CIS 511: Theory of Computation **Feb 14, 2017**

Lecture 10: Time Complexity, Continued

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

Asymptotics

Definition 10.1 *An algorithm with running time $t(n)$ takes no more than $O(t(n))$ steps on any input of size n. This is sometimes called* ***worst-case complexity****.*

Recall the class DTIME($t(n)$) is the set of languages L for which there exists a single-tape Turing machine that decides L in $O(t(n))$ time.

What about a multi-tape machine? We have seen that the language $L = \{ww^R | w \in \{0,1\}^*\}$ can be decided in $O(n)$ on a two-tape TM but is in $\Omega(n^2)$ on a single-tape machine. This language is in DTIME(n^2) because we consider only single-tape machines.

The Class P

Definition 10.2 *The class P is equal to $\bigcup_{k \geq 0} DTIME(n^k)$. That is, the class of languages decided in polynomial time on a single-tape deterministic Turing machine.*

Why is P an important class? First, a problem being in P implies there is a method of solving that problem without enumerating all possible solutions, which would typically require exponential time. Second, P is a robust class with a lot of nice compositional properties. The sum, product, and composition of polynomials is a polynomial. This means that if we have a polynomial-time subroutine, if we call it a polynomial number of times, our algorithm is still polynomial-time. Thirdly, we have the Extended Church-Turing Thesis, which states that every reasonable model of computation can simulate any other reasonable model of computation with only polynomial-time slowdown. This implies that the class P is invariant under models of computation (numbers of tapes, modern computers, quantum computers). We don't actually know that if is true, but there is no proof yet that it isn't correct.

Some languages we know are in P:

- $PATH = \{\langle G, s, t\rangle | G$ *a digraph* s, t *vertices, there is an* $s - t$ *path*$\}$.

  If G has m vertices, then there may be m^m paths to explore. But, we can run BFS from G starting at s. If there exists a path, we eventually reach t. As a side note, BFS requires m bits of space, keeping track of whether we have visited a node or not. Open question: can we do better?

- $UPATH$ *- the same, but for undirected paths.*

- *GCD: given $a, b \in \mathbb{N}$, find the greatest common divisor of a and b. The input size is $\log a + \log b$. The Euclidean algorithm is actually in P.*

The jury is still out on:

- $FACTORING = \{\langle N, K\rangle | N\ has\ a\ factor\ strictly\ between\ 1\ and\ K\}$. *This is a sufficient subroutine to factor an integer, and if this is in P, then integer factorization is as well. Note that the language $PRIME$, which determines if a number is prime is actually in P, but is not sufficient to do factorization (as far as we know).*

The Class NP

Definition 10.3 *A non-deterministic Turing machine is called a* ***decider*** *if it halts on every branch.*

Definition 10.4 *The running time of a non-deterministic Turing machine on an input x is the maximum number of steps taken on any branch.*

Definition 10.5 *A language L is in $NTIME(t(n))$ if there is an NTM M that decides L in time $O(t(n))$ on all inputs of length n.*

Let M be an NTM that runs in time $f(n)$. Then on all of its branches, M halts in time at most $f(n)$ on all of its branches. We showed earlier how to make a deterministic TM that simulates an NTM. The branching tree has height at most $f(n)$, and the number of possible transitions at each step was some number B. Then the number of nodes in the tree is at most $b^{f(n)}$. This is no longer polynomial. To our knowledge, simulating a non-deterministic decider on a deterministic decider results in an exponential blowup in running time.

We therefore can say that $NTIME(f(n)) \subseteq DTIME(2^{O(f(n))})$

Definition 10.6 *The class NP is equal to $\bigcup\limits_{k\geq 0} NTIME(n^k)$. That is, the class of languages decided in polynomial time on a non-deterministic Turing machine.*

Some languages in NP:

- $SAT = \{\phi | \phi\ is\ a\ Boolean\ formula\ with\ some\ satisfying\ assignment\}$
- *TSP: given a weighted graph, does there exist a tour with cost at most B? We can get around the branching factor being a function of the input by introducing a $\log n$ depth factor at each step to identify each vertex with a binary index.*