Discrete Structures: Homework 2 Hints

You are not required to read or to do anything with these hints. They are provided just in case you find unsurmountable difficulties with any HW2 problem.

Problem 1 (Coloring Invariant: This is a **Problem 50** from your textbook (Rosen2019, p.115).
You could read 1.8.8 Tilings chapter from the textbook: (Rosen2019, pp. 108-111) textbook. It shows that certain chess-boards cannot be divided into 1×2 rectangles. You should answer some questions:

- How to color any chessboard so that every rectangle 1×3 has certain colors, no matter how it is positioned?
- As we know, a square has various symmetries (you can rotate or flip it). Clearly, some of the 64 choices (how to cut one tile) can be reduced to a small number of subcases (for example, it really does not matter *which* corner tile you remove; ability/inability to cut into rectangles will preserve. Make sure that you know, which subcases are independent.

Note. There are multiple ways to build new problems out of this. Many are used in high-school mathematics. See e.g. MIT paper on tilings: `https://bit.ly/2v1ddBJ`, also one written by Agnese Z. and Maruta A. (LU Neklātienes matemātikas skola): `https://bit.ly/31jimRL`.

Problem 2 (Segment-Line Bijection): The problem asks, if the number of points on the open line segment $(0; 1)$ (or the half-open segment $[0; 1)$) is the same as the set of all real numbers. See `https://bit.ly/3b9NoQT` for a discussion, how a unit circle can be mapped to the line of real numbers. There are some technical differences between this problem and your problem:

- A circle is a different line segment. Angles change from 0 to 2π (not necessarily in the interval you need).
- In the part **(b)** you should be careful, where you map the point $x = 0$. Unsurprisingly, it is easier to map open intervals $(a; b)$ into open line $(-\infty; +\infty)$ than half-open or closed intervals $[a; b)$ or $[a; b]$. But all these cases are doable: clearly all the line segments (large and small) have bijections to $\mathbb{R}$.

Note. All kinds of variations can be invented – how about a bijection from a union of two line segments $([0; 1] \cup [2; 3])$ and the set of all real numbers? Or to the set $[0; +\infty)$? Or to all the infinite sequences of 0s and 1s?

Problem 3 (Continuous functions are same as reals: This seems somewhat surprising: The number of all continuous functions $f : \mathbb{R} \to \mathbb{R}$ is the same as the number of real numbers $\mathbb{R}$. It might seem that there should be more ways to draw a wiggly continuous curve on a paper than there are points on a line, but, in fact, these are the same set cardinalities.

- To prove this, you actually need to use the fact that all functions are continuous: It is enough to know their values in all the rational points (this was given in your problem).
- The big trick is to show how to encode a (contable) set of infinite decimal fractions as a single decimal fraction. You might need to remember infinite Hilbert hotel (and hosting infinitely many infinite lines of guests there).
- If you use decimal notation, please make sure that you handle the cases where $0.4(9) = 0.5(0)$ correctly. Some numbers have two decimal representations. If you do not use decimal notation, then you do not have this worry (but you may have other worries).

Problem 4: This is a problem of how a "naive" algorithm might work: a loop in a loop in a loop, etc. Your task is to count the nested loops carefully. And give some estimates on how many times each loop is executed. For example, if you run 3 nested loops over n elements (or, perhaps, even $n/2$ loops over at least $n/2$ elements each), you get $O(n^2)$ time already.
In this problem you simply count the steps.

Problem 5:

- It turns out that it is possible to do better than $(n - 1) + (n - 2) = 2n - 3$ (first find the largest element from n elements; then find the smallest element from the $n - 1$ remaining ones).
- Maximum finding by simply comparing the current maximum with everything in the list does not yield useful information for finding minimum. You just learn that there are lots of numbers less than your current maximum.
- It is possible to do comparisons so that each comparison yields information **both** for the largest and the smallest elements.

Problem 6: This is a proof related to Problem 5 (prove that it is possible to spend just $(3/2)n - 2$ comparisons

or similar). Part **(b)** shows a new "superoptimistic" time metric: Assume that you already know the right answer. How long will it take to prove that you were right? Usually it is easier to reason in these situations, because you do not need to sort through different subcases or find out new truths. Just demonstrate your answer.

Problem 7: This is Problem 27 from (Rosen2019, p.214). Ternary search does not differ very much from the binary; there are different comparisons to make in order to know which path to follow.
Ternary search may sometimes be useful, if it is more convenient to create trees that branch three ways instead of two ways. This could happen, e.g. in a database, if the memory page (depends on hardware) fits exactly one such 3-way branching node, but storing 2-way branching node would be wasteful.

Problem 8: This is a variant of Problem 10 from (Rosen2019, p.214). Repetitive squaring a number is much more efficient to compute large powers (compared to repetitive multiplication). To get x^{1024} you need either to square just 10 times in order to get x^2, $x^4 = x^{2^2}$, $x^8 = x^{2^3}$, …, $x^{2^{10}}$. It is 10 multiplications rather than 1023.
There is still a challenge - what happens if the power is not 2^n? How to pick and multiply known values a^{2^0}, a^{2^1}, a^{2^2}, etc. to get a^b?

Problem 9: This is a variant from Problem43 (Rosen2019, p.259). In modular arithmetic you can do most of the operations with congruences (add, subtract, multiply). This problem shows that there are some things that you cannot do. These examples are not difficult to find; you just need to be flexible with what you select as m.

Problem 10: Clearly all the powers d^n (remainders, if divided by 41) will eventually start to repeat themselves, since every remainder d^k leads to the next remainder $d \cdot d^k = d^{k+1}$. The only problem: For many d, these powers start to loop too fast. Much faster than it takes to consider all the 40 possible values.
So your task is to find a d (just one of them, if there are several) such that the powers d^n take all the 40 values between 0 and 41.
Note. This problem leads to the Diffie-Hellman cryptography algorithm. Restoring integer n (modulo m) from d^n (same modulo m) is named *discrete logarithm.* See `https://bit.ly/2RP2aVe`.