItem(order_id, product_id)`;

- products as JSON documents with the attributes `id`, `name`, `category`, and additional properties that may differ between products;
- customers as graph vertices connected by directed `FOLLOWS` edges. The vertices use the same customer identifiers as the `Customer` table.

1. Explain what a multi-model database is. Propose how all the given data could be stored in a purely relational database.
2. Compare the multi-model and relational solutions in terms of the suitability of the data representation, data integrity, querying, and query optimization.
3. Describe how the following query would be evaluated in both solutions:

   > Find the names of customers who are followed by the customer Alice and who have ordered at least one product from the "database" category.

   For the multi-model solution, specify which partial operations would be performed over the individual data models and how their results would be combined.

Solution sketch

1. A multi-model database supports multiple data models and queries over them within a single system. The relational solution could, for example, be extended with the tables

   `Product(id, name, category)`,

   `ProductProperty(product_id, name, value)`

   and

   `Follows(follower_id, followed_id)`.

   The identifiers in the last two tables are covered by the appropriate primary and foreign key constraints.
2. The multi-model solution represents document and graph data more naturally. The relational solution provides a uniform data model, integrity constraints, and joint optimization of the entire query, but variable product properties and more complex graph queries are more difficult to represent.
3. In the graph, find the customers followed by Alice; in the documents, find the products in the „database“ category; and in the relational tables, find the customers who ordered at least one of these products. Combine the results using customer and product identifiers. In the relational database, the same query is evaluated by joining the tables `Customer`, `Follows`, `Order`, `OrderItem`, and `Product`.

6 File Organization and Indexing (specialization WDOP)

Consider a database table containing a large number of records stored in external memory.

1. Explain the differences between a heap file, a sequential file, and an indexed sequential file. Compare them in terms of record lookup, record insertion, and range queries.
2. Explain the differences between primary and secondary indexes and between direct and indirect indexing. Give an example of a suitable use case for each type.
3. Consider the table

   `Student(id, surname, year, averageGrade)`.

   Propose a suitable file organization and indexes for the following operations:

- finding a student by `id`,
- listing students ordered by surname,
- finding all students in a particular year of study,
- frequent insertion of new students.

Justify your choices and discuss possible trade-offs between these requirements.

Solution sketch

1. A heap file has no defined record order and supports fast insertion, but searching generally requires scanning the file. A sequential file is ordered by a key and supports efficient sequential processing and range queries, but insertions are more difficult. An indexed sequential file supplements ordered data with an index and supports both direct and sequential access at the cost of maintaining the index.
2. A primary index corresponds to the physical ordering of the file, whereas a secondary index is built over a different attribute. A direct index contains references to individual records, whereas an indirect index may refer, for example, to data blocks or another index level.
3. A primary index is suitable for lookup by `id`. Secondary indexes can be created for `surname` and `year`; an index on `surname` can also support the required ordering. An unordered file organization or a dynamic index structure is suitable when insertions are frequent. Additional indexes accelerate queries but slow down insertions and require additional storage.

7 Railway Service Database (specialization WDOP)

A railway operator uses a simple relational database to keep track of its train services. The schema includes the following tables:

```
STATION(id, name:CHAR, region:CHAR, ...)
SERVICE(id, number:CHAR, category:CHAR, name:CHAR, ...)
TRIP(id, service_id, departure_date:DATE, capacity:INT)
```

Columns named `id` are primary keys; foreign keys always have the form `table-name_id`. Data types are indicated after the column names. The data in the individual tables may look, for example, as follows:

```
STATION
id    name                region
1     Praha hl.n.         Hlavní město Praha
2     Pardubice hl.n.     Pardubický kraj

SERVICE
id    number    category    name
1     273       EC          Metropolitan
2     275       EC          Metropolitan

TRIP
id    service_id    departure_date    capacity
1     1             2026-10-01        420
2     2             2026-10-01        365
```

A record in the `SERVICE` table represents a scheduled train service (e.g., *EC 273 Metropolitan* or *Os 9640*), while a record in the `TRIP` table represents a particular run of that service on a given day. For simplicity, you may assume that each service has a fixed route and fixed arrival and departure times throughout the validity period of the timetable; i.e., you do not need to consider temporary closures, diversions, or other exceptional changes.

Questions:

1. The database schema shown above is incomplete. Extend the schema with the missing table(s), including a description of their columns and identification of primary and foreign keys, so that the timetable of individual trips can be represented. In particular, it must be possible to determine which stations a train passes through, in what order, and its arrival and departure times at these stations.

2. Write an SQL `SELECT` query that lists all unique services that include the station "Pardubice hl.n." on their route.
3. Write an SQL `SELECT` query that for each day returns a total number of passengers that can travel on direct trains from "Petrov u Prahy" to "Krhanice". Notes: As "day" we understand the date of train's departure; as "direct train" we understand that both stations have to be on the train's route, in the correct order.

Solution sketch **Question 1:** To represent the timetable, a relation between individual trips and stations needs to be added, for example:

```
TRIP_STOP(trip_id, station_id, stop_order:INT,
          arrival_time:TIME, departure_time:TIME)
```

The primary key may, for example, consist of `(trip_id, stop_order)` (`(trip_id, station_id)` is possible, but prevents circular lines). Both `trip_id` and `station_id` are foreign keys.

The simplification stated in the description also makes it possible to store the timetable at the service level instead:

```
SERVICE_STOP(service_id, station_id, stop_order:INT,
             arrival_time:TIME, departure_time:TIME)
```

Question 2:

```
SELECT DISTINCT s.id, s.number, s.category, s.name
FROM service s
JOIN trip t ON t.service_id = s.id
JOIN trip_stop ts ON ts.trip_id = t.id
JOIN station st ON st.id = ts.station_id
WHERE st.name = 'Pardubice hl.n.'
```

Question 3:

```
SELECT t.departure_date, SUM(t.capacity) AS total_capacity
FROM trip t
JOIN trip_stop ts_from ON ts_from.trip_id = t.id
JOIN station st_from ON st_from.id = ts_from.station_id
JOIN trip_stop ts_to ON ts_to.trip_id = t.id
JOIN station st_to ON st_to.id = ts_to.station_id
WHERE st_from.name = 'Petrov u Prahy'
  AND st_to.name = 'Krhanice'
  AND ts_from.stop_order < ts_to.stop_order
GROUP BY t.departure_date
```

8 Recommender Systems (specialization WDOP)

1. Explain the basic working principles of item-based k-nearest neighbours (Item-KNN) for collaborative filtering.

   Next, consider the following matrix of binary implicit feedback (1 indicates an interaction was observed between the user and the item, while 0 otherwise). Pick a suitable similarity metric for this case, provide its definition, and answer which item is the most similar to `I1` according to this metric.

	I1	I2	I3	I4	I5
U1	1	1	1	0	0
U2	1	1	0	1	0
U3	1	0	1	0	1
U4	0	1	1	1	0
U5	0	0	1	0	1

2. Can a standard Item-KNN model recommend newly added items that have not yet received any user interactions (i.e., solve the new item problem)? Explain how or why not. If not, describe one way in which the method could be modified or extended to enable recommendations of such newly emerging items.
3. Recommendation accuracy is an important objective, but it does not provide a complete picture of the quality of a recommender system. Describe at least two beyond-accuracy objectives that may be considered when evaluating or

designing recommender systems. For each objective, briefly explain its meaning and sketch a metric that can be used to measure the fulfillment of such an objective.

Solution sketch **Question 1:** Item-KNN represents each item through the users who interacted with it (or their ratings) and computes pairwise similarities between items. Depending on the underlying data, Pearson's correlations, cosine similarity, or Jaccard similarity are frequently used in this step.

For a target user, candidate items are scored based on their similarity to items with which the user has previously interacted, typically using a sum or weighted sum of these similarities. The highest-scoring unseen items are then recommended.

While other metrics are plausible, Jaccard similarity is a particularly reasonable choice for the binary feedback scenario. For two items i and j, let U_i and U_j denote the sets of users who interacted with them. Their Jaccard similarity is

$$J(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}.$$

For `I1`, the similarities are

$$J(I1,I2) = 2/4 = 0.5, \quad J(I1,I3) = 2/5 = 0.4,$$
$$J(I1,I4) = 1/4 = 0.25, \quad J(I1,I5) = 1/4 = 0.25.$$

Thus, `I2` is the most similar item to `I1` according to Jaccard similarity.

Question 2: A newly added item without interactions cannot normally be recommended by a purely collaborative Item-KNN model, because its similarity to the existing items cannot be estimated from interaction data. The simplest solution is a content-based modification of the algorithm, in which similarities are computed from item metadata or textual descriptions instead of interactions.

Question 3: There are many beyond-accuracy objectives and metrics. The most common examples include:

- *Diversity*: recommendations should contain mutually different items; e.g., *intra-list diversity*.
- *Novelty*: recommendations should contain items that are less likely to be already known to the user; e.g., *inverse popularity*, *freshness*.
- *Coverage*: the recommender should utilize a sufficiently large portion of the available catalog; e.g., *catalog coverage*.
- *Exploration*: recommendations should include items that expand beyond the established user profile; e.g., maximal distance from the profile, fraction of items from unseen categories.
- *Serendipity*: recommendations should be pleasantly surprising, i.e., relevant despite not being expected; *estimated relevance* $\times$ *unexpectedness*.
- *Fairness*: recommendation outcomes or exposure should not systematically disadvantage particular users, items, or providers; e.g., *exposure disparity*.

9 Combinatorics and estimates (specialization OI-G-O, OI-G-PADS, OI-G-PDM, OI-O-PADS, OI-O-PDM, OI-PADS-PDM)

Let us consider the following games of chance:

(a) An opaque bag contains n black and n white balls, which we take out one by one in random order. We keep track of how many of each type of ball we have already taken out, and if at any point the number of black balls strictly exceeds the number of white balls, we lose (for example, if the first ball drawn is black, we lose immediately, and we also lose if it is white and the next two are black; for any other combination of the first three balls, the game continues). If we manage to take out all the balls without this situation occurring, we win.

(b) We flip a coin $2n$ times in a row. We win if we get heads and tails an equal number of times (that is, exactly n times each).

Determine exact formulas for the probabilities of winning in games (a) and (b). In which of these games do we have (asymptotically for large n) a higher probability of winning? You may use the formulas for standard quantities and estimates

(explicitly mentioned in the SFE Requirements) without proof.

Solution sketch For game (a), let us imagine that even if we lose, we continue taking out all the remaining balls. Let us write an opening parenthesis for each white ball we take out and a closing parenthesis for each black one. In this game, we win if and only if the resulting string is well matched. The number of all well matched strings consisting of n opening and n closing parentheses is given by Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$. The number of all possible (equally likely) ways to arrange n black and n white balls (which are indistinguishable from one another) is $\binom{2n}{n}$ – it suffices to determine the positions of the black balls. The probability that we win thus is

$$\frac{C_n}{\binom{2n}{n}} = \frac{1}{n+1}.$$

For game (b), we can win in $\binom{2n}{n}$ ways (the number of choices for the positions of the n "head" flips out of $2n$ positions in total). The total number of (equally likely) outcomes of $2n$ flips is 2^{2n}. The probability that we win is

$$\frac{\binom{2n}{n}}{2^{2n}} = \frac{1}{\Theta(\sqrt{n})},$$

using the well-known estimates of binomial coefficients.

Therefore, we are asymptotically $\Theta(\sqrt{n})$-times more likely to win in the game (b).

10 Perfect graphs (specialization OI-G-PDM, OI-O-PDM, OI-PADS-PDM)

- Write the definition of a perfect graph, and state the strong and the weak perfect graph theorems.
- Which of the following claims are true? Your justification can use the weak perfect graph theorem, but not the strong one:
  1. If a graph G has minimum degree $|V(G)| - 2$, then G is perfect.
  2. Let H be the graph whose vertices are the fields of the usual 8×8 chessboard and two vertices are joined by an edge if the corresponding fields are directly adjacent horizontally, vertically or diagonally (that is, a king on one of the fields can move to the other one). The graph H is perfect.

Solution sketch

- A graph G is perfect when every induced subgraph G' of G satisfies $\chi(G') = \omega(G')$, that is, its chromatic number is equal to its clique number.

  Strong perfect graph theorem: A graph G is perfect if and only if neither G nor its complement $\overline{G}$ contains any induced cycle of length at least five.

  Weak perfect graph theorem: A graph G is perfect if and only if its complement $\overline{G}$ is perfect.
- The first claim is true, the second one is false.
  1. If a graph G has minimum degree $|V(G)| - 2$, then its complement $\overline{G}$ has maximum degree one, that is, $\overline{G}$ is a partial matching. Clearly the graph $\overline{G}$ is perfect (its induced subgraphs are partial matchings as well, and they either consist only if isolated vertices and have both clique number and chromatic number one, or have at least one edge and both clique number and chromatic number two). The weak perfect graph theorem thus implies that the graph G is perfect as well.

     It is of course also possible to prove this directly, by arguing about the clique number and the chromatic number of all induced subgraphs of G.
  2. Let v_{xy} denote the vertex of H corresponding to the field in the x-th column and the y-th row. Then $v_{13}v_{12}v_{21}v_{32}v_{43}v_{34}v_{24}$ is an induced 7-cycle in H, whose chromatic number is 3 and clique number is 2. Hence, the graph H is not perfect (even though both its clique number and its chromatic number are equal to four).

11 Network flows (specialization OI-G-PADS, OI-O-PADS, OI-PADS-PDM)

1. Formulate basic ideas of Dinitz's algorithm for maximum flow in a network. (Detailed pseudo-code is not required.)
2. How many phases (iterations of the outer loop) can the algorithm perform? What is the time complexity of a single phase? What does it imply for total time complexity of the whole algorithm? All bounds should be expressed as functions of the number of vertices n and the number of edges m of the network given; you may assume that $n \in \mathcal{O}(m)$.
3. Consider a network in the shape of a cyclic $t \times t$ grid: the vertex set is $\{0, \ldots, t-1\} \times \{0, \ldots, t-1\}$, directed edges from a vertex (x, y) lead to vertices $((x+1) \bmod t, y)$ and $(x, (y+1) \bmod t)$, the source is the vertex $(0, 0)$, the sink is $(t-1, t-1)$, all edge capacities belong to the set $\{0, \ldots, 99\}$. Formulate a tight upper bound on time complexity of the Dinitz's algorithm for networks of this type. Express it as a function of the parameter t. Explain why it holds.

Solution sketch

1. See the Guide to the Labyrinth of Algorithms, section 14.4.
2. The number of phases never exceeds n (each phase increases the length of the shortest non-saturated path from the source to the sink). A phase consists of construction of the residual network (this needs $\mathcal{O}(m)$ time), cleaning of the network ($\mathcal{O}(m)$) and computation of a blocking flow — every such computation finds at most m augmenting paths, each processed in $\mathcal{O}(n)$ time. The total time per phase is therefore in $\mathcal{O}(nm)$ and the whole algorithm runs in $\mathcal{O}(n^2 m)$ time.
3. The network consists of $n = t^2$ vertices and $m = 2t^2$ edges. Outgoing edges of the source form a cut of capacity smaller than 200, which also bounds the size of every flow. Since capacities are integers, each phase increases flow size by at least 1, so the total number of phases cannot exceed 200. One phase has an overhead of $\mathcal{O}(m)$ and then it spends $\mathcal{O}(n)$ time per augmenting path; however, the total number of such paths over the whole computation is also at most 200. Therefore the total time complexity is $\mathcal{O}(200 \cdot (n+m)) = \mathcal{O}(t^2)$. This bound cannot be improved, since the overhead of the first phase itself is $\Omega(t^2)$.

12 Extrema of functions (specialization OI-G-O, OI-G-PADS, OI-G-PDM, OI-O-PADS, OI-O-PDM, OI-PADS-PDM)

Let A and B be fixed real numbers. Consider the function $f(x)\colon \mathbb{R} \to \mathbb{R}$ defined as

$$f(x) = (x - A)^2 + Bx.$$

1. What is the range of the function f?
2. Assume that A and B are both nonnegative. Consider the planar region R bounded from the left by the line $x = 0$, from the right by the line $x = 1$, from below by the x-axis, and from above by the graph of f. Formally,

   $$R = \{(x, y) \in \mathbb{R} \times \mathbb{R};\ 0 \le x \le 1 \wedge 0 \le y \le f(x)\}.$$

   Show that the area of R is equal to $A^2 - A + \frac{B}{2} + \frac{1}{3}$ square units.
3. Let us constrain the values of A and B by $0 \le A \le B$. For which values of A and B satisfying this constraint is the area of R minimized?

Solution sketch

1. The derivative of f equals $2(x - A) + B$. It is negative for $x \in (-\infty, A - B/2)$, equals zero for $x = A - B/2$ and is positive otherwise. Thus, $x = A - B/2$ is the unique point where f attains its minimum, with $f(A - B/2) = AB - B^2/4$. The function f has the limit $+\infty$ for $x \to +\infty$ as well as for $x \to -\infty$ and is clearly continuous, hence its range is $[AB - B^2/4, +\infty)$. One may also solve this exercise by transforming f as $f(x) = (x - (A - B/2))^2 + AB - B^2/4$.
2. The area of R equals $\int_0^1 f(x)\mathrm{d}x$. The function f admits the antiderivative $F(x) = (x - A)^3/3 + Bx^2/2 + c$. By simplifying the expression $F(1) - F(0)$ we get the desired result.
3. Let us denote the area calculated in the previous part as $\Phi(A, B)$. The function Φ is increasing in B. For any $A < B$ we thus get $\Phi(A, A) < \Phi(A, B)$. Thus, the minimum of Φ subject to the constraint $0 \le A \le B$ can only be attained

for $A = B$. Differentiating the function $\Phi(A, A) = A^2 - A/2 + 1/3$ shows that it has a unique minimum in $A = 1/4$. $\Phi(A, B)$ is thus minimized in $(A, B) = (1/4, 1/4)$. One may also solve this exercise by using Lagrange multipliers.

13 Planning (specialization UI-SU, UI-ZPJ, UI-ROB)

1. Formally define the planning problem. What is the classical representation of state under the closed world assumption?
2. What is the difference between a goal state and a goal condition? What is the difference between a fluent and a static (rigid) atom?
3. Consider a world where there are boxes that can lie on top of each other or on the ground. Only one box can fit on top of a box, but we can build stacks of unlimited height, and there is unlimited space on the ground. We have one crane available, which can lift a box that has no other box lying on it, and move it onto the ground or onto another box that has nothing on it. Write this problem as a planning domain. (Hint: we are only interested in the domain, not a problem instance.)

Solution sketch

1. A planning problem consists of a set of states, a set of actions, and a transition function ($state \times action \to state$). A problem instance additionally includes an initial state and a goal condition. A state is represented by a set of valid atoms (predicates). Under the closed world assumption, it is assumed that everything not listed as a true atom is false (i.e., it suffices to list only the true atoms in a state).
2. A goal condition is a set of atoms that must hold in every goal state. A goal state can contain additional true atoms, there can be multiple goal states. A static atom is an atom whose truth value never changes, whereas a fluent can change its truth value between states.
3. The problem can be modeled using the following predicates and actions:

   Predicates: OnBox(box1, box2), OnGround(box1), EmptyCrane(), Carrying(box1), EmptyAbove(box1)

   Actions:

   PickFromTable box1

   preconditions: OnGround(box1), EmptyCrane(), EmptyAbove(box1)

   effects: not OnGround(box1), not EmptyCrane(), not EmptyAbove(box1), Carrying(box1)

   PickFromStack box1, box2

   preconditions: OnBox(box1, box2), EmptyCrane(), EmptyAbove(box1)

   effects: not OnBox(box1, box2), not EmptyCrane(), not EmptyAbove(box1), Carrying(box1), EmptyAbove(box2)

   PutOnTable box1

   preconditions: Carrying(box1)

   effects: not Carrying(box1), EmptyCrane(), OnGround(box1), EmptyAbove(box1)

   PutOnStack box1 box2

   preconditions: Carrying(box1), EmptyAbove(box2)

   effects: not Carrying(box1), not EmptyAbove(box2), EmptyCrane(), OnBox(box1, box2), EmptyAbove(box1)

   Since there is only one crane, it would alternatively be possible to model the move actions directly, but then it would be necessary to distinguish between the combinations of moves from a stack/ground to a stack/ground.

14 Evaluation of binary classifier (specialization UI-SU, UI-ZPJ)

A hospital considers using one of two machine learning models (model A and model B) for diagnosis of a rare disease. These models would use already existing data about patients and no new tests would be required. As usual in this field, the models were evaluated using their *sensitivity* and *specificity*. Sensitivity is the proportion of true positives (correctly positively

diagnosed patients) among all patients with the disease (i.e. the recall of the model), while specificity is the proportion of true negatives (correctly diagnosed healthy patients) among all healthy patients. Assume that 2% of people have the disease (i.e. the prevalence is 2%) and that the hospital tested the models on 5,000 patients. Assume, that model A has sensitivity 0.95 and specificity 0.9, while model B has sensitivity 0.8 and specificity 0.99.

1. Calculate the full confusion matrix for both models. Calculate the accuracy, precision, recall, and F1 score for both models.
2. Consider the following two scenarios:
   A. After using the model, the hospital can perform an additional test on all patients who were diagnosed as positive by the model. This additional test is relatively cheap and gives definitive answer, if the patient has the disease or not. Only patients with the disease will be treated.
   B. There is no additional test available. All patients diagnosed as positive by the model will be treated for the disease. The treatment is expensive and has serious side effects.

   Which model would you recommend for each of the scenarios? Justify your answer.

Solution sketch

1. – Model A:
   - True Positives (TP): 0.02 * 5000 * 0.95 = 95
   - False Negatives (FN): 0.02 * 5000 * (1 - 0.95) = 5
   - True Negatives (TN): 0.98 * 5000 * 0.9 = 4410
   - False Positives (FP): 0.98 * 5000 * (1 - 0.9) = 490

   Confusion Matrix:
   $$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} = \begin{bmatrix} 95 & 490 \\ 5 & 4410 \end{bmatrix}$$
   Accuracy: (TP + TN) / Total = (95 + 4410) / 5000 = 0.90 Precision: TP / (TP + FP) = 95 / (95 + 490) = 0.16 Recall: TP / (TP + FN) = 95 / (95 + 5) = 0.95 F1 Score: 2 * (Precision * Recall) / (Precision + Recall) = 0.28

   – Model B:
   - True Positives (TP): 0.02 * 5000 * 0.8 = 80
   - False Negatives (FN): 0.02 * 5000 * (1 - 0.8) = 20
   - True Negatives (TN): 0.98 * 5000 * 0.99 = 4851
   - False Positives (FP): 0.98 * 5000 * (1 - 0.99) = 49

   Confusion Matrix:
   $$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} = \begin{bmatrix} 80 & 49 \\ 20 & 4851 \end{bmatrix}$$
   Accuracy: (TP + TN) / Total = (80 + 4851) / 5000 = 0.99 Precision: TP / (TP + FP) = 80 / (80 + 49) = 0.62 Recall: TP / (TP + FN) = 80 / (80 + 20) = 0.80 F1 Score: 2 * (Precision * Recall) / (Precision + Recall) = 0.70

2. A. As an additional cheap test is available for all patients diagnosed as positive, it is more important to have a high sensitivity (recall) to ensure that most patients with the disease are correctly identified. Therefore, Model A would be recommended in this scenario as it misses only 5 patients with the disease, while Model B misses 20 patients.
   B. Since there is no additional test available and all patients diagnosed as positive will be treated, we need to minimize false positives to avoid unnecessary treatment and its associated costs and side effects. Therefore, Model B would be recommended in this scenario, as it has only 49 false positives compared to Model A's 490 false positives.

15 Model ensembling (specialization UI-SU, UI-ZPJ)

Ensembling is a technique in which we combine the outputs of several trained models (e.g. by averaging them) to obtain a better prediction than with a single model.

1. What is the asymptotic computational cost of inference with M models compared to inference with a single model? Justify your answer.
2. Assume M regression models trained to minimize mean squared error, where the error of the i-th model is ϵ_i, and the errors of the individual models are uncorrelated. What is the expected error of the ensemble (the average of the predictions), expressed in terms of M and ϵ_i? What is the consequence of this result – to what extent does it pay off to keep adding more models to the ensemble?
3. The assumption of uncorrelated errors from the previous point is hard to satisfy in practice. Assume you train the models using stochastic gradient descent. Name at least three concrete techniques you could use to make the errors of the individual models less correlated.

Solution sketch

1. Each model has to be run independently of the others, so the total cost grows linearly with the number of models. (Preprocessing of the input can potentially be shared across models if it is identical, but the prediction computation of each model itself must be carried out separately.)
2. From the definition of mean squared error $\mathbb{E}[\epsilon_i] = 0$. Therefore, if the errors ϵ_i and ϵ_j are uncorrelated, their covariance is zero and $\mathbb{E}[\epsilon_i \epsilon_j] = 0$ for $i \neq j$. The ensemble error is $\bar{\epsilon} = \frac{1}{M}\sum_{i=1}^{M} \epsilon_i$. The squared error is

$$\mathbb{E}[\bar{\epsilon}^2] = \frac{1}{M^2}\mathbb{E}\left[\left(\sum_{i=1}^{M} \epsilon_i\right)^2\right] = \frac{1}{M^2}\sum_{i=1}^{M} \mathbb{E}[\epsilon_i^2] = \frac{1}{M}\sum_{i=1}^{M} \frac{1}{M}\mathbb{E}[\epsilon_i^2],$$

   since the cross terms $\mathbb{E}[\epsilon_i \epsilon_j]$ for $i \neq j$ vanish due to the uncorrelatedness. The expected squared error of the ensemble is thus M times smaller than the average error of an individual model.

   Consequence: The error decreases with M only as $1/M$, i.e. with diminishing marginal returns – going from one model to two models reduces the error far more than going from nine models to ten. While the computational cost grows linearly with M (see part 1), the benefit of adding further models shrinks quickly, so beyond a certain point adding more models to the ensemble stops being worthwhile relative to the computational resources spent. Moreover, the error never reaches zero (only approaches it in the limit $M \to \infty$), and in practice errors are never fully uncorrelated, so the real improvement is even smaller than the formula suggests.
3. Models can be made less correlated by, for example:
   - training on different random subsets of the training data (bagging/bootstrap),
   - using different random weight initializations,
   - using a different random order of shuffling the training data across epochs,
   - using different hyperparameters or slightly different architectures,
   - restricting each model to a random subset of the input features,
   - using random data augmentation (different for each model).

   Each of these sources of randomness causes individual models to make different mistakes, bringing the setting closer to the assumption of uncorrelated errors.

16 Linear Regression and Lasso Regularization (specialization UI-SU)

Consider the training data with an independent variable x and a target variable y given in the table below. We will fit a linear model without regularization and with $L1$ regularization on this data.

x	y
-1	1
0	3
1	4

1. Describe the linear model and the least squares method. How are the model coefficients estimated?
2. Consider the model without regularization (OLS). Compute the model coefficients for the data in the table.
3. Consider Lasso regression ($L1$ regularization) with regularization parameter $\alpha = 100$. Describe the training objective of Lasso regression and determine the model coefficients for the given data.

Solution sketch

1. A linear model with a single independent variable estimates two parameters, β_0 and β_1:

$$f(x) = \beta_0 + \beta_1 x.$$

The least squares method minimizes the objective function

$$\sum_{i=1}^{3} (f(x_i) - y_i)^2.$$

To estimate the parameters, we add a column of ones to the data matrix and solve

$$\beta = (X^T X)^{-1} X^T y.$$

2.

$$\begin{aligned}
\beta &= (X^T X)^{-1} X^T y \\
&= \left(\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix} \\
&= \left(\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} 8 \\ 3 \end{bmatrix} = \begin{bmatrix} 8/3 \\ 3/2 \end{bmatrix}.
\end{aligned}$$

The model parameters are $\beta_0 = \frac{8}{3}$, $\beta_1 = \frac{3}{2}$, the resulting model is $f(x) = \frac{8}{3} + \frac{3}{2}x$.

Predictions on the training data are

$$f(-1) = \frac{7}{6}, \qquad f(0) = \frac{8}{3}, \qquad f(1) = \frac{25}{6}.$$

The sum of squared errors is

$$\left(1 - \frac{7}{6}\right)^2 + \left(3 - \frac{8}{3}\right)^2 + \left(4 - \frac{25}{6}\right)^2 = \frac{1}{36} + \frac{1}{9} + \frac{1}{36} = \frac{1}{6}.$$

Thus, the training error is

$$\frac{1}{6}.$$

3. Lasso regression minimizes

$$\sum_{i=1}^{3} (f(x_i) - y_i)^2 + \alpha |\beta_1|.$$

Because the regularization strength is very large, $L1$ regularization can force the coefficient exactly to zero $\beta_1 = 0$. (For these data, the threshold at which the coefficient becomes zero is $\alpha = 6$.)

The intercept is equal to the mean of the target values:

$$\beta_0 = \bar{y} = \frac{1+3+4}{3} = \frac{8}{3}.$$

Therefore, the resulting model is

$$f(x) = \frac{8}{3}.$$

17 Representing text for a neural network (specialization UI-ZPJ)

Neural networks operate on numerical inputs, whereas digital text is a sequence of discrete symbols. In this question, we will discuss how to convert a raw piece of text into a form that can be processed by a neural network.

1. Describe how a raw piece of text can be converted into a form that a neural network can process. In particular, explain what determines the "units" the text is broken into and why a fixed-size set of such units is needed. How are the units represented numerically, and what is the final form of the input to the network?
2. Consider a simple algorithm that splits text into units strictly on whitespace and nothing else. What problems would such an algorithm have? How are these problems typically addressed in practice? (Assume that the language separates words with whitespace, as English does.)
3. Suppose you have trained a neural network processing text. One way to check if the input representation captures something meaningful is to pick a unit and look at its nearest neighbors in the vector space. Explain what that means algorithmically: How would you find the nearest neighbors of a given unit? What would you expect these neighbors to look like if the representations were trained well?

Solution sketch

1. The text is first split into a sequence of discrete units drawn from a fixed, finite vocabulary (e.g. words). This is necessary because the network needs a bounded lookup table of representations to map units to vectors. An unbounded vocabulary would make this table infinite (you can invent new words, there may be possible typos), and most units would occur too rarely to learn anything useful about them. Each unit in the vocabulary is mapped to a vector of real numbers (e.g. via a lookup table indexed by the unit's ID). The input to the network is a sequence of these vectors. Alternatively, we can imagine that each unit is represented as a one-hot vector, which is then mapped to a dense vector via a learned embedding layer.
2. Splitting text strictly on whitespace has two main kinds of problems. First, a vocabulary made of whole words is very large – especially for morphologically rich languages (many word forms per lemma) or multilingual models – and yet test data will inevitably contain words that were never seen when the vocabulary was built (out-of-vocabulary, OOV), especially if the test data differs substantially from the training data; such units cannot be represented at all. Second, whitespace does not separate a word from punctuation attached to it, so "dog,", "dog", and "dog." would be treated as three different units, needlessly inflating the vocabulary with near-duplicate entries and making it harder for the model to recognize that these all refer to the same underlying word. In practice, the punctuation problem is addressed by explicitly separating it off using rules or regular expressions. The vocabulary-size/OOV problem is addressed with subword tokenization (byte-pair encoding, WordPiece, SentencePiece/Unigram), which splits rare or unknown words into smaller, frequently occurring substrings – this keeps the vocabulary finite and bounded, and, in the extreme, any word (even unseen ones) can still be encoded by falling back to individual characters or bytes.
3. Algorithmically, this means iterating over all units in the vocabulary, computing the similarity (e.g. cosine similarity) between the embedding of each of them and the embedding of the chosen unit, and picking the K units with the highest similarity (smallest distance). If the representations have been trained well, the nearest neighbors found this way should be units that are semantically or syntactically related to it, e.g. synonyms, words from the same category (days of the week, colors, etc.), or words that tend to appear in similar contexts. If, instead, the nearest neighbors look

essentially unrelated or random, this suggests the representations have not converged to anything meaningful, e.g. due to insufficient training data, too few training steps, or a bug in the training procedure.