Alex Hubers
CSC:341 Spring 2024

Lecture Notes #11A

#1 A Review of the Worst Mathematical Notation I can Think Of

Definition 1. *Let $g : \mathbb{N} \to \mathbb{R}^+$ be a function. We say that $f \in \mathcal{O}(g)$ if positive integers c and n_0 exist such that, for every integer $n \geq n_0$,*

$$f(n) \leq cg(n)$$

Observe that $\mathcal{O}(g)$ is the set of all such f. When $f \in \mathcal{O}(g)$, we say that g is an asymptotic upper bound for f.

Note that it is more common in the literature to write that $f(n) = \mathcal{O}(g(n))$, but this is nonsense. Call me crazy but I have certain expectations of equality—for one, that equality forms an equivalence relation. Yet you will observe that the entity on the left (f) is a function and the entity on the right ($\mathcal{O}(g(n))$) is a set of functions—hardly making this equality relation a reflexive one. Further, I expect equivalence to be transitive. Observe the fruits of this relation being a transitive one:

Claim 1. *n^2 equals n^3.*

Proof. Well, $n^2 = \mathcal{O}(n^3)$, and $\mathcal{O}(n^3) = n^3$, so by transitivity $n^2 = n^3$. □

Anyway, please refrain from writing $f = \mathcal{O}(g)$ and instead write $f \in \mathcal{O}(g)$. Here is an example proof.

Claim 2. *Let $f(n) = 3n + 1$. Then $f \in \mathcal{O}(n)$.*

Proof. Consider $c = 4$ and $n_0 = 1$. Then, for all $n \geq 1$, we have

$$3n + 1 \leq 4n$$

□

This may be the first and last proof of this sort we give in this class. In practice, one can do away with the formalities. We all know by now that $3x^3 + 2x^2 + x + 1$ is $\mathcal{O}(n^3)$, and so forth. Further, in this class, it is often enough to just make sure that a given function is polynomial or non-polynomial.

Also note that there are other notations for best-case and average-case. They're in Sipser. I will define them as I need them.

#2 TM Complexity

Back to the point.

Let M be a deterministic TM decider. The running time or time complexity of M is the function $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of steps that M uses on an input of length n.

And an example:

Let M_1 decide $A = \{0^k1^k \mid k \geq 0\}$ as follows, with input string w.

1. M_1 scans across the tape and rejects if a 0 is found to the right of a 1.
2. Repeat the next step if both 0s and 1s remain on the tape:
   (a) Scan across the tape, crossing off a single 0 and a single 1.
3. If 0s still remain after all the 1s have been crossed off, or vice versa, reject. Otherwise, if all cells are marked, accept.

In class, let's "count" how many steps this TM takes as a function of its input length.

1. In stage 1, we do a linear sweep of length n.
2. In stage 2(a), we do a (worst-case) linear scan of the input tape to mark off a 0 and 1. This can occur at most $\frac{n}{2}$ times.
3. Finally, we do a last sweep to see if there are any 0s or 1s left. This is linear.

We end up with a worst-case of $n + \frac{n}{2}n + n$ count, which is $\mathcal{O}(n^2)$.

This leads into our definition of "complexity classes".

Definition 2. *Let $t : \mathbb{N} \to \mathbb{R}^+$. Define the time complexity class, $\text{TIME}(t)$, to be the collection of all languages that are decidable by an $\mathcal{O}(t)$-time Turing Machine.*

E.g., M_1 decides A in $\mathcal{O}(n^2)$ time, and so $A \in \text{TIME}(n^2)$.

#3 Complexity Relationships

Here is the big claim:

Claim 3. *All "reasonable" deterministic computational models are polynomially equivalent.*

A proof would mean listing all such "reasonable" models and showing that they can be converted amongst eachother in polynomial time. This is painful. But let's consider one example.

Claim 4. *Let t be a function where $t(n) \geq n$. Then every $t(n)$ time multitape Turing machine has an equivalent $\mathcal{O}(t^2(n))$ time single-tape Turing machine.*

Proof Idea. Recall how we showed k-tape Turing machines equivalent to single-tape Turing machines—we put all of the tapes onto just a single tape and then used special characters to denote the positions of each tape head. We can show (in detail omitted) that simulating one step of the multi-tape machine uses at most $\mathcal{O}(t(n))$ steps on the single tape machine. Hence the total time used is $\mathcal{O}(t^2(n))$. □

#4 The Class P

Here is our first major complexity class.

Definition 3. *P is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine. In other words,*

$$P = \bigcup_k \text{TIME}(n^k)$$

So P is the set of all languages solvable in *some* polynomial time–which we deem to be "reasonable".

Some examples of languages in P.

$$\begin{aligned}\textsc{path} =& \{\langle G, s, t\rangle \mid \text{ Directed graph } G \text{ has a path from } s \text{ to } t\} \\ \textsc{relprime} =& \{\langle x, y\rangle \mid x \text{ and } y \text{ are relatively prime}\}\end{aligned}$$

And here's a proof for PATH.

Claim 5. PATH $\in P$.

Proof. Let M decide PATH as follows, with M taking input $\langle G, s, t\rangle$:

1. Place a mark on node s.
2. Repeat the following until no additional nodes are marked:
   (a) Scan all the edges of G. If an edge (a, b) is found going from a marked node a to an unmarked node b, mark node b.
3. If t is marked, accept. Otherwise, reject.

□

Now, for complexity analysis:

1. Step 1 is a linear pass through the list of vertices in G.
2. Step 2(a) will repeat at most $|V|$ times. Each iteration of 2(a) can be implemented in polynomial time (we don't care how).
3. Step 3 is clearly a linear pass.

Note that we are ignoring details—the point is that the runtime should be the costs of steps 1-3 and each of these steps is polynomial. Therefore, PATH is polynomial.