CMSC 351: P, NP, Etc. Part 1

Justin Wyss-Gallifent

August 15, 2022

1 Introduction

Here we're going to abstract our knowledge of time complexity to look at classifying problems. In order to do this we need to introduce the concepts of decision problems, Turing machines, etc.

2 Informalities

A great example to consider is the generalization of Sudoku. Starting with an $n^2 \times n^2$ grid with some squares filled with values between 1 and n^2, is it possible to complete the grid with numbers between 1 and n^2 so that each row and each column contains exactly one of each and such that each of the n^2 subsquares of size $n \times n$ contains exactly one of each?

Example 2.1. Classic Sudoku is $3^2 \times 3^2 = 9 \times 9$, the numbers are between 1 and $3^2 = 9$, and there are $3^2 = 9$ subsquares of size 3×3.

■

Example 2.2. Simple Sudoku is $2^2 \times 2^2 = 4 \times 4$, the numbers are between 1 and $2^2 = 4$, and there are $2^2 = 4$ subsquares of size 2×2.

■

Example 2.3. More complicated is $4^2 \times 4^2 = 16 \times 16$, the numbers are between 1 and $4^2 = 16$, and there are $4^2 = 16$ subsquares of size 4×4.

■

If a Sudoko board is partially filled in we can ask whether it can be completely filled in.

If a suggested solution is given it is easy to check if the solution is valid and this check can be performed in an amount of time which grows at a polynomial rate with the size of the board. Can you write an algorithm for this?

However actually coming up with a solution seems to be much harder. It appears that (although nobody has proven this) that the amount of time grows at an exponential rate with the size of the board and that nothing is really any better than actually trying all combinations.

Thus we might informally suggest that checking a given solution is easier than finding a solution.

3 Decision Problems

3.1 What is a Decision Problem?

Definition 3.1.1. A *decision problem* is a problem which taken an input, formally an *instance*, and produces either YES or NO.

We'll often abstractly write Q for a decision problem and I for an instance and then $Q(I) = YES$ or $Q(I) = NO$ depending on whether the result is YES or NO for that instance.

Example 3.1. Q: Given two real numbers x and y, is the sum greater than 100?
I: $x = 50, y = 80$ yields $Q(I) = YES$
I: $x = 10, y = 10$ yields $Q(I) = NO$

■

Example 3.2. Q: Given a natural number n, does n have nontrivial factors?
I: $n = 10$ yields $Q(I) = YES$
I: $n = 5$ yields $Q(I) = NO$

■

Example 3.3. Q: Given a list of integers A, is it sorted?
I: $A = [1, 2, 3]$ yields $Q(I) = YES$
I: $A = [2, 1, 3]$ yields $Q(I) = NO$ ■

Example 3.4. Q: Given a graph G, does it contain a cycle?

■

Example 3.5. Q: Given a set A is there a subset which adds to 0?
I: $A = \{1, 4, -3, 5, -1\}$ yields $Q(I) = YES$
I: $A = \{1, 4, 5, 3, -2\}$ yields $Q(I) = NO$

■

3.2 Rephrasing as Decision Problems

Many types of problems which are not decision problems can be rephrased as decision problems. The rephrased problem will not be exactly the same but will carry over the same notion in a computationally equivalent sense.

Example 3.6. Non-decision problem: Find a solution to a partially filled Sudoku board.

This can be rephrased as:

Q: Given a partially filled Sudoku board, does a solution exist?

Observe that deciding if a solution exists essentially seems to be as difficult as finding one.

■

Example 3.7. Non-decision problem: For a graph G, find the shortest path between two vertices s and t.

This can be rephrased as:

Q: Given two vertices s and t and a length k, is there a path of length less than or equal to k between s and t?

Observe that deciding if there is a path of length less than or equal to k essentially seems to be as difficult as finding one.

■

3.3 Proofs and Witnesses

Definition 3.3.1. Given a decision problem and an instance if the answer is YES then a *proof*, or witness, or *certificate* is essentially proof that the answer is yes.

Informally we'll use the term *invalid witness* to refer to something which doesn't work.

Example 3.8. Q: Given a set, is there a subset which adds to 0?
I: $\{1, 4, -3, 5, -1\}$ has $\{4, -3, -1\}$ as a witness.

■

Note 3.3.1. If we have access to a valid witness for a decision problem and instance then we can say that the answer is YES. However if we don't have access to a witness or if we have an invalid witness then we cannot say that the answer is no.

Example 3.9. For the decision problem: Given a set, is there a subset which adds to 0?
For the instance $\{1, 4, -3, 5, -1\}$ if we present $\{1, 4\}$ then this is an invalid witness and we know nothing.

■

4 Turing Machines

4.1 Basics

Definition 4.1.1. A *Turing machine* consists of:

- An infinitely long (in both directions) tape divided into cells. Each cell is either blank or contains one of a finite set of symbols.
- A read-write head which points to one cell at a time. It can read the cell or write to it, or it can move one cell left or right.
- A current state taken from a finite set of possible states and which is initialized at the start.
- A finite table of instructions.

A Turing machine is given a starting state and then it proceeds to run according to its state, its set of rules, and the tape. It may then stop operating at some ending state.

While this may seem particularly simple in reality every single algorithm that we generally think of and work with can be modeled by a particular Turing machine.

4.2 Deterministic

Definition 4.2.1. A *deterministic Turing machine* (DTM) is a Turing machine which has a single course of action for any combination of it's internal state state and the input/memory. In such a case one starting state will result in one ending state.

Note 4.2.1. All of the thinking we've been doing in this class is based around the idea of a DTM.

4.3 Non-Deterministic

Definition 4.3.1. A *non-deterministic Turing machine* (NTM) is a Turing machine which can have more than one course of action for any combination of its internal state and the input/memory. In such a case one starting state can (but doesn't have to) result in many ending states.

Note 4.3.1. Understand that NTM should be treated as a thought experiment used to study computing rather than a specific machine on which algorithms are actually run.

Example 4.1. If we are trying to solve the decision problem as to whether there is a subset of $\{-1, -2, 3, 4\}$ then a DTM needs to check all possible subsets one by one whereas a NTM can try them simultaneously.

This means it could add all subsets simultanously, or it could compare all subsets against a different set simultaneously, or whatever. The key point is that these operations happen simultaneously.

■

Example 4.2. If a DTM is doing a breadth-first search on a graph then at a given vertex it must check each adjacent vertex one by one in some order. An NTM can check them all simultaneously and the result is many ending states, one corresponding to each branch.

■

Example 4.3. If a DTM is trying to factor a number by brute force then it must try every factor. An NTM can try them all simultaneously and the result is many ending states, one corresponding to each result.

■

Time complexity can of course then change drastically when we are thinking about DTMs v NTMs.

Example 4.4. If a problem of size n involves a star graph with $n+1$ vertices (one central vertex connected to n other vertices, none of which are connected to each other) and each of the n outer vertices require a $\Theta(1)$ calculation then a DTM renders the problem $\Theta(1+n) = \Theta(n)$ as it has to check the n vertices one by one but a NTM renders the problem $\Theta(1+1) = \Theta(1)$ as it can check them simultaneously by branching.

■

A NDM can obviously emulate a DTM because it can be instructed to simply follow one course of action. A DTM can to some degree emulate a NDM but only in a breadth-first sense. It cannot follow all possibilities to all depths simultaneously.

5 Thoughts, Problems, Ideas

1. Prove that if an algorithm has time complexity $T(n) = n!$ then the algorithm does not run in polynomial time.

2. Consider this decison problem:

   YES $\vee$ NO: Given a set S with n elements, is there a subset which adds to zero?

   For each of the following inputs, first answer YES or NO. If the answer is YES, give an example of a valid witness and an invalid witness.

   (a) $S = \{5, 1, 8, -2, 10, -2, 100, 7, -2\}$

   (b) $S = \{-10, 20, 1, 2, 3, 6, 80, -11, 4\}$

   (c) $S = \{-2, -1, 4\}$

3. Consider this decison problem:

   YES $\vee$ NO: Does the integer $n \geq 2$ have a factor which is not 1 or itself?

   For each of the following inputs, first answer YES or NO. If the answer is YES, give an example of a valid witness and an invalid witness.

   (a) $n = 100$

   (b) $n = 97$

   (c) $n = 51$

4. Consider this decison problem:

   YES $\vee$ NO: Does the graph with n vertices labeled 0 through $n - 1$ and represented by a given adjacency list have a cycle?

   For each of the following inputs, first answer YES or NO. If the answer is YES, give an example of a valid witness and an invalid witness.

   (a) $[[1, 3], [0, 2], [1, 3], [0, 2]]$

   (b) $[[1, 2, 3], [0], [0], [0]]$

   (c) $[[1, 2], [0, 3, 4], [0], [1], [1]]$