Written Exam

10:00 – 12:30, February 4, 2020

Entrance Examination (AY 2020)

Department of Computer Science
Graduate School of Information Science and Technology
The University of Tokyo

Notice:

(1) Do not open this problem booklet until the start of the examination is announced.

(2) Answer the following 4 problems. Use the designated answer sheet for each problem.

(3) Do not take the problem booklet or any answer sheet out of the examination room.

Write your examinee's number in the box below.

Examinee's number	No.

Problem 1

Suppose that a set of real numbers $A = \{a_1, a_2, \cdots, a_n\}$ $(n > 0)$ is given, where all the numbers in A are different from each other. Assume that the only operations allowed for real numbers are comparisons between two real numbers. You may use appropriate data structures when describing algorithms below.

Answer the following questions.

(1) Describe an algorithm that finds the smallest number from A. The algorithm cannot use real number comparison operations more than n times.

(2) Describe an algorithm that partitions A into two sets, $S = \{a_i | a_i < x\}$ and $L = \{a_i | a_i \geq x\}$, for a given real number x. The algorithm cannot use real number comparison operations more than n times.

(3) Describe an algorithm that finds the k smallest numbers from A for a given integer k $(0 < k \leq n)$. The average-case time complexity of the algorithm must be $O(k + n)$. It is not required that the output of the algorithm is sorted.

(4) Describe an algorithm that finds the median of A. The average-case time complexity of the algorithm must be $O(n)$. Here, the *median* of A is defined as the $\lfloor \frac{n+1}{2} \rfloor$-th smallest number in A, where $\lfloor m \rfloor$ is the integer that satisfies $m \leq \lfloor m \rfloor < m + 1$.

(5) Give the number of real number comparison operations required in the worst case for the algorithm you described in question (4).

Problem 2

Suppose that d is a real number represented as a floating point number, where $\frac{1}{2} < d < 1$, and that we wish to compute an approximation of $\frac{1}{d}$ by using only addition, subtraction, and multiplication (i.e., without using a primitive instruction or library for division). Let f be the real function defined by $f(x) = \frac{1}{x} - d$.

Answer the following questions.

(1) Let $y = p_\xi(x)$ be the equation of the tangent line of the graph $y = f(x)$ at $x = \xi > 0$. Express x_0 such that $p_\xi(x_0) = 0$ in terms of ξ and d.

(2) Express the value of ξ that satisfies $x_0 = \xi$ in terms of d, for x_0 in question (1) above.

(3) Construct an iterative algorithm which, given an initial approximation of $\frac{1}{d}$ as an input, iteratively improves the approximation of $\frac{1}{d}$. You may only use addition, subtraction, and multiplication; you must not use other mathematical functions and operations such as division. You need not show the condition for the convergence to $\frac{1}{d}$.

(4) Show the order of the convergence of the algorithm obtained in question (3) when the input is sufficiently close to $\frac{1}{d}$.

Problem 3

Consider a processor that executes every instruction in one clock cycle with no stall, excepting load/store instructions. There is a cache memory between the processor and a main memory. In case of a *cache hit* for a load/store instruction, it takes two clock cycles to complete the instruction. In case of a *cache miss* for a load/store instruction, it takes 100 clock cycles to complete the instruction.

Answer the following questions.

(1) Suppose that, in the execution of a program on the processor, 20% of the executed instructions are load/store instructions, and the average cache hit rate is 95%. Calculate the average IPC (instructions per cycle) up to two places of decimals.

(2) Describe briefly what the term "capacity miss" means in the context of a cache memory.

(3) Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be $N \times N$ dense matrices. Assume that the following pseudocode for a multiplication of $\boldsymbol{A}$ and $\boldsymbol{B}$ is executed on the processor, and the average IPC is low due to numerous capacity misses on the cache memory. Describe a programming technique to reduce capacity misses in a cache memory for achieving higher IPC. Explain why the technique can reduce capacity misses by indicating a pseudocode. If necessary, you can introduce your own assumptions, for example, on the matrix sizes and the capacity of the cache memory.

```
float a [N * N]; /* Matrix A */
float b [N * N]; /* Transposed matrix of matrix B */
float c [N * N]; /* Initialized to zero matrix */
int i, j, k;

for(i=0; i<N; i++){
  for(j=0; j<N; j++){
    for(k=0; k<N; k++){
      c[i*N + j] += a[i*N + k] * b[j*N + k];
    }}}
```

(4) It is known that a W-way set-associative cache memory ($W > 1$) often achieves a better cache hit rate than a direct-mapped cache memory with the same capacity. Explain why.

Problem 4

Let Σ be a finite alphabet (i.e., a finite set of letters). For $w_1, w_2 \in \Sigma^*$, we write $w_1 \cdot w_2$ for the concatenation of w_1 and w_2, and write ϵ for the empty sequence. For a map $f \in \Sigma \to \Sigma^*$, we define $f^* \in \Sigma^* \to \Sigma^*$ inductively by $f^*(\epsilon) = \epsilon$ and $f^*(aw) = f(a) \cdot f^*(w)$ for any $a \in \Sigma$ and $w \in \Sigma^*$. For example, if $\Sigma_1 = \{a, b\}$ and $f_1 \in \Sigma_1 \to \Sigma_1^*$ is the map defined by $f_1(a) = ab$ and $f_1(b) = a$, then $f_1^*(ab) = aba$. For a language $L \subseteq \Sigma^*$ and a map $f \in \Sigma \to \Sigma^*$, we define $f^{-1}(L) \subseteq \Sigma^*$ by:

$$f^{-1}(L) = \{w \in \Sigma^* \mid f^*(w) \in L\}.$$

For example, for Σ_1 and f_1 above, ${f_1}^{-1}(\{a, b, aba\}) = \{b, ab\}$.

Answer the following questions.

(1) Let Σ_2 be $\{a, b, c\}$ and $f_2 \in \Sigma_2 \to \Sigma_2^*$ be the map defined by $f_2(a) = ab, f_2(b) = ba$, and $f_2(c) = a$. Compute ${f_2}^{-1}(\{c, aba, abba\})$.

(2) Let Σ_2 and f_2 be those of question (1). Let $L_2 = \{(ab)^n a \mid n \geq 0\} = \{a, aba, ababa, abababa, \ldots\}$. Express the language ${f_2}^{-1}(L_2)$ by using a regular expression.

(3) Suppose $f \in \Sigma \to \Sigma^*$, and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is a deterministic finite automaton, where Q, δ, q_0, and F are respectively the set of states, the transition function, the initial state, and the set of final states of $\mathcal{A}$. You may assume that the transition function $\delta \in Q \times \Sigma \to Q$ is a total function. Let $\mathcal{L}(\mathcal{A})$ be the language accepted by $\mathcal{A}$. Give a deterministic finite automaton that accepts $f^{-1}(\mathcal{L}(\mathcal{A}))$.

(4) Prove the correctness of your answer for question (3) above.