State Final Examination (Sample Questions)

2025-06-23

1 Deterministic finite automata (shared topics)

1. Write a definition of the language $L(A)$ accepted by a given deterministic finite automaton $A = (Q, \Sigma, \delta, q_0, F)$ including a definition of its extended transition function δ^*.
2. Let $S \subseteq \Sigma^*$ be a nonempty finite set of words, Q be the set of all prefixes of words in S (including the empty word ε and the words from S), and $A = (Q, \Sigma, \delta, \varepsilon, S)$ be a deterministic finite automaton with the state set Q. Define its transition function δ so that it accepts the language

$$L(A) = \{w \in \Sigma^* \mid w \text{ contains some } s \in S \text{ (as a substring)}\}.$$

3. Construct a deterministic finite automaton accepting the language

$$\{w \in \{0,1\}^* \mid w \text{ contains 010 or ends with 10}\}.$$

Solution sketch

1. $L(A) := \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$ where $\delta^* \colon Q \times \Sigma^* \to Q$ is defined inductively by

$$\begin{aligned} \delta^*(q, \varepsilon) &:= q, \\ \delta^*(q, wx) &:= \delta(\delta^*(q, w), x) \end{aligned}$$

   for every $q \in Q$, $w \in \Sigma^*$ and $x \in \Sigma$.
2. For every $w \in Q$ and $x \in \Sigma$,

$$\delta(w, x) := \begin{cases} w, & \text{if } w \in S, \\ s, & \text{if } w \notin S \text{ and } wx \text{ ends with some } s \in S \text{ (take e.g. the shortest)}, \\ v, & \text{where } v \text{ is the longest suffix of the word } wx \text{ that is in } Q, \text{ otherwise.} \end{cases}$$

3. The automaton $A = (\{\varepsilon, 0, 1, 01, 10, 010\}, \{0,1\}, \delta, \varepsilon, \{10, 010\})$ is on the figure above.

2 Simple memory manager (shared topics)

Assume throughout the entire question a 32-bit ***little endian*** processor, ***bit 0 is understood to be the LSb.***

Our goal is to implement a simple heap for dynamic allocation of sub-blocks of memory within a continuous block of memory (not to be confused with the heap data structure). The entire heap will have a fixed size given during initialization, and will not use any optimizations beyond the function described here. We want to organize the memory of our heap as a continuous sequence of sub-blocks, where each sub-block must always be ***aligned to 2 bytes***. Each sub-block in the heap is either *free* or *allocated* (used).

Free sub-blocks have the following structure:
`+0`: `Size` [16-bit] = size of the sub-block ***including this header***. Previous text implies that bit 0 of this field would always be 0, thus we do not store this zero. Instead, bit 0 of the `Size` field has a special meaning — it indicates whether the sub-block is free or allocated. `0` = *free*, `1` = *allocated*. Thus, for a free sub-block:

`bit 0`: `0` = free
`+2`: `Next` [16-bit] = offset of the next free sub-block from the start of the entire heap — thus free sub-blocks form a singly linked list. The last free sub-block has the value `-1` in this field.
`+4`: the rest of the sub-block is unused and filled with zeros.

Allocated sub-blocks have the following structure:
`+0`: `Size` [16-bit] = has the same meaning as in a free sub-block, except:
`bit 0`: `1` = allocated
`+2`: `Payload` – the actual data of the sub-block (our allocator returns all these bytes zeroed out).
Note: Allocated sub-blocks do not have a `Next` field — the memory location that held the `Next` field in a free block becomes part of the `Payload` upon allocation, and must be zeroed out by the heap allocator.

For the whole heap, we remember the offset of the first free sub-block from the start of the heap. Initially, the heap consists of exactly 1 free sub-block, whose size corresponds to the entire heap size. When a request is made to allocate a new sub-block, we receive the requested payload size in bytes and find the first large-enough free sub-block with the lowest offset from the start of the heap (i.e., a first-fit strategy).

Assume the following class skeleton in C#, which you will complete (you may write the required implementation in C#, Java, or C++ — for Java or C++, assume an equivalent implementation of the skeleton in the given language, and you may suitably adjust constructor and method parameters according to the conventions of the given language) (`ushort` is a 16-bit unsigned integer):

```
class Heap {
        private ushort _firstFreeOffset = 0;
        private byte[] _heap;

        public Heap(byte[] availableMemory) {
                _heap = availableMemory;
                // TODO
        }
        private ushort FindFirstFree(ushort payloadSize) { // TODO }
        private void Mark(ushort offset, bool isFree) { // TODO }

        public void SetUshort(ushort offset, ushort value) { ... }
        public ushort GetUshort(ushort offset) { ... }
        public ushort Alloc(ushort payloadSize) { ... }
        public void Free(ushort payloadOffset) { ... }
}
```

You should **not** implement the `SetUshort` and `GetUshort` methods, but instead use them appropriately in your code. The `SetUshort` method stores the `value` at the given `offset` from the start of the heap in little endian order. The `GetUshort` method returns a 16-bit value stored in the heap at the given `offset` (interpreting bytes in little endian order).

You should also **not** implement the `Alloc` and `Free` methods — these methods will be implemented by someone else using your methods. For task 4 below, it is useful to know: the `Alloc` method allocates space in the heap for a `payloadSize`-sized payload and returns the offset of the payload; the `Free` method takes the offset of a payload returned by some call to `Alloc` and frees the sub-block to which the payload belongs.

Tasks:

1. Complete the constructor implementation to correctly initialize an empty heap.
2. Write the implementation of the `FindFirstFree` method, which returns the offset of the first free sub-block of a suitable size. This method does not modify the state of the sub-block.
3. Write the implementation of the `Mark` method, which assumes that there is a valid heap sub-block at the given `offset`, and changes its state to "allocated" or "free" based on the `isFree` parameter. The method will not modify the `Next` field.
4. Assume that we have been given 22 bytes of memory for our heap – this memory is initially filled with zero bytes. Assume we then perform the following operations on the heap:
   A = `Alloc(2)`
   B = `Alloc(4)`
   C = `Alloc(2)`
   `Free(B)`

Write a hexdump of the 22 bytes of memory in the final state of the heap after all these operations — fill your solution into one of the tables below (multiple copies are included redundantly in case you need to revise your answer).

offset	+0	+1	+2	+3	+4	+5	+6	+7
0x0000								
0x0008								
0x0010								

offset	+0	+1	+2	+3	+4	+5	+6	+7
0x0000								
0x0008								
0x0010								

Solution sketch

offset	+0	+1	+2	+3	+4	+5	+6	+7
0x0000	05	00	00	00	06	00	0E	00
0x0008	00	00	05	00	00	00	08	00
0x0010	FF	FF	00	00	00	00	–	–

3 Inner product (shared topics)

Consider a map of the form $\langle x, y\rangle = x^T A y$, where $A \in \mathbb{R}^{n\times n}$ is a symmetric matrix.

1. Decide whether the following statements are true:
   (a) If the map $\langle x, y\rangle = x^T A y$ is an inner product on $\mathbb{R}^n$, then $\det(A) > 0$.
   (b) If $B \in \mathbb{R}^{n\times n}$ is a symmetric nonsingular matrix, then the expression $\langle x, y\rangle = x^T B^2 y$ defines an inner product on $\mathbb{R}^n$.
2. Consider an inner product on $\mathbb{R}^n$ given by the expression $\langle x, y\rangle = x^T A y$ for a suitable matrix $A \in \mathbb{R}^{n\times n}$. For this inner product, express the Cauchy–Schwarz inequality using matrix and arithmetic operations instead of the norm and innner product.
3. Decide whether the following statement is true: For every natural number $n \geq 1$, the matrix

$$A = \begin{pmatrix} 1 & \dots & \dots & 1 \\ \vdots & 2 & \dots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \dots & n \end{pmatrix}$$

   (i.e., $A \in \mathbb{R}^{n\times n}$ has entries $a_{i,j} = \min(i, j)$) is positive definite. Formulate the relevant criterion and verify it, or show a counterexample.

Solution sketch

1. (a) Yes, the matrix A is positive definite. Therefore, it has positive eigenvalues and consequently also the determinant.
   (b) Yes, the matrix B^2 is symmetric and has positive eigenvalues. Therefore, it is positive definite.
2. Cauchy–Schwarz inequality: For every $x, y \in V$ we have

$$|\langle x, y\rangle| \leq \|x\| \cdot \|y\|.$$

   In our case the inequality takes the form

$$|x^T A y| \leq \sqrt{x^T A x}\sqrt{y^T A y}.$$

3. Yes, matrix A is positive definite. Its Cholesky decomposition reads as $A = LL^T$, where

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 1 & \dots & \dots & 1 \end{pmatrix}.$$

   Positive definiteness can also be verified by Gaussian elimination or by Sylvester's criterion.

4 Central Limit Theorem (shared topics)

1. State the Central Limit Theorem.
2. Assume that a program processes one input in a random amount of time with a mean of 100 ms and a standard deviation of 20 ms. Use the Central Limit Theorem to estimate the probability that processing one hundred inputs will take more than 10.3 seconds. Write the exact formula using Φ and also compute the value using the table below.
3. Explain the assumptions we made (does it matter what the distribution of the program's runtime is?) and how realistic they are.

x	-2.5	-2.0	-1.5	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5
$\Phi(x)$	0.01	0.02	0.07	0.16	0.31	0.5	0.69	0.84	0.93	0.98	0.99

Solution sketch 1. Statement of the theorem: Let X_1, X_2, ... be identically distributed i.i.d. random variables with mean μ and variance σ^2. Define $Y_n = \frac{(X_1+\cdots+X_n)-n\mu}{\sqrt{n}\cdot\sigma}$.

Then

$$Y_n \xrightarrow{d} N(0,1).$$

In other words, if F_n is the cummulative distribution function of Y_n, then

$$\lim_{n\to\infty} F_n(x) = \Phi(x) \quad \text{for every } x \in \mathbb{R}.$$

We say that the sequence Y_n converges to $N(0,1)$ *in distribution*.

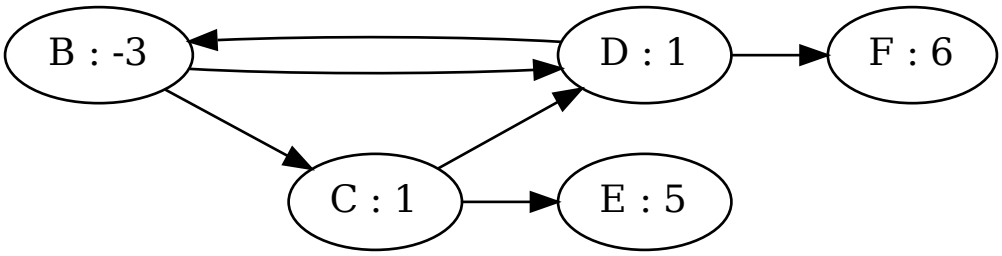
2. Let X_i denote the runtime for the i-th input. The event $X_1 + \cdots + X_{100} > 10.3$ is equivalent to $Y_{100} > \frac{10.3-100\cdot 0.1}{\sqrt{100}\cdot 0.02} = 1.5$. According to the CLT, $P(Y_{100} < 1.5) \approx \Phi(1.5)$. Therefore, the desired probability is approximately $1 - \Phi(1.5) \approx 1 - 0.93 = 0.07$, that is, roughly 7%.

3. The CLT does not require any special assumptions about the distribution of the individual X_i (as long as the variables are identically distributed with finite mean and variance). However, we do assume independence — which might be problematic; for example, if the computer is overloaded, it may run slower on all inputs. We also replace the limit with the value at $n = 100$, but this is typically not an issue when applying the CLT.

5 Bellman equation (specialization UI-SU, UI-ZPJ, UI-ROB)

The robot operates within a finite state space S. For each state $s \in S$, we know the set of possible actions $A(s)$ and the probability $P(s'|s,a)$ that the robot transitions from state s to state $s' \in S$ given the action $a \in A(s)$. The robot receives a reward $R(s)$ when traversing a state $s \in S$, but the total utility from state s' is discounted by a factor of $\gamma = 0.9$.

(1) Write the equation for calculating the maximum utility $U(s)$ in state s.



The diagram illustrates the state space, the rewards in each state, and two possible actions from each state, except for the terminal states E and F. The probability that the robot transitions to a state indicated by the directed edge of the chosen action is 0.8, while with the remaining probability of 0.2, it transitions to the state indicated by the second possible action.

(2) Write the system of equations whose solution yields the utilities of the individual states, given the robot's actions (policy) $B \to C$, $C \to D$, and $D \to F$.

(3) Which algorithm would we use to solve this system?

(4) How can we determine from the computed utilities whether the specified choice of actions yields the maximum possible utilities for all states?

(5) How would we proceed to find actions for all states that maximize utility?

Solution sketch

1. $U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s' \in S} P(s'|s,a)U(s')$
2. $u_b = -3 + 0.9(0.8u_c + 0.2u_d)$

   $u_c = 1 + 0.9(0.8u_d + 0.2 \cdot 5)$

   $u_d = 1 + 0.9(0.8 \cdot 6 + 0.2u_b)$
3. Any appropriate algorithm for solving a system of linear equations: Gaussian elimination, Gauss–Jordan elimination, approximate value iteration, etc.
4. For every state s, we check whether the maximum in Bellman's equation is attained for the chosen action.
5. If the condition in (4) is not satisfied, we modify policy by choosing actions maximizing (1) for the current utility. We repeat steps (2-4) until (4) is satisfied.

6 Linear regression (specialization UI-SU, UI-ZPJ)

1. For training data set X_{train} and target values t_{train} describe the linear regression model, and a loss function that is minimized in the linear regression method.
2. Explain how to explicitly calculate the optimal solution on the training data.
3. Formulate the optimization using the stochastic gradient descent.
4. Comment on advantages and disadvantages of both approaches, especially from the point of view of possible error checking on a validation data set.

Solution sketch Model definition and detailed expressions are in the slides for first two lectures on NPFL129.

We need to solve the equation where the derivative of the loss is equal to zero. This can be done either using the inverse matrix, or using the SVD decomposition. The disadvantage of this method is that inverting the matrix is slow and the matrix is large. Additionally, the solution may overfit the training data.

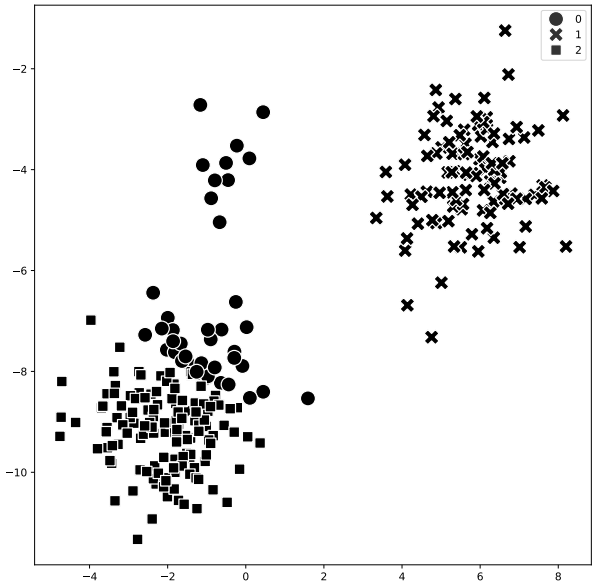
SGD requires less memory and we can monitor the validation loss and use early stopping to avoid overfitting.

7 Clustering (specialization UI-SU, UI-ZPJ)

1. Describe briefly the k-means algorithm. How are the centroids (cluster centers) initialized? How do their positions change during the run of the algorithm? When does the algorithm terminate?
2. What objective does the k-means algorithm optimize?
3. We ran the k-means algorithm for $k = 3$ on data with three clusters shown below. The different markers (×, •, ■) show the clusters found by the algorithm. Why did the algorithm fail to find the optimal clusters according to the objective above? How can we increase the chance that the algorithm finds the optimal clusters?

Solution sketch

1. The initial centroids can be selected randomly from the data. During the run of the algorithm, each data point is first assigned to the closest centroid and then the centroid is recomputed as the average of data points assigned to it. These steps iterate until the algorithm converges (the assignment of points to clusters does not change), or for a specified number of iterations.
2. The algorithm minimizes
$$\sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} ||x_i - \mu_k||^2,$$
where N is the number of points, K is the number of clusters, and $z_{i,k}$ denotes that the point x_i is assigned to cluster with center μ_k ($z_{i,k} = 1$ if the point is assigned to the cluster and 0 otherwise).



3. The middle cluster that is assigned to the part of the left one is much smaller than the other two clusters. The chance for a better result can be increased by re-starting the algorithm with different initial centroids and taking the best found clustering based on the objective above.

8 EM Algorithm (specialization UI-SU)

1. Describe the EM algorithm.
2. Consider the probabilistic model defined by a Bayesian network with a variable $Cluster$ with values from the set $\{1, 2\}$ as a parent of variables $Color$ and $Holes$ with domains $\{blue, green\}$ and $\{yes, no\}$, respectively. There is no direct edge between the variables $Color$ and $Holes$.

   Create a list of the model parameters and initialize them by values from the set $\{0.4, 0.6\}$. The parameters should form a valid Bayesian network and the conditional probabilities given $Cluster = 1$ and given $Cluster = 2$ should not be identical.
3. Choose one of the parameters in the previous model and perform its EM update from your initial parameters using the following data. The questionmark "?" denotes missing values.

Color	Holes	Cluster
blue	no	?
blue	yes	?
blue	no	?
green	yes	?
green	yes	?

You may use the following approximations: $0.6^0 * 0.4^3 \approx 0.064$, $0.6^1 * 0.4^2 \approx 0.096$, $0.6^2 * 0.4^1 \approx 0.144$, $0.6^3 * 0.4^0 \approx 0.216$.

Solution sketch

1. The EM algorithm is used to estimate model parameters from data with missing values or latent variables. It takes some initial model parameters and iterates steps E and M until convergence.

   E Infer the expected values of the hidden variable to "complete" the data.

M Find the maximal likelihood parameter estimate for the full data.

2. The model parameters (with the initial values for the next step):

parameter	initial value	denoted by
$P(Cluster = 1)$	0.6	θ
$P(Color = blue \mid Cluster = 1)$	0.6	θ_{B1}
$P(Color = blue \mid Cluster = 2)$	0.4	θ_{B2}
$P(Holes = yes \mid Cluster = 1)$	0.6	θ_{H1}
$P(Holes = yes \mid Cluster = 2)$	0.4	θ_{H2}

3. The simplest update is for the parameter θ. We iterate the data $[color_j, holes_j]$ rows and sum $\sum_j P(Cluster = 1|color_j, holes_j)$ divided by the number of data rows.

Color	Holes	$P(Cluster = 1\|color_j, holes_j)$
blue	no	$\frac{\theta\cdot\theta_{B1}\cdot(1-\theta_{H1})}{\theta\cdot\theta_{B1}\cdot(1-\theta_{H1})+(1-\theta)\cdot\theta_{B2}\cdot(1-\theta_{C2})} = \frac{0.6*0.6*0.4}{0.6*0.6*0.4+0.4*0.4*0.6} = \frac{0.6*0.6*0.4}{0.4*0.6} = 0.6$
blue	yes	$\frac{\theta\cdot\theta_{B1}\cdot\theta_{H1}}{\theta\cdot\theta_{B1}\cdot\theta_{H1}+(1-\theta)\cdot\theta_{B2}\cdot\theta_{H2}} = \frac{0.6*0.6*0.6}{0.6*0.6*0.6+0.4*0.4*0.4} = \frac{0.216}{0.216+0.064} = 0.77$
blue	no	as above, 0.6
green	yes	$\frac{\theta\cdot(1-\theta_{B1})\cdot\theta_{H1}}{\theta\cdot(1-\theta_{B1})\cdot\theta_{H1}+(1-\theta)\cdot(1-\theta_{B2})\cdot\theta_{H2}} = \frac{0.6*0.4*0.6}{0.6*0.4*0.6+0.4*0.6*0.4} = 0.6$
green	yes	as above, 0.6

For our initial parameters, the (rounded) updated value θ is $\frac{4\cdot0.6+0.77}{5} = \frac{3.17}{5} = 0.634$.

9 Noisy channel and machine translation (specialization UI-ZPJ)

1. Describe how the machine translation task can be transformed to the noisy channel model (in the context of the traditional statistical machine translation).
2. Explain, how to train parameters of both main components (models) in this approach.

Solution sketch

– The classical statistical machine translation is often formulated as the decoding problem in the noisy channel model, that searches for the most probable sentence in the target language e that could generate the sentence in the source language f:

$$e^* = \arg\max_e P(e|f)$$

Using the Bayes rule, we can rewrite this expression as:

$$e^* = \arg\max_e P(f|e) \cdot P(e)$$

Where:

– $P(f|e)$ is the **translation model** – models how the sentence e would be translated to f,

– $P(e)$ is the **language model** – expresses the probablity and fluency of the sentence in the target language

– The language model is typically trained on large monolingual corpora in the target language. Often, n-gram models are used, that approximate the probability of a sentence as a product of conditional probabilities:

$$P(e) \approx \prod_{i=1}^{n} P(e_i|e_{i-1}, \ldots, e_{i-n+1})$$

The probability is estimated using relative frequencies of word sequences, typically with additional smoothing.

– The translation model is trained from **paralel corpora**, that contain corresponding sentences in the source and target languages. The training consists of:

1. Word alignment, e.g. using IBM models.
2. Phrase extraction based on the alignment.

3. Computation of translation probabilities, e.g.:

$$P(f|e) = \frac{\text{frequency of pair } (f, e)}{\text{frequency of } e}$$