CS 131 – Problem Set 9

Problems must be submitted by Monday November 24, 2025 at 11:59pm, on Gradescope.

How to submit: This problem set contains a written portion and lean portion. For the lean portion, download and complete `ps9.lean` as instructed in the problems. Upload the written document to HW9 Written Porition and the lean file to HW9 Programming Portion on gradescope. **Note:** In case you haven't realized this on the previous autograded homework assignments, the grade you receive is whatever the autograder gives you. There are no manual adjustments. So wait for the autograder to run and make sure it gives you the grade you think you should be getting. For the lean part, make sure you upload a .lean file. Any other file format will not be graded correctly from the autograder.

Problem 1. (20 points)

Consider the integer sequence defined recursively by

$$\begin{aligned} \mathsf{myseq}(0) &= 0, \\ \mathsf{myseq}(1) &= 1, \\ \mathsf{myseq}(n+2) &= 5\,\mathsf{myseq}(n+1) - 6\,\mathsf{myseq}(n) \quad \text{for all } n \in \mathbb{N}. \end{aligned}$$

Prove, by induction on n, that for all natural numbers n,

$$\mathsf{myseq}(n) = 3^n - 2^n.$$

To solve the problem fill out `theorem myseq_bound` in `ps9.lean`. You may find the provided lemmas in `ps9.lean` file useful in completing the inductive step.

Problem 2. (20 points)

Consider a mitotic cell-division process that starts at time $t = 0$ with a single cell. Each existing cell either divides (producing two daughter cells and ceasing to exist as the parent) or does not divide (we assume no tripolar mitosis, because computer scientists get nervous when nature doesn't adhere to powers of two). Assume time proceeds in discrete steps $t = 0, 1, 2, \ldots$. For each time t let

$$d_t = \text{the number of cells that have divided by time } t \quad \text{(and thus no longer exist)},$$

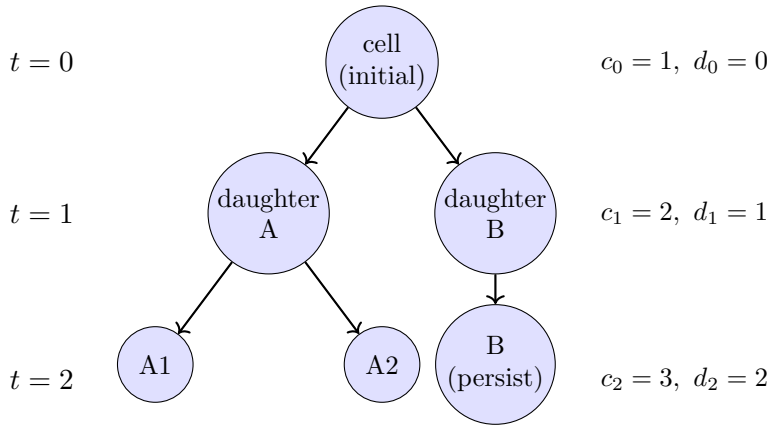
and

$$c_t = \text{the number of cells that exist at time } t \quad \text{(and thus have not yet divided)}.$$

Prove, by induction, that for every time $t \in \mathbb{N}$,

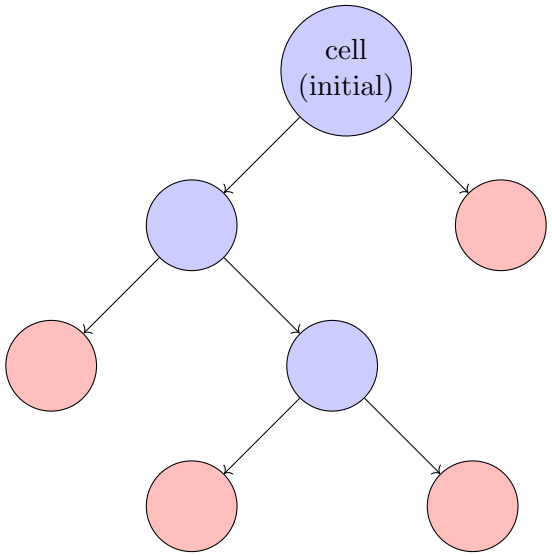
$$c_t = 1 + d_t.$$

Below is the illustration of the first two steps (times $t = 0, 1, 2$), with one row per step. This diagram shows how each cell either divides or persists without dividing over time.



One advantage of the above representation is that it makes it easy to count the cells that exist at a specific time (by counting the number of nodes is a certain row.) However, determining how many cells have divided up to a certain time requires traversing the entire tree and manually counting how many nodes have split into two. In addition, if a cell does not divide, it appears repeatedly in subsequent time steps, leading to duplicated representations of the same cell. The alternative diagram below shows each cell once. It ignores the temporal progression and simply shows the genealogical state at some time t.

Below is the illustration of the genealogical tree of indicating what cell division has occurred up to time t. The blue cells illustrate the cells that have divided by time t and the red cells illustrate the cells that exist at time t. In this example $c_t = 4$ and $d_t = 3$.



The advantage of the above representation is that it clearly distinguishes between cells that have already completed division and those that have not. Nodes at the ends of the tree — often called leaf nodes — correspond to cells that have not divided, while internal nodes — nodes that

are not leaves — represent cells that divided at some earlier time and therefore no longer exist. When every node in a tree has either two children or none, it is called a *full* binary tree. Binary cell division always produces a full binary genealogical tree.

Problem 3. (20 points)

Prove by strong induction that any positive integer can be represented as a sum of distinct powers of two. In other words, prove $\forall n \in \mathbb{Z}^+ \; \exists s_1 > s_2 > \cdots > s_\ell \in \mathbb{Z}^{\geq 0} : \; n = 2^{s_1} + 2^{s_2} + \cdots + 2^{s_\ell}$.

Hint: Remember that you are not only proving that each number has a sum of powers of two, but that **they must be distinct**.

Problem 4. (20 points)

In this problem we will be showing that any boolean function, $f : \{0,1\}^n \to \{0,1\}$, can be represented by a boolean formula of a certain size. Here we define formula size to be the number of $\cdot, +$ and $\bar{\cdot}$ operators that appear in the formula. No other operators are allowed to appear in the formula. You will prove that any boolean function can be represented by a formula of size $3(2^n) - 4$ (or less!).

For example, the function $f(x, y, z)$ which outputs 1 if a majority of the inputs is 1 can be expressed by the formula $xy + yz + xz$. This formula uses 3 multiplications, 2 additions, and 0 complements. Thus this formula has size 5 which is less than $3(2^3) - 4 = 20$.

Determining whether there are formulas of small size for a given boolean function is one of the biggest open problems in Computer Science. This problem is very closely related to the P = NP problem (probably **the** biggest open problem in CS) which you will learn more about in CS 330 and CS 332.

a) [5 points] Finish the base case, which we have started for you below. **Hint:** try to represent f and r without using constants 0 and 1.

Base case: When $n = 1$ there are four functions we need to represent and we can use at most $3(2^1) - 4 = 2$ operators. Here are the truth tables for the four functions, which we denote by f, h, g, and r.

x	f	h	g	r
0	1	0	1	0
1	1	1	0	0

We can represent $f(x) \equiv$, $h(x) \equiv$, $g(x) \equiv$, $r(x) =$, each of which uses at most two operators as desired.

b) [20 points] Complete the inductive step using induction.

Problem 5. (20 points)

Let a_n be the sequence defined by $a_1 = 1$, $a_2 = 8$, and $a_n = a_{n-1} + 2a_{n-2}$ for all $n \geq 3$. Prove that $a_n = 3 \cdot 2^{n-1} + 2(-1)^n$ for all positive integers n using strong induction. Explain why you have to use strong induction.