Warsaw Center
of Mathematics
and Computer Science

May 21st, 2015

qualification exam for PhD fellowships in WCMCS institutions

---

code . . . .

name . . . .

field of study . . . .

Each candidate chooses **5 problems** and notes the choice on the examination card by checking suiable boxes below. If more then 5 problems are chosen, only the first 5 will be evaluated.

Solution for each of the chosen problems must be written on a **separate piece of paper**.

Each piece of paper must be signed with the **four-digit code** given above. They **must not** be signed with the candidate's name.

Chosen problems:

1. ☐	2. ☐	3. ☐	4. ☐	5. ☐
6. ☐	7. ☐	8. ☐	9. ☐	10. ☐
11. ☐	12. ☐	13. ☐	14. ☐	15. ☐

Signature: . . . .

Problem 1. Solve the following optimization problem:

$$\text{find the minimizer of} \quad -\sum_{i=1}^{n} \log(\alpha_i + x_i)$$

for $i = 1, ..., n$, $x_i \geq 0$,

$$(1, ..., 1)^T (x_1, ..., x_n) = 1,$$

provided $\alpha_i > 0$ for each $i = 1, ..., n$.

Problem 2. Find the number of all subsets of the set $\{1, \ldots, n\}$, such that they do not contain subsequent two elements.

Problem 3. Let p be a prime number different from 2, and let $a \in \mathbb{Z}_p^*$. Examine a perturbation $\sigma_a : \mathbb{Z}_p^* \ni b \mapsto ab \in \mathbb{Z}_p^*$. Show that $a^{(p-1)/2} = sgn(\sigma_a)$, where sgn denotes the sign of perturbation.

Problem 4. Let l^∞ be a space of bounded sequences of numbers from $\mathbb{R}$ with the sup-norm. Given $\{a_1, a_2, \ldots\}$ Define $P\{a_1, \ldots\} = \{a_1/2, a_2/4, \ldots, a_i/2^i, \ldots\}$, $P \colon l^\infty \to l^\infty$.

Show that P is a compact operator (if $\{f_n\} \subset l^\infty$ uniformly bounded then $\{Pf_n\}$ contain a Cauchy subsequence).

Problem 5. For the matrix

$$A = \begin{bmatrix} 3/2 & 1/2 & 0 & 0 \\ 1/2 & 3/2 & 0 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{bmatrix}$$

we apply the inverse power method

$$\vec{y}_k := (A - \sigma I)^{-1} \vec{x}_{k-1}, \qquad \vec{x}_k := \vec{y}_k / \|\vec{y}_k\|_2, \qquad k = 1, 2, \ldots$$

with the initial vector $\vec{x}_0 = [1, 0, 0, 1]^T$. Find $\sigma \in \mathbb{R}$ for which

(a) the sequence $\{\vec{x}_k\}_{k=0}^{\infty}$ converges to an eigenvector of A?

(b) the sequence $\{\eta_k\}_{k=0}^{\infty}$, where $\eta_k = \vec{x}_k^T A \vec{x}_k$, converges to an eigenvalue A?

Problem 6. Let $X_1, ..., X_n$ be a random sample from distribution with the probability density

$$f_\theta(x) = \begin{cases} \frac{\theta}{(1-x)^2} \cdot e^{-\theta x/(1-x)} & \text{for } 0 < x < 1 \\ 0 & w\ p.p. \end{cases}$$

where $\theta > 0$ is a unknown parameter. Let $\theta_0 > 0$ be known number. Construct test of hypothesis $H_0 : \theta \geq \theta_0$ agaist alternative $H_0 : \theta < \theta_0$, which is uniformly most powerful test at confidence level α.

(Quartiles of typical distributions are known)

Problem 7. Let $f : \mathbb{R} \to \mathbb{R}$, $f(0) = 0$ be a continuous function. Consider the initial problem

$$\frac{dx}{dt} = f(x), \qquad x(0) = x_0.$$

(a) Let $f \in C^1(0; \infty)$, $f(x) > 0$ for $x > 0$, $\lim_{x \to 0^+} f'(x) = +\infty$ and there exists a primitive function F(x) of function $\frac{1}{f(x)}$ with $F(0) = 0$.

Solve the problem for $x_0 = 0$.

(b) Let $f \in C^1(\mathbb{R})$, concave, unimodular, having just one maximum at point $x = a > 0$. Analyze the solution (drawing them) for any $x \geq 0$ i $t \geq 0$.

Problem 8. Let $X = (X_1, X_2)$ be a two dimensional random vector with distribution function $N(0, I_2)$, where I_2 is 2×2 identity matrix. Compute the probability $\mathbb{P}[|X_1| + |X_2| \leq 1]$.

Problem 9. Find an algorithm that solves the following decision problem in linear time.

Input: Undirected planar graph G.
Question: Does graph G contain (as a subgraph) a four-element clique?

Problem 10. Consider a given graph $G = (\{1, \ldots, n\}, E)$ together with a function $w : E \to \mathbb{R}_+$, two distinguished distinct vertices $s, t \in V$, and a number $k \in \mathbb{N}_+$. By length of a path in G we mean sum of values of w for all edges in the path. The paths, considered as sequences of vertices, can be ordered lexicographically. Find an efficient algorithm that computes the kth path in the lexicographic order in the set of the shortest paths from s to t, or detects that no such path exists.

Problem 11. What is the computational complexity of the following decision problem?

Input: Regular expression E and a natural number n, represented in binary.
Question: Does $L(E)$ contain at least one word of length n?

Problem 12. For a language L, let *half*(L) denote the language $\{w : ww \in L\}$. Is the following decision problem decidable?

Input: Context-free language L.
Question: Is *half*(L) nonempty?

Problem 13. Is it possible to express in first-order logic that an undirected graph is regular?

Recall that an undirected graph is *regular*, if all vertices have the same degree. We represent a graph as a relational structure $\langle V, E \rangle$, where $E \subseteq V \times V$.

Problem 14. We say that a sequence a of integers of length n *covers* a sequence b, if for every $i \in \{1 \ldots n\}$, sum of the first i elements of a is not smaller than sum of the first i elements of b.

Below we give an example, possibly erroneous implementation of function `covers` that decides whether a covers b. Find and correct an error in the implementation, and then verify the function `covers` using Hoare logic.

```
int covers(int[] a, int la, int[] b, int lb) {
    int s = 0;

    for(int i=0; i<la; i++) {
        s += a[i] - b[i];
        if (s<=0) return 0;
    }
    return 1;
}
```

Problem 15. Consider a system for detecting suspicious network activity. It works on a multi-processor router and selectively monitors the network traffic passing through the router. The numerous threads comprising the system work as follows. Each incoming packet is immediately classified by highly specialized hardware circuits into one of so-called streams. From time to time, for each stream, the system selects a random packet, which is then subjected to a more detailed and computationally expensive analysis. Based on the analysis, various alarms for the network administrators can be triggered.

Your task is to implement the selection of a random packet from a stream and to prove the correctness of your solution. The solution should aim to maximize the performance and scalability of the system with respect to the total number of streams, S, handled by the router. More specifically, you have to implement the following functions and data structures:

Struct/class `Stream` The implementation should specify all and only those fields of `Stream` that describe the data structures required by your solution.

Function/method `void packetClassified(Stream * s, Packet * p)` This function is invoked whenever the router has received an incoming packet, `p`, and has classified it into a stream, `s`, (objects `s` and `p` are allocated and garbage-collected automatically by the router). The function may be invoked independently and in parallel by many threads, for instance, representing different network interfaces of the router. Nothing else is known about the distribution of these invocations in time and between the streams.

Function/method `Packet * getRandomPacket(Stream * s)` This function is invoked whenever the router needs a random packet from a stream, `s`, for a more detailed analysis. It must return a packet selected uniformly at random among all packets classified into stream `s` in the period either from the last invocation of the function for stream `s`, if there was such an invocation, or since the start of the system, if the function has not been invoked for stream `s` so far. If in this period no packets were classified into stream `s`, the function must return `null`. The function may be invoked independently and in parallel by many threads. Nothing else is known about the distribution of these invocations in time and between the streams.

Function/constructor `void initStream(Stream * s)` This function is invoked when the router boots, before it starts receiving any packets. Its task is initializing the data structures for stream `s`.

For controling the lifetime of packets, one should use functions/methods `void pinPacket(Packet * p)` and `void unpinPacket(Packet * p)`. Both these functions are safe with respect to invocations by concurrent threads. The first marks a packet, so that it will not be garbage-collected. In particular, each packet for which `pinPacket` is *not* invoked within function `packetClassified` will be garbage-collected after `packetClassified` returns. By symmetry, `unpinPacket` unmarks the packet, thereby making it a candidate for garbage collection. It may be assumed that the function analyzing the randomly selected packets calls `unpinPacket` after having analyzed each of the packets returned by `getRandomPacket`.

For generating random numbers, one should use the router's hardware generator of pseudorandom numbers, which is available through function `double hwRandom()`. The function returns a pseudorandom, uniformly distributed number from $[0 \ldots 1)$ It is thread-safe. It may also be assumed that operations on `double` do not suffer from numerical errors.

For managing concurrency, one can select arbitrary standard synchronization mechanisms available from the user space, but it is important to explain the properties that those mechanisms must satisfy and justify the choice of the mechanisms. It may be assumed that the implemented functions will be invoked by a fixed number of threads, P, which roughly corresponds to the number of processors available on the router.