11. COMPLEXITY ACROSS COMPUTATIONAL MODELS

Mihai-Valentin DUMITRU
mihai.dumitru2201@upb.ro

November 2024

In the previous lecture, we started our investigation of the field of Complexity Theory and set up some important definitions about what computational resources are, how to characterize a TM's performance using *worst-case complexity* and how to abstract away unimportant constants using *asymptotic notations*.

We insisted on the fact that our main goal is to classify and compare *problems*. We're not interested just in how efficient a particular TM is; we want to know what's the best machine that could solve the problem.

At this point, you probably have many questions and doubts. Perhaps through the many examples and arguments presented, you can accept the Church-Turing thesis: that what Turing Machines can solve is precisely what is algorithmically solvable. But Turing Machines are very simplistic models of computation. For the simplest tasks (like the lab exercises) we had to write huge transition tables; the resulting machines ended up making lots and lots of transitions even on small inputs.

Now that we start exploring *efficiency*, the usefulness of Turing Machines as a model of computation may seem dubious. Indeed, we shall see that multitape Turing Machines can solve some problems *asymptotically faster*. Single-tape TMs can simulate these, but slower. However, the simulation only has a *polynomial* "slowdown". Whatever a 2-tape TM can do in $O(t(w))$ time, a single-tape TM can do in $O(t(w)^2)$ time.

We will argue for the case of lumping together all polynomials (n, n^2, n^3, n^8 etc.) and formulate the definition of a (very famous) complexity class: P.

Let's revisit the problem from the previous lecture, `PALINDROME`, and reanalyze the difference between the single-tape and 2-tapes solution.

1 The (limited) power of an extra tape

Recall our single-tape TM that checked if a binary string is a palindrome:

	0	1	$\square$
$start$	$remember0, \square, \rightarrow$	$remember1, \square, \rightarrow$	$Y, \square, -$
$remember0$	$remember0, 0, \rightarrow$	$remember0, 1, \rightarrow$	$expect0, \square, \leftarrow$
$remember1$	$remember1, 0, \rightarrow$	$remember1, 1, \rightarrow$	$expect1, \square, \leftarrow$
$expect0$	$reset, \square, \leftarrow$	$N, 1, -$	$Y, \square, -$
$expect1$	$N, 0, -$	$reset, 1, \leftarrow$	$Y, \square, -$
$reset$	$reset, 0, \leftarrow$	$reset, 1, \leftarrow$	$start, \square, \rightarrow$

In the last lecture, we briefly analyzed its behavior and concluded that its running time is $O(n^2)$ – an upper bound. It can be shown[†] that **any** single-tape TM solving `PALINDROME` has a running time of $\Omega(n^2)$ – a lower bound. The quadratic solution is optimal, we can't go asymptotically faster.

But recall the 2-tape solution that solves the same problem. Its running time is $O(n)$ – asymptotically faster!

In the third lecture, we saw that adding a tape did not get us any *computability* power; a k-tape TM can solve precisely the same problems as a single-tape TM. But the extra tape did pay off in terms of efficiency.

The question now is: how much? Can a 2-tape TM be arbitrarily faster than a single-tape one? Would adding more tapes be even better?

The answer is, surprisingly, no. A quadratic improvement is all we can achieve by adding extra tapes (regardless of the number of tapes).

[†]It is out of scope for this course, but see section 4.

We have actually proved a weaker result in a previous lecture: that we can't get better than cubic improvement. Recall Theorem 6.1 from the lecture on multitape machines: we claimed that each k-tape TM can be simulated by a single-tape one and proved it by describing the simulation.

The single-tape machine kept the contents of all k tapes one after the other, separated by a special symbol, $\diamond$. To simulate one transition, the TM had to first go through the tape-contents and seek the k underlined symbols indicating the positions of the k tape heads. After all were found, the TM had to go back through the whole tape, find the heads again and perform the necessary changes; sometimes we had to "extend" one of the simulated tapes, by shifting all the following symbols one cell to the right.

Let's label the run-time of the simulated k-tape machine on input w with $t(w) \geq |w|$[‡]. Because the machine makes $t(w)$ transitions, it can't write more than $t(w)$ cells; so the total content of the simulated tapes is less than $k(t(w)+1)+1$ symbols (including the $\diamond$s). For each simulated transition, the single-tape machine would go through the tape symbols twice (to search all the currently-read symbols, then to modify them), so roughly $2kt(w)$. In the worst case, each of the k tape-contents would need to be expanded, which means shifting symbols to the right of that tape's head by one. That is at most around $k^2t(w)$ transitions necessary[§].

There are $t(w)$ transitions to simulate; without getting very specific, we can see that the total number of transitions is a function in $\Theta(t(w)^2)$.

Notice that it doesn't matter how large k is (but it's always a constant, which doesn't depend on the input length).

The multitape Turing Machine did help speed things up, but it didn't provide us with unbounded power. In practice, such a difference (between n and n^2) would be very important; we don't mean to imply that it's negligible. But in order to build an elegant and robust classification of problems, we will have to increase our level of abstraction past the asymptotic notations.

In the previous lecture we concluded that PALINDROME $\in$ **DTIME**(n); but if we only looked at single-tape solutions, we could only prove the weaker statement: PALINDROME $\in$ **DTIME**(n^2).

This is problematic, because it requires us to always mention precisely which model we are talking about when classifying a problem: a single tape machine, or a multitape one?

2 Other models of computation

There are other models of computation that can be used to produce solutions that are asymptotically faster than what is possible on single-tape or multitape Turing Machines. One example is the Random Access Machine (RAM) model, which is closer in functioning to a modern day, tangible computer: it has an arbitrary amount of instantly accessible registers, which can store integers of arbitrary sizes, which can be added or subtracted in constant time. But any RAM program that takes $t(w)$ steps, can be simulated by a multitape TM in time $O(t(w)^3)$ (or by a single-tape machine in time $O(t(w)^6)$).

In fact, for many alternative models of computation it is true that their runs can be simulated by a single-tape TM with only a polynomial slowdown[††].

This leads to the following two key ideas:

1. the optimal time to solve a problem can vary *asymptotically* based on which model of computation is used
2. all "reasonable"[‡‡] models of computation can simulate each other with only a polynomial slowdown

We want to talk about the complexity of problems without having to narrow-down precisely on a particular model of computation. We will use *polynomiality* as the central criterion.

[‡]We can prove that the only sublinear time-complexity possible is constant time, which is not very interesting; the single-tape machine would also run in constant time.

[§]Obviously, we don't have to shift $t(w)$ symbols for each simulated tape; the tapes with a higher index number, that are placed more to the right of the single tape, will require shifting fewer symbols than the lower-numbered tapes. This showcases perfectly the strength of asymptotic notations: we can be quite sloppy and still reach sane, rigurous conclusions. What we care about in this case is that each shift requires $\Theta(t(n))$ transitions.

[††]Two very important, possible exceptions involve nondeterminism and quantum computation. It is not known whether a single-tape Turing Machine can simulate a nondeterministic Turing Machine with a polynomial slowdown; similarly, it is not known whether a single-tape Turing Machine can simulate a quantum Turing Machine with a polynomial slowdown. In both cases, it is widely believed that the answer is "no". We'll return in detail to the idea of nondeterminism in a future course; unfortunately, quantum computation is out of scope for us.

[‡‡]The term "reasonable" is informal here, but its purpose is to exclude nondeterministic and quantum models.

3 The class P

If we treat all polynomials the same, then we can abstract away the specific model of computation. In other words, if we say that a problem is "decidable in polynomial time", then we don't need to specify whether we're referring to a single-tape TM, a multitape TM, a RAM program etc.

The second big advantage of polynomials, is the fact that they represent a *closed-class* of functions under many operations: addition, subtraction, multiplication etc. If we add, subtract, multiply etc. two polynomials, we get a polynomial. We can *combine* polynomial solutions to get another polynomial solution.

For these reasons, we will focus on the class of problems decidable in polynomial time. This class is known as P and we shall consider it our model for "tractable problems"[§§] (in contrast, a problem that is not in P is "intractable", it's not practically possible to solve it).

Definition 11.1.

$$P = \bigcup_{k \in \mathbb{N}} \mathbf{DTIME}(n^k)$$

4 References and further reading

In 1965, Richard Stearns and Juris Hartmanis published the article: *"On the computational complexity of algorithms"* [1], which is one of the pioneering works of complexity theory. Among other results, they prove that a single-tape Turing Machine can simulate a multitape Turing Machine with a square slowdown.

In the 1966 paper *"Two-tape simulation of multitape Turing machines"* [2], Hennie and Stearns show that for any $k \geq 3$, a 2-tape TM can simulate a k-tape TM with only a logarithmic slowdown.

These results are neatly presented in a uniform fashion in the textbook *"Computational complexity: a modern approach"* [3], by Sanjeev Arora and Boaz Barak, in part one, chapter 1. In general, the textbook is a good (and way more expansive) companion to our study of computational complexity.

The 1984 article *"Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines"* [4] by Wolfgang Maass proves (among other things) that a single-tape TM can't solve `PALINDROME` in $o(n^2)$ time. For an example of applying the technique of *crossing sequences* to prove this fact, check out this lecture:
`https://www.eecs.yorku.ca/course_archive/2008-09/W/6115/palindrome.pdf`

For a definition of the Random Access Machine model, together with a polynomial simulation of it on a multitape Turing Machines, see Christos Papadimitriou's textbook *"Computational Complexity"* [5], chapter 2, section 6 (particularly, the simulation is in the proof of theorem 2.5).

Bibliography

[1] Juris Hartmanis and Richard E Stearns. "On the computational complexity of algorithms". In: *Transactions of the American Mathematical Society* 117 (1965), pp. 285–306.

[2] Fred C Hennie and Richard Edwin Stearns. "Two-tape simulation of multitape Turing machines". In: *Journal of the ACM (JACM)* 13.4 (1966), pp. 533–546.

[3] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[4] Wolfgang Maass. "Quadratic lower bounds for deterministic and nondeterministic one-tape Turing machines". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. 1984, pp. 401–408.

[5] Christos Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[§§] This is a very contentious statement; a problem whose optimal solution is in $DTIME(n^{1000})$ for a multitape machine doesn't seem to match our intuition of what is tractable. Indeed, researchers in complexity theory have proposed alternative formal classes to capture the intuitive notion of "tractable". We are just taking the first steps in this direction.