ECE 374 B - Fall 2023

Homework 9

- **Submit your solutions electronically on the course Gradescope site as PDF files.** If you plan to typeset your solutions, please use the LaTeX solution template on the course web site. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app (or an actual scanner, not just a phone camera). We will mark difficult to read solutions as incorrect and move on.
- **Every homework problem must be done *individually*.** Each problem needs to be submitted to Gradescope before 6AM of the due data which can be found on the course website: https://ecealgo.com/homeworks.html.
- For nearly every problem, **we have covered all the requisite knowledge required to complete a homework assignment prior to the "assigned" date.** This means that there is no reason not to begin a homework assignment as soon as it is assigned. Starting a problem the night before it is due a recipe for failure.

Policies to keep in mind

- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details.
- **Being able to clearly and concisely explain your solution is a part of the grade you will receive.** Before submitting a solution ask yourself, if you were reading the solution without having seen it before, would you be able to understand it within two minutes? If not, you need to edit. Images and flow-charts are very useful for concisely explain difficult concepts.

See the course web site (https://ecealgo.com) for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

1. A *strongly independent* set is a subset of vertices S in a graph G such that for any two vertices in S, there is no path of length two in G. Prove that *Strongly Independent Set* is NP-hard.

2. For any integer k, the problem kPARTITION is defined as follows:

   - INPUT: A set S of kn positive integers.
   - OUTPUT: TRUE if the elements of S can be split into n subsets, each with k elements, whose sums are all equal, and FALSE otherwise.

   For this problem, you may assume that 3PARTITION is NP-hard.

   (a) Describe and analyze a polynomial-time algorithm for 2PARTITION, or prove that it is NP-hard.

   (b) Describe and analyze a polynomial-time algorithm for 12PARTITION, or prove that it is NP-hard.

3. A *domino* is a 1×2 rectangle divided into two squares, each of which is labeled with an integer[1]. In a *legal arrangement* of dominos, the dominos are lined up end-to-end so that the numbers on adjacent ends match. An example of a legal arrangement of dominos is given below:

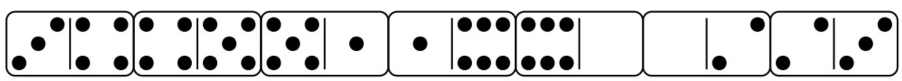

Figure 1. A legal arrangement of dominos in which every integer between 0 and 6 appears twice.

   For each of the following problems, either describe and analyze a polynomial-time algorithm or prove that the problem is NP-*complete*:

   (a) Given an arbitrary bag D of dominos, is there a legal arrangement of *all* the dominos in D?

   (b) Given an arbitrary bag D of dominos and an integer n, is there a legal arrangement of dominos from D in which every integer between 1 and n appears exactly twice?

4. For each of the following decision problems, either sketch an algorithm or prove that the problem is undecidable. Recall that w^R denote the reversal of string w. For each problem, the input is an encoding $\langle M, w\rangle$ of a Turing machine M and its input string w.

   (a) Does M either accept w or reject w^R?

   (b) If we run M on input w, does M ever change a symbol on its tape?

   (c) If we run M on input w, does M ever reenter its start state?

[1]These integers are usually represented by pips, exactly like dice. On a standard domino, the number of pips on each side is between 0 and 6, although one can buy sets with up to 9 or even 12 pips on each side; we will allow arbitrary integer labels. A standard set of dominos contains exactly one domino for each possible unordered pair of labels; we do *not* assume that the inputs to our problems have this property.