Discrete Structures: Homework 2

Due Date: February 10, 2020. Submit PDF file to the "Homework 2" folder in ORTUS.

Problem 1: A standard 8×8 chessboard has one of the squares removed (you do not know which one), so now it only has 63 squares. Is it possible to cut the remaining chess-board into 21 rectangles of sizes 1×3?

Problem 2: Define a bijective (one-to-one and onto) function betweeen:
(a) The line segment $(0; 1)$ and the set of all real numbers.
(b) The line segment $[0; 1)$ and the set of all real numbers.

Problem 3: Define a bijective function between the set $\mathcal{A}$ of all continuous functions $f : \mathbb{R} \to \mathbb{R}$ and the set $\mathbb{R}$ itself.
Hint. You can use the fact that two continuous functions $f_1, f_2 : \mathbb{R} \to \mathbb{R}$ are equal, iff they are equal for all their rational arguments:

$$(\forall x \in \mathbb{Q},\ (f_1(x) = f_2(x))) \ \leftrightarrow\ (f_1 \equiv f_2).$$

(This fact can be proven so that, if $f_1(x) \neq f_2(x)$ for some irrational argument $x \notin \mathbb{Q}$, then either f_1 or f_2 (or both) are no longer continuous. You do not need to do this proof.)

Problem 4: Suppose we have n subsets $S_1, \ldots, S_n$ of the set $\{1, 2, \ldots, n\}$. There is a brute-force algorithm that determines whether there is a disjoint pair of these subsets. It loops through the subsets; for each subset S_i, it then loops through all other subsets; and for each of these other subsets S_j, it should loop through all elements k in S_i to determine whether k also belongs to S_j.
Give a big-O estimate for the number of times the algorithm needs to determine whether an integer is in one of the subsets.

Problem 5: Assume that every comparison between two numbers costs you 1 eurocent (other CPU time and memory cost you nothing).
Describe the optimal algorithm for finding **both** the largest and the smallest integer in a finite list of n different integers (the list is unsorted). You can describe the steps of that algorithm in human language or pseudocode similar to the K.Rosen's textbook.

Problem 6: You are given n coins with different weights. You should find the lightest and the heaviest coin using scales that allow to compare two coins at a time.
(a) Does there exist an algorithm that can find the lightest+heaviest coin using less than $(3/2)n - 2$ comparisons? Justify your answer.
(b) What is the smallest number of comparisons to **demonstrate** (e.g. as a forensics expert in a court) which is the lightest and the heaviest coin, if that forensics expert has already somehow guessed the right answer.

Problem 7: The *ternary search algorithm* locates an element in a list of strictly increasing integers by successively splitting the list into three sublists of equal (or as close to equal as possible) size, and restricting the search to the appropriate piece. Write in detail the steps of this algorithm. How many comparisons does it need in the worst possible situation?

Problem 8: You have two large numbers a, b (both can have up to 100 digits in decimal notation). You want to compute a^b using a "black box" that can multiply two numbers of any size, and one multiplication costs 1 eurocent. How many eurocents would it cost to compute a^b in the worst case?
Hint. One can clearly do this better than brute force: with up to $10^{100} - 2$ multiplications (just multiply a to itself $b - 1$ times). For example, computing a^4 can be done by just two multiplications: $a^2 = a \cdot a$, and $a^4 = a^2 \cdot a^2$. You do not need to spend three multiplications like this: $a^4 = a \cdot a \cdot a \cdot a$.

Problem 9: Find counterexamples to show that all the propositions given below are wrong:
(a) If there is an integer $m > 2$, $ac \equiv bc \pmod{m}$, and $c \not\equiv 0 \pmod{m}$, then one can cancel non-zero multiplier c on both sides of the congruence:

$$a \equiv b \pmod{m}.$$

(b) If $m \geq 2$ is an odd number, and positive integers a, b, c, d satisfy both $a \equiv c \pmod{m}$ and $c \equiv d \pmod{(m-1)}$, then

$$a^c \equiv b^d \pmod{m}.$$

Problem 10: Alice and Bob have agreed to use powers of number d in order to send each other messages. When Alice guesses a positive integer $n \in \{1, 2, \ldots, 40\}$, she does not send that integer n directly, but instead she computes and sends the remainder, if d^n is divided by 41.
Find an example of a number d that Alice and Bob can agree on, so that Bob can always understand, which n Alice meant, just looking at the remainder of d^n.
(You may need a computer and a 2-3 lines of Python code to find this out.)