State Final Examination (Sample Questions)

Fall 2022

1 Programming (3 points)

Choose an arbitrary mainstream, object-oriented, statically typed programming language (C++, C#, or Java), denote your selection in your answer. Then:

1. Implement/declare a suitable object-oriented interface (using interfaces or abstract classes) for non-oriented graphs $G = (V, E)$ in their most common form (each edge is between exactly two vertices, there is at most one edge between each pair of vertices). The main motivation is to define a common grounds for different graph implementations (representing the graph as list of neighbours, as matrix of incidence, ...), so that a programmer of graph algorithms (e.g., BFS, minimal spanning tree, ...) can write the code only once and it will still work with various graph representations.

   The interface should cover 3 basic entities (*vertex*, *edge*, and *graph*). The *vertex* provides access to all adjacent edges, the *edge* provides access to its two adjacent vertices, the *graph* provides access to the list of all vertices and list of all edges. The interface provides read-only access to this data – i.e., the graph structure is immutable.

   Each vertex and each edge has a *tag*. Vertex tag may indicate whether the vertex has been visited, for instance. Edge tag may indicate its weight. In general, different tag types may be suitable for different applications, however, a particular graph instance has fixed data types for vertex tags and for edge tags. The interface should be sufficiently generic so that the programmer may choose an arbitrary vertex tag and edge tag types for the graph instance. Furthermore, the interface will provide means to read and write the tag of each vertex and edge.

2. Consider an implementation of your interfaces where vertex tags indicate the membership of the vertex in a particular connected component of the graph ((type `int`, zero-based index) and edge tags represent their lengths (type `float`, or similar type in your language of choice). The edge tags are initially set, the vertex tags must be computed by you.

   Write the body of a function, using your interface, which will receive a graph and a real number r as arguments. Your code will compute the connected components of the graph and set the vertex membership information in each vertex tag (actual values in the tags are not important, the important thing is that vertices in the same component have the same tag). Edges which are longer than r are ignored when computing the components (i.e., as if they were not present in the graph at all).

2 Virtual memory (3 points)

Consider a processor architecture with support for paging and 32-bit virtual and physical addresses. Assume (for simplicity) that the processor uses a single-level page table to perform address translation. The page size is 4 KiB. In addition to essential attributes, each page allows setting whether a process is allowed to modify its contents and/or execute code in it. The processor also provides information about accesses to each page, indicating whether a page has been accessed and whether it was written to.

Part A

1. Design and illustrate the format of a page table entry and explain the meaning of all fields. The page table entry must contain all necessary information and must be efficiently accessible to the processor (at most 1 memory read).
2. Give the number of entries in the single-level page table and determine the amount of memory it will occupy for each running process.

Part B

1. Using pseudocode, write a function `create_pte`, that will create a page table entry for the given physical memory address and page attributes. The function should have the following (or similar) signature:

   ```
   pte_t create_pte(address_t phys_addr, bool present, bool writable, bool executable)
   ```

   The `address_t` type is an integer type that allows storing the physical memory address given in parameter `phys_addr`. The `pte_t` type is a suitable integer type that represents the page table entry you designed (and meets efficient-access criterion).

2. Using pseudocode, write a function `set_mapping`, that will store the given page table entry for the given virtual address into the page table. The function should have the following (or similar) signature:

   ```
   void set_mapping(pte_t[] page_table, address_t virt_addr, pte_t entry)
   ```

   Assume that the page table is represented as an array of entries of type `pte_t`, which means that you can treat the `page_table` parameter as an array. The `virt_addr` parameter represents the virtual memory address for which you are supposed to set the page table entry given in the `entry` parameter.

Part C

Assume that a **contiguous** block of 4 virtual memory pages starting at address `0x0000B000` has been mapped to a **contiguous** block of physical memory. The block represents a part of process heap, its contents can be therefore read and modified, but it is not possible to execute code in it. Also assume that a process **read/loaded** the contents of a 4-byte variable from virtual address `0x0000CE90`, which resulted in the processor accessing physical memory at address `0xA1018E90`. The process then **wrote/stored** a modified value of the variable to memory at virtual address `0x0000D854`.

1. List physical memory addresses that contain the page table entries for the above block of memory, given that the process page table is stored in a contiguous block of physical memory starting at address `0x13400000`.
2. Give the (hexadecimal) values representing the page table entries (including all page attributes) for the above range of virtual memory.

3 Sequences (3 points)

Let $(a_n) = (a_1, a_2, \ldots)$ be a sequence of real numbers.

1. Define what it means that (a_n) is covergent (has a finite limit).
2. Define what it means that (a_n) is unbounded.
3. Prove that if (a_n) is unbounded then it is not covergent.

4 Bases and orthogonal bases (3 points)

Consider the matrix $A = BC$, where

$$B = \begin{pmatrix} 1 & 3 \\ 2 & 2 \\ 1 & 2 \\ 1 & 0 \\ 1 & -1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix}.$$

1. Find a basis of the column space (i.e., the range) of A.
2. Define the term orthogonal basis.
3. Find an orthogonal basis of the column space of A.

5 Graph Coloring, DPLL (AI-ML) (specialization question – 3 points)

Let $G = (V, E)$ be an undirected graph with N vertices ($|V| = N$).

1. Define the term "conjunctive normal form" of a formula in propositional logic.
2. Write a formula in propositional logic that is satisfiable if and only if the graph G is colorable by k colors. Express the formula in conjunctive normal form (CNF) for a given graph G and a given $k > 0$.

3. Write the formula for a complete graph with three vertices and $k = 3$. Demonstrate how to use the DPLL algorithm with this formula.

6 Overfitting and regularization (AI-ML) (specialization question – 3 points)

We want to train a linear model for regression.

1. Define the mean squared error (MSE). Define the loss function used for L_2 regularization.
2. Assume we have two linear models M_1, M_2 for data with four attributes. Let

$$\begin{aligned} M_1(\vec{x}) &= x_1 - 2x_2 + 4x_3 - x_4 + 3 \\ M_2(\vec{x}) &= x_2 - 5x_3 + x_4 - 1. \end{aligned}$$

   Assume that both models have the same MSE for the given data. Which of these models has lower value of the L_2 regularization loss?
3. Describe at least one other technique used to improve the generalization properties of machine learning models (in general, does not have to be for linear models).

7 Decision trees (AI-ML) (specialization question – 3 points)

1. Define the decision tree. Describe the algorithm for training of decision trees.
2. Define the splitting criteria used for classification and for regression (at least one criterium for classification and one for regression).
3. Consider data with categorical attributes A_1, A_2 and target value C in the table below. Which attribute (and why) will be selected for splitting in the root of a decision tree for regression?

A_1	A_2	C
0	0	5
0	0	3
0	1	4
1	0	6
1	1	2
1	1	4

8 k nearest neighbors (AI-ML) (specialization question – 3 points)

1. Describe how the technique of k nearest neighbors (k-NN) works for classification and for regression: How is the model trained? How are the predictions made?
2. Consider data with two attributes A_1 a A_2 and two classes $C \in \{0, 1\}$ given in the table below. Draw a plot of the data. How would the k-NN algorithm for $k = 3$ classify new input $(5, 5)$? Use Manhattan distance.
3. Would the answer for the question above change for different k? How? Give at least one value of k for which the answer is different (if such a k exists).

A_1	A_2	C
0	3	0
2	2	0
3	0	0
4	2	0
6	6	1
4	5	1
1	1	1