State Final Examination (Computer Science Sample Questions)

Fall 2022

1 Intersection of regular expressions (3 points)

1. Give the definition of a regular expression. Formulate Kleene's theorem.
2. Consider the following pair of regular languages over the alphabet $\Sigma = \{a, b\}$:

$$R_1 = ((a+b)(a+b)(a+b))^*$$
$$R_2 = (a+b)^*a$$

Construct a *deterministic* finite automaton A accepting the intersection of the languages given by R_1 and R_2, that is, such that the following holds:

$$L(A) = L(R_1) \cap L(R_2)$$

2 Graph connectivity (3 points)

1. Define a strongly connected component of a directed graph $G = (V, E)$.
2. Describe an algorithm (as fast as possible) which computes the decomposition of G into its strongly connected components.
3. The police department of Poobanana made all streets one way. The mayor originally claimed that one can still (legally) get from any intersection to any other intersection, but after being confronted with a counterexample, he made his claims weaker. Say that an intersection x is *good* if, for every y reachable from x, one can also reach x from y. The mayor claims that 95% of intersections in Poobanana are good. Design an algorithm which verifies his claim. The algorithm should ideally run in linear time.

3 SQL Querying (3 points)

Consider the following relational schema:

- Actor (IDU, first_name, last_name, nationality, age), IDU is a key
- Movie (ID, title, genre, color), ID is a key
- Acts (IDU, ID), IDU and ID form the key, IDU is a subset of Actor.IDU, ID is a subset of Movie.ID

Implement the following in SQL:

1. Add entries about the movie 'The Imitation Game' and its actor named Benedict Cumberbatch. (Make up the rest of the information.)
2. Create the table Director (with the attributes ID, first name, last name) and add the respective relation to films (i.e. which films the director directs). Ensure referential integrity.
3. List the names of actors who play in blockbusters (movies with more than 200 actors).
4. Remove the color column for movies.

4 Programming - graphs (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).

5 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.

6 Routing table (3 points)

Let us have a standard router using the following routing table:

Destination	Netmask	Interface	Gateway	Metric
195.113.19.0	255.255.255.0	(A) 195.113.19.20	on-link	1
172.217.23.0	255.255.255.0	(B) 172.217.23.55	on-link	1
216.58.0.0	255.255.0.0	(C) 216.58.201.78	on-link	1
185.17.100.0	255.255.255.0	(B) 172.217.23.55	(D) 172.217.23.111	10
185.17.200.0	255.255.255.0	(B) 172.217.23.55	(E) 172.217.23.222	10
104.0.0.0	255.0.0.0	(C) 216.58.201.78	(F) 216.58.90.100	10

First, explain the meaning of the individual columns in the routing table.

Next, we gradually receive the following standard IPv4 datagrams on the network interface (A) 195.113.19.20:

1. Datagram for recipient (G) 185.17.200.50
2. Datagram for recipient (A) 195.113.19.20
3. Datagram for recipient (F) 216.58.90.100
4. Datagram for recipient (H) 74.6.231.21
5. Datagram for recipient (I) 104.103.85.139
6. Datagram for recipient (J) 255.255.255.255
7. Datagram for recipient (K) 216.58.255.255

In order to simplify our situation, we will symbolically denote all the mentioned IP addresses by letters.

For each of the previously enumerated situations, describe how a given datagram will be processed. In case it will be forwarded, in which direction and to what MAC (EUI-48) address within the data link L2 layer? It means a specific well-known MAC address, otherwise the MAC address corresponding to our IP addresses (A) to (K). Explain and justify.

7 Codes (DMS) (3 points)

1. Define the following concepts: A code over the alphabet Σ, length, size, and the minimum distance of a code.
2. Suppose C is a code over alphabet Σ of length ℓ, size s and minimum distance $d \geq 2$. Let C' be the code obtained from C by deleting the last letter of each code word. What is the length and the size of C'? Prove that the minimum distance of C' is at least $d-1$.
3. Prove that every code of length ℓ over the alphabet Σ and of minimum distance d has size at most $|\Sigma|^{\ell-d+1}$.

8 Ramsey theory (DMS) (3 points)

1. Formulate the Ramsey theorem for graphs, with arbitrary number of colors.
2. Let $R(a,b)$ be the smallest integer such that every graph with $R(a,b)$ vertices contains a clique of size a or an independent set of size b. Prove that $R(a,b) \leq R(a-1,b) + R(a,b-1)$.
3. Show that for every positive integer b, there exists a triangle-free graph of chromatic number at least $(R(3,b+1)-1)/b$.

9 Matchings in graphs (DMS) (3 points)

1. State the Tutte's theorem on the existence of perfect matching.
2. Which of the following claims are true and why?
   (a) Every 2-edge-connected 3-regular graph has a perfect matching.
   (b) Every 4-edge-connected 4-regular graph has a perfect matching.
   (c) If a graph G with n vertices has a perfect matching, then every independent set in G has size at most $n/2$.
   (d) If the largest independent set in a graph G with n vertices has size at most $n/2$, then G has a perfect matching.

   Justify your answers.

10 Basic formalisms for description of natural languages (CL) (3 points)

1. List and describe three basic components of Chomsky's Transformational grammar
2. Explain the notion of transformation in Chomsky's Transformational grammar – which two parts define transformations?
3. Why it is necessary to use typed feature structures in unification grammars?

11 Morphological and syntactic analysis of a natural language (CL) (3 points)

1. What is a morphological tag? List at least five features that are often encoded in morphological tag sets.
2. Describe dependency trees, constituent trees and differences between them.
3. Draw (a) a constituency parse tree and (b) a dependency tree for the sentence *I saw a fox.*

12 Language modeling (CL) (3 points)

1. Consider the following toy corpus: *the cat cut the hat* How many different bigrams of characters (including whitespace) are in this corpus? How many occurences are there in total (i.e. including repetitions)?
2. Explain N-gram language modeling in Natural Language Processing. Write down the formula for 3-grams (trigrams).
3. Why do we need smoothing in language modeling?