November 4, 2025

Problem Set 4

Exercise 1. Limits of Decidability. So far, we have been focusing on languages of TMs. The *decidable* languages, denoted as **TD** or **RE**.
Also, consider the set of languages *recognized* by TM's. These languages have TM's that accept when the input is in the language. However, for inputs not in the language, there is not specification on the TM (it does not even need to halt). This class of language is called **TR** or **R**.
However, are there languages that are not decided by a TM?

1. ($\star$) Show there are more languages than there are TMs, thus conclude that are undecidable languages.

2. ($\star$) Show that $HALT = \{\langle M, x\rangle, M \text{ halts on input } x\}$ has a TM that recognizes it, so $HALT \in \mathbf{TR}$.

3. ($\star\star$) Show that $HALT$ is undecidable, so $HALT \notin \mathbf{TD}$. (Hint: Prove it by contradiction).

Co-TR is the class of languages who's complements (every strings not in the language) are **TR**.

4. ($\star\star$) TR-TD Theorem: Show that $\mathbf{TD} = \mathbf{TR} \cap \mathbf{Co\text{-}TR}$.
   Conclude that the complement of $HALT$, denoted $\overline{HALT}$, is not in **Co-TR**.

Exercise 2. Levels of Undecidability. Now that we know there are undecidable languages, are there languages that are "more" undecidable?
This notion is captured in the Arithmetical Hierarchy. It is composed of 2 "pillars": Σ_i's and Π_i's. **TD** $= \Sigma_0 = \Pi_0 =$ **Co-TD**, by definition (note that **TD** $=$ **Co-TD**). $\Sigma_i = \{L : (x \in L) \leftrightarrow \exists y_1 \forall y_2 \cdots Q y_i : M(x, y_1, y_2, \ldots, y_i)\}$.
$\Pi_i = \{L : (x \in L) \leftrightarrow \forall y_1 \exists y_2 \cdots Q y_i : M(x, y_1, y_2, \ldots, y_i)\}$.
In both cases M is a **TD** predicate (so its boolean output can be decided by a TM). Also note that a different M may be used for each L.
The Q's are either $\forall$ or $\exists$ quantifiers based on i, since the quantifiers alternate for a total of i times, in either "pillar" starting with $\exists$ or $\forall$ for Σ_i and Π_i, respectively.

1. ($\star\star$) This will be a useful **TD** predicate for the next questions.
   A configuration is a string that includes the tape contents (except infinite sequence of blanks at the right-hand side), the position of the head and the state.
   An accepting computation path is a sequence of configurations that lead of the acceptance of the input.
   Show that $ACH = \{\langle M, x, c\rangle : c$ is an accepting computation path for M on input $x\}$ is in **TD**.

2. ($\star\star$) Show that **TR** $= \Sigma_1$ and **Co-TR** $= \Pi_1$.

3. ($\star\star$) Show that $L \in \Sigma_i$ if and only if $\overline{L} \in \Pi_i$.

4. ($\star\star$) Show that $\Sigma_i \cup \Pi_i \subseteq \Sigma_{i+1}$ and $\Sigma_i \cup \Pi_i \subseteq \Pi_{i+1}$.

5. ($\star\star\star$) Show that $ALL_{TM} = \{\langle M\rangle : L(M) = \Sigma^*\}$ is in Π_2.

6. ($\star\star\star$) Show that $COFINITE_{TM} = \{\langle M\rangle : M$ accepts all but a finite set of strings$\}$ is in Σ_3.

Exercise 3. Finding the Majority. Suppose we have an array A with n entries. If an element x appears more than $n/2$ times, we call x the *majority element* of array A. In this problem, we will consider various algorithms to find the majority element of a given array.

1. ($\star$) Show that the majority element of an array is well-defined. That is, show that any array can have at most one majority element.

2. ($\star\star$) Devise a straightforward deterministic $O(n)$ algorithm to find the majority element of an array (if it exists). How much space does this use?

However, we can solve this problem with sublinear space if we make our algorithm randomized. For the rest of this problem, you may assume that we can store numbers in $O(1)$ space.

3. ($\star$) Suppose we can randomly select an entry from our array A. How do we use this to create a randomized algorithm which is correct at least half of the time? (Hint: If an element is the majority, then if we select a random element, it will be the majority with probability at least 1/2.). What is the time and space complexity of this approach?

4. ($\star\star$) How can we modify the previous algorithm to be correct with probability at least $1 - \delta$, for some chosen $\delta > 0$?. What is this version's time and space complexity?

This randomized algorithm has efficient space complexity, but it isn't always correct. Can we find the majority deterministically and with sublinear space?

5. ($\star$) Observe that whenever two distinct elements appear, at most one can be the majority. What happens if we repeatedly remove pairs of different elements from the array? Show that if the majority exists, it will never be removed by this process

6. ($\star\star$) Design an $O(n)$ time algorithm using $O(1)$ space which utilizes this observation. (Hint: You will want to store a counter and current candidate while traversing the array.)

7. ($\star\star\star$) Prove the correctness of this algorithm. How does this algorithm compare to the randomized approach?

8. ($\star\star$) What if the array was instead given as a stream of data you could only read from once? Which of the above algorithms would still work and why?

Exercise 4: Sphere Packing via Independent Sets. A *sphere packing* is an arrangement of equal-radius balls in space where none of the balls overlap. Sphere packings are instrumental in the theory of error-correcting codes (ECCs), which are interesting in their own right and have several practical applications. ECCs also appear in the construction of the *monster group*, the largest sporadic simple group having order $\approx 8.08 \times 10^{53}$. The correspondence between sphere packings, ECCs, and the sporadic simple groups is detailed in the book *Sphere Packings, Lattices and Groups* by John Conway and Neil Sloane.

The goal of this exercise is to introduce a *graph-theoretic* construction of sphere packings via independent sets. Given a graph $G = (V, E)$, an **independent set** $S \subseteq V$ is a subset of vertices that are not connected by edges in G. In general, the problem of finding large independent sets is NP-hard. An independent set can have special meaning in certain classes of geometric graphs. Let P be a finite set of points in $\mathbb{R}^d$ and define the **unit ball graph** $G(P)$ to be the graph where P is the set of vertices, and two points are connected by an edge if their distance apart is at most 1. That is,

$$V = P \subset \mathbb{R}^d, \quad \text{and} \quad (x, y) \in E \text{ if and only if } \|x - y\| \leq 1.$$

1. ($\star$) Let P be the set of vertices of a regular hexagon of side length 1. Draw the unit ball graph $G(P)$. What is the largest independent set you can find in $G(P)$? What happens to $G(P)$ if the hexagon has side length 0.5? 2?

2. ($\star\star$) Show that an independent set in $G(P)$ corresponds to a non-overlapping arrangement of spheres of radius 1 centered at points in P. This packing on P can be extended to an infinite periodic packing on the plane.

Thus, the problem of finding a "dense" sphere packing reduces to finding a large independent set on a unit ball graph. In the following problems, we show how to compute an exact maximum independent set (MIS) **in polynomial time** in the special case that G is a two-dimensional unit ball graph lying in a bounded-width strip of the plane (again, note that MIS is NP-hard in general).

3. ($\star\star$) Let A be a directed acyclic graph with weighted vertices. Show that one can compute the longest weighted path in A in polynomial time.

4. ($\star\star$) Let P be a set of points in $\mathbb{R}^2$ *whose y-coordinate is in the range* $[0, k)$. We call a subset of points $B \subseteq P$ a *block* if the points in B lie on the same vertical slab of unit width, i.e.

   $$\lfloor x_1 \rfloor = \lfloor x_2 \rfloor \qquad \text{for all } (x_1, y_1), (x_2, y_2) \in P'.$$

   A subset $B \subseteq P$ is an *independent block* if B is a block and also an independent set of $G(P)$. Show that $|B| \leq 2\lceil 2k/\sqrt{3} \rceil$ for all independent blocks B. Hint: partition the vertical strip into rectangles of width $1/2$ and height $\sqrt{3}/2$.

5. ($\star\star\star$) Show there exists an algorithm for maximum independent set with time complexity

   $$O\left(|P|^{4\lceil 2k/\sqrt{3} \rceil}\right)$$

   on unit disk graphs lying in a region of bounded width k. Hint: use the independent blocks of P to form a weighted directed acyclic graph, then apply (3).