# Languages & Countability

1. Consider an alphabet with m symbols. How many strings are there of length at most k? (If you quickly recognize the answer, rederive it.) How many languages are there that only have strings of length at most k?
2. The power set of a set A, denoted P(A), is the collection of all subsets of A. For example, if , then . Show that for any set A there is no surjective (i.e.onto) function .

Note: Such a surjective f would have the property that for every set , there is an such that f(a) = S. Don’t assume that A is finite.

Hint: Use a diagonalization-like argument.

# Turing Machines

1. Consider the set of Turing machines having a single two-way infinite tape and only two input symbols, blank (b) and one (1). The BB-n game is to find a Turing machine from this class that has exactly n operational (non-halting) states and terminates on the empty string after the maximum number of head shifts. (Another version asks for the maximum number of 1’s on the tape.)

Play the BB-2 game. Let 0 be the initial state and H the halting state. Start with the transitions

0,’b’) = (1,’1’,R)

2. Recall the BB-n game. We restrict ourselves to Turing machines with two tape symbols, blank and 1, and allow the machine a two-way infinite tape. For every n > 0, we have a game where the goal is to find the Turing machine that executes the maximum number of shifts while still halting eventually. We define to be the mapping between the number of operational (non-halting) states and this maximum numbers of shifts.

Consider the following argument (taken from Wikipedia). Find the contradiction and draw the proper conclusion. Prepare to argue that other steps are correct.

1. Let EvalS be the Turing machine that given n 1s on the tape, halts with S(n) 1s on the tape.
2. Let Clean denote a Turing machine cleaning the sequence of 1s initially written on the tape.
3. Let Double denote a Turing machine that given a tape with n 1s it will produce 2n 1s on the tape and then halt.
4. The Turing machines Double | EvalS | Clean can be composed into a single Turing machine having some number of states, say .
5. There is a machine that has exactly states and simply writes out 1s to the tape. Call this machines Create\_n0.
6. Let BadS denote the composition Create\_n0 | Double | EvalS | Clean. This machine has exactly states.
7. After the EvalS part of BadS runs, the tape will have 1s on it.
8. The cleaning phase of BadS will require at least shifts, so BadS, which has states, will use more than shifts.

3. Typically a Turing machine tape is defined to be infinite in only one direction. Suppose, however, that we let it be infinite in both directions. Can this new machine decide or recognize any language that a regular Turing machine cannot? Argue why or why not.

4. Let’s re-examine Turing’s argument that the two-dimensional nature of paper is non-essential. Consider a two-dimensional Turing machine that has symbols written on an grid, which extends infinitely to the right and downward. There is a single read-write head. Now, however, it goes up or down in addition to left and right. Assume that the initial position of the head is the upper left corner of the grid and that the initial string is in the sequence of squares to the right of the initial position. The rest of the grid is blank.

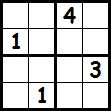
Argue that multitape Turing machines are equivalent to two-dimensional Turing machines.

# Undecidability

1. There are many incarnations of the diagonalization paradox. One famous one defines a barber to be “one who shaves all men, and only those men, who do not shave themselves.” Identify the paradox and illustrate the problem with a grid. Then explain how one can resolve the paradox, exploring as many solutions as you can think of.
2. Consider the language . Is this language decidable? Is its complement?

# NP-Completeness

1. Describe how you might use nondeterminism to simplify the construction of a Turing machine that performs the following tasks.
   1. Decide the language is a binary string
   2. Recognizes the language {<M> | M accepts some string}
2. Construct a SAT-based Sudoku solver for the simplified 4x4 version of the game. (Don’t write out all the clauses. Exploit similarity to be concise.)



We’ll index the rows and columns as 1,2,3,4. For any solution, we’ll define the variable to be true if the number in the rth row and cth column is n and define it to be false otherwise.

1. Create a CNF formula over the variables X that enforces the rules that every square must have a number and that no square can have two numbers.
2. Add clauses that enforce rules of Sudoku : that every number appears exactly once in every row, once in every column, and once in every quadrant.
3. Add the clauses particular to this puzzle.

3. 1-in-3 SAT is the set of 3-CNF formulas such that there is a satisfying assignment that makes exactly one variable in each clause true. For 1-in-3 SAT, we notate a clause as R(x,y,z), which is true if exactly one of the literals x,y,z is true. Show that 1-in-3 SAT is NP-complete by reducing 3-SAT. These steps are recommended.

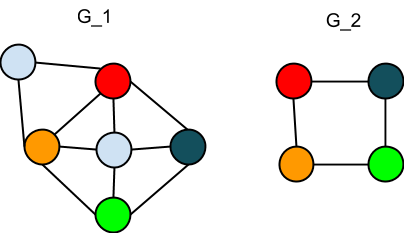
1. Show that is satisfied if and only if there is a way to assign such that is satisfied.
2. Show that is satisfied if and only if there is a way to assign such that is satisfied.
3. Complete the proof.

4. The *subgraph isomorphism problem*can be defined as follows.

**Input**: Two graphs and .

**Decision**: Is a subgraph of ? I.e. is there an injective (one-to-one) function such that for every , we have ?

For example, below is a subgraph of (you may infer a mapping from the colors).



Show that subgraph isomorphism is NP-complete.

5. The *1-forgiven 3-coloring problem* can be defined as follows.

**Input**: A graph .

**Decision**: Is there a is a mapping such that ? In other words, it is possible to color the vertices (red, green, or blue) so that at most 1 edge has vertices of the same color?

Prove that 1-forgiven 3-coloring is NP-complete.

6. Consider the *0-1 integer programming problem.*

**Input**: An matrix and a length vector b with integer entries

**Decision**: Is there x with 0-1 entries such that ?

Prove that 0-1 integer programming is NP-complete. Hint: Reduce 3-CNF SAT to it.

7. Consider the *half 3-CNF satisfiability problem*.

**Input**: An collection of 3 literal clauses (e.g. (

**Decision**: Is there an assignment of the variables so that exactly ½ of the clauses are satisfied?

Prove that half 3-CNF SAT is NP-Complete. Hint Reduce 3-CNF SAT to it. It’s okay to repeat the same clause.

8. The *subset sum problem* can be defined as follows.

**Input**: A multiset of positive integers and an integer

**Decision**: Is there a subset such that ?

An alternative version allows the set to contain negative numbers and defines the positive instances to be those where there is a subset such that

1. Reduce the first version to the second version.
2. Reduce the second version to the problem, given a set of integers , is there a subset whose sum is at most 2015 in absolute value?