The Time Hierarchy Theorem

1 Sample consequences of the Time Hierarchy Theorem

The above "intuitive" statement of the Time Hierarchy Theorem is imprecise; it has scare-quotes around the "any" time bound, and it doesn't say mathematically what "slightly more/less" means. Before we get into precise details, though, let us give some example consequences of the theorem (which *do* follow from the precise versions, as you will see).

Example (THT Example 1)**.** The Time Hierarchy Theorem applied with $f(n) = 2^n$ implies the result mentioned in the last chapter, that there is a language $L \in \mathsf{TIME}(3^n)$ such that $L \notin \mathsf{TIME}(1.1^n)$. Recalling that $\mathsf{TIME}(1.1^n)$ is a *set* of languages, and $\mathsf{TIME}(3^n)$ is a superset of $\mathsf{TIME}(1.1^n)$, we can write the Time Hierarchy Fact in a couple of ways using set-theory notation:

$$\mathsf{TIME}(1.1^n) \subsetneq \mathsf{TIME}(3^n); \quad \text{or,} \quad \exists L \in \mathsf{TIME}(3^n) \setminus \mathsf{TIME}(1.1^n).$$

Example (THT Example 2)**.** In the full version of the Time Hierarchy Theorem, taking $f(n) = n^{1.5}$ will allow us to conclude that

$$\mathsf{TIME}(n) \subsetneq \mathsf{TIME}(n^2);$$

in other words, there is a language solvable in quadratic time but not solvable in linear time. Using $f(n) = n^{2.5}$, we can get

$$\mathsf{TIME}(n^2) \subsetneq \mathsf{TIME}(n^3);$$

that is, there is a language solvable in cubic time but not solvable in quadratic time. More generally, using $f(n) = n^{c+.5}$ for any constant $c \in \mathbb{N}^+$, we can get

$$\mathsf{TIME}(n^c) \subsetneq \mathsf{TIME}(n^{c+1}).$$

This last example tells us that

$$\mathsf{TIME}(n) \subsetneq \mathsf{TIME}(n^2) \subsetneq \mathsf{TIME}(n^3) \subsetneq \mathsf{TIME}(n^4) \subsetneq \mathsf{TIME}(n^5) \subsetneq \cdots$$

We have a *hierarchy* of time-complexity classes here, each one containing strictly more languages than the previous one. This kind of thing is why it's called the *Time Hierarchy Theorem*.

2 THT: many tiers of quality

There is no one standard formulation of the "Time Hierarchy Theorem". The reason is that you can seek different tiers of quality (*455 Slang*) in terms of:

(a) how badly you want to allow "any" time function $f(n)$;

(b) how "slight" is "slightly" when you prove L is solvable in "slightly more" than $f(n)$ time, but not in "slightly less" than $f(n)$ time.

The harder you work in your proof, the higher quality tier "Time Hierarchy Theorem" you can get. The thing is, it's already fairly challenging just to get a "C-tier" Time Hierarchy Theorem, and it's not clear the effort/reward tradeoff for getting higher-tier versions is favorable for a course like 15-455. We will more or less end up *proving* a "B-tier" version, *stating* an "A-tier" version, and *alluding to* an "S-tier" version (the highest tier).

Here is an "A-tier" version, which is a (slightly worse) version of what you will see in most textbooks:

Theorem (A-tier quality Time Hierarchy Theorem). *Let $f(n)$ be a clockable function with $n \log n \leq O(f(n))$. Then there exists a language L such that*

- $L \in \mathsf{TIME}(f(n) \cdot \log f(n))$;
- $L \notin \mathsf{TIME}(\frac{f(n)}{\log f(n)})$.

This version is more than enough to give the conclusions discussed in Example (THT Example 1) and Example (THT Example 2). For example, as discussed in the last chapter, $f(n) = n^{2.5}$ is a perfectly normal and clockable function, and $n \log n \leq O(n^{2.5})$. Note also that $\log f(n) = 2.5 \log n$. Thus Theorem (A-tier quality Time Hierarchy Theorem) tells us that

$$\mathsf{TIME}(\tfrac{n^{2.5}}{\log n}) \subsetneq \mathsf{TIME}(n^{2.5} \log n),$$

which is a very fine result. Since $n^2 \le O(\frac{n^{2.5}}{\log n})$ and also $n^{2.5}\log n \le O(n^3)$, we can immediately derive the weaker but simpler-looking

$$\mathsf{TIME}(n^2) \subsetneq \mathsf{TIME}(n^3).$$

Similarly, we can get the whole hierarchy

$$\mathsf{TIME}(n) \subsetneq \mathsf{TIME}(n^2) \subsetneq \mathsf{TIME}(n^3) \subsetneq \mathsf{TIME}(n^4) \subsetneq \mathsf{TIME}(n^5) \subsetneq \cdots$$

Exercise. Let $1 < a < b$ be any real constants (e.g., $a = 1.1$, $b = 3$) . Show that $\mathsf{TIME}(a^n) \subsetneq \mathsf{TIME}(b^n)$.

Hint. Use Theorem (A-tier quality Time Hierarchy Theorem) with $f(n) = c^n$, where c is some rational number strictly between a and b.

Remark. Although the "A-tier" Time Hierarchy Theorem is very sharp, it can still be improved. It's a little-known fact that there are even stronger ("S-tier") versions that allow you to show, for example, that

$$\mathsf{TIME}(n^3) \subsetneq \mathsf{TIME}(n^3 \cdot (\log n)^{.0001}).$$

This is really angels-dancing-on-the-head-of-a-pin territory though, and arguably not too insightful. Remember, these running times are on the 1-tape TM model, where we already have moderate weirdness like **Palindromes** requiring $O(n^2)$ time (whereas it's $O(n)$ for more usual algorithmic models). So it's really splitting hairs to be concerned with logarithmic running time factors for 1-tape TMs, let alone sub-logarithmic factors.

Proving the A-tier Time Hierarchy Theorem requires too many hacks/tricks, and we will be quite content with the following version, which we *will* prove:

Theorem (B-tier quality Time Hierarchy Theorem). *Let $f(n)$ be a clockable function with $n^2 \log n \le O(f(n))$. Then there exists a language L such that*

- $L \in \mathsf{TIME}(n^2 \cdot f(n) \cdot \log f(n))$;
- $L \notin \mathsf{TIME}(\frac{f(n)}{\log f(n)})$.

The two differences here are: (a) $f(n)$ has to be at least $n^2 \log n$, as opposed to just $n \log n$; and, (b) we have an extra n^2 factor in the upper bound on L's time complexity. Still, these things are not too bad. Suppose we take $f(n) = n^{c+.5}$, where $c \ge 2$ is an integer constant (these are clockable functions). Then we conclude

$$\mathsf{TIME}(\tfrac{n^{c+.5}}{\log n}) \subsetneq \mathsf{TIME}(n^{c+2.5} \cdot \log n) \quad \Longrightarrow \quad \mathsf{TIME}(n^c) \subsetneq \mathsf{TIME}(n^{c+3}).$$

So we can get a "hierarchy" like

$$\mathsf{TIME}(n^2) \subsetneq \mathsf{TIME}(n^5) \subsetneq \mathsf{TIME}(n^8) \subsetneq \cdots,$$

which still nicely illustrates the sentiment that "more time allows algorithms to solve more languages". Also:

Exercise. Show that the B-tier Time Hierarchy Theorem is still enough to derive $\exists L \in \mathsf{TIME}(3^n) \setminus \mathsf{TIME}(1.1^n)$, and more generally $\mathsf{TIME}(a^n) \subsetneq \mathsf{TIME}(b^n)$ for any constants $1 < a < b$.

Remark. These two versions of the Time Hierarchy Theorem merely say that a language L satisfying the properties *exists*. Another way to improve these results is to name a *specific* language satisfying the properties. For example, referring to the previous exercise, it would be nice to know that $L = \textbf{BoundedAccepts}_{2^\bullet}$ works; i.e., that

- $\textbf{BoundedAccepts}_{2^\bullet} \in \mathsf{TIME}(3^n)$;
- $\textbf{BoundedAccepts}_{2^\bullet} \notin \mathsf{TIME}(1.1^n)$.

We will indeed show this at the end of the chapter.

Before we prove the B-tier version of the Time Hierarchy Theorem, we will warm up by proving D-tier and C-tier versions. In fact, all of these versions of the Time Hierarchy Theorem can be seen as extensions of *Turing's Theorem* on the unsolvability of the Halting/TM-acceptance problem. So we will review that first.

3 Turing's Theorem

Let us recall Turing's proof of the undecidability of the **Accepts** problem. (As we discussed in earlier chapters, we may assume without loss of generality that all TMs mentioned are standard-alphabet TMs.)

Theorem. *(Turing, 1936.) The language* ***Accepts*** *is undecidable; in other words, there does not exist any decider TM A for* ***Accepts****.*

Proof. The proof is by the "diagonalization" method. We assume for the sake of contradiction that TM code A deciding **Accepts** exists. We now describe "diagonalizing TM code" D. In brief:

$D(\langle M \rangle)$ runs $A(\langle M, \langle M \rangle \rangle)$ and does the opposite.

In more details:

- The TM code D first interprets its input string in $\{0,1\}^*$ as the encoding of a TM M. (Recall that by GUIDO, *every* possible string input to D counts as a TM. Garbage strings are interpreted as some default standard TM, say the one that immediately rejects every input.)
- D then copies its input string over, adding a little punctuation if necessary, thereby arranging for the string $\langle M, \langle M \rangle\rangle$ to be on its tape. This will be an input for the TM A.
- D now enters into a subroutine that contains the exact "code" of A, except with A's accept/reject states swapped.

Notice that D is definitely a decider TM: it halts on every input, because A (by assumption) is a decider and thus halts on every input.

We can now obtain a contradiction by considering the question of whether $D(\langle D \rangle)$ accepts or not. When D is given $\langle D \rangle$ as input, it ends up running $A(\langle D, \langle D \rangle\rangle)$ and doing the opposite. But by assumption, $A(\langle D, \langle D \rangle\rangle)$ gives the correct answer to the question, "Does $D(\langle D \rangle)$ accept?" So we have a contradiction; if $D(\langle D \rangle)$ accepts then it means that $D(\langle D \rangle)$ does not accept, and vice versa.

Therefore we conclude that the initial assumption was false; the TM A deciding **Accepts** cannot exist. □

4 D-tier quality Time Hierarchy Theorem

In the proof of Turing's Theorem, $D(\langle M \rangle)$ took a hypothetical **Accepts**-decider A, ran it on $\langle M, \langle M \rangle\rangle$, and did the opposite. Now, this is a fantasy, because we ultimately know there *is* no **Accepts**-decider A. However, as we saw last chapter in Theorem (**??**), there *is* a **BoundedAccepts**$_{2^\bullet}$ decider, call it B, with running time $O(3^n)$. (Just to recall, this B takes as input $\langle M, w \rangle$, computes $2^{|w|}$, then uses an alarm-clocked universal TM to see if $M(w)$ accepts within $2^{|w|}$ steps.) What happens if we just repeat Turing's proof using B?

Theorem (D-tier quality Time Hierarchy Theorem)**.** *There exists a language L such that*

- $L \in \mathsf{TIME}(3^n)$*;*
- *there is no (standard-alphabet) TM M deciding L with running time $T_M(n) \le 2^n$.*

Proof. We will say what L is in a roundabout way. We will first describe a decider TM called D. Then we will say,

L is whatever language is decided by D; that is, $L = \{x : D(x) \text{ accepts}\}$.

(As you will see, the L we produce might be called something like **Doesn'tBoundedAcceptSelf**$_{2^\bullet}$.)

As suggested, the proof is very similar to Turing's, but using the decider B for **BoundedAccepts**$_{2^\bullet}$. TM algorithm D takes as input the description of a TM M and does the following:

$D(\langle M \rangle)$ runs $B(\langle M, \langle M \rangle\rangle)$ and does the opposite.

Let L be the language decided by D. The algorithm D basically copies its input string (time: $O(n^2)$) and then does B, which as we know takes time $O(3^n)$. Thus the running time of D is $O(3^n)$, so $L \in \mathsf{TIME}(3^n)$.

To complete the proof, suppose for the sake of contradiction that there is some (standard-alphabet) TM M deciding L with running time $T_M(n) \leq 2^n$. Then for any input w:

(i) $M(w)$ gives the same answer as $D(w)$ (because M and D decide the same language, L).

(ii) Furthermore, $M(w)$ gives this answer in at most $2^{|w|}$ steps.

To get a contradiction, we consider the input $w = \langle M \rangle$:

(a) $M(\langle M \rangle) = D(\langle M \rangle)$, because of (i) above.

(b) $D(\langle M \rangle) \neq B(\langle M, \langle M \rangle \rangle)$, by definition of D.

(c) $B(\langle M, \langle M \rangle \rangle) = M(\langle M \rangle)$ provided $M(\langle M \rangle)$ runs in at most $2^{|\langle M \rangle|}$ steps, by definition of "B decides **BoundedAccepts**$_{2^\bullet}$".

(d) $M(\langle M \rangle)$ *does* run in at most $2^{|\langle M \rangle|}$ steps, by (ii) above.

Putting together (a)–(d), we get $M(\langle M \rangle) \neq M(\langle M \rangle)$, a contradiction. □

5 Beating every O(.) simultaneously

Why do we say Theorem (D-tier quality Time Hierarchy Theorem) is a "D-tier" Time Hierarchy Theorem? For one thing, it's about specific functions 2^n and 3^n; but this is not hard to generalize. The serious reason is that, contrary to first glance, it does not actually prove $L \notin \mathsf{TIME}(2^n)$. In fact, it does not even prove $L \notin \mathsf{TIME}(1.1^n)$, the original goal.

Important. To see why Theorem (D-tier quality Time Hierarchy Theorem) does not prove $L \notin \mathsf{TIME}(1.1^n)$, recall that $\mathsf{TIME}(1.1^n)$ is all the languages decidable in time $\underline{\mathbf{O}}(1.1^n)$. The $O(\cdot)$ is important here. Theorem (D-tier quality Time Hierarchy Theorem) only shows that L cannot be solved with running time literally at most 2^n. The following would be a perfectly consistent state of affairs:

- L cannot be decided by any TM M with running time $T_M(n) \leq 2^n$.
- L *can* be decided by some TM M with $T_M(n) = 1000 \cdot 1.1^n$, and hence $L \in \mathsf{TIME}(1.1^n)$.

The catch here concerns "small input lengths": $1000 \cdot 1.1^n \leq 2^n$ is not true for small n (specifically, it fails for integers $n < 12$).

Exercise. Show that it is even consistent with the statement of Theorem (D-tier quality Time Hierarchy Theorem) that $L \in \mathsf{P}$.

What we really want is an improved version of Theorem (D-tier quality Time Hierarchy Theorem) that rules out, say, $O(1.1^n)$ running time. Achieving this lower bound against *any* $O(1.1^n)$ running time is both important *and* moderately tricky. (It is one of the trickiest proofs we study in 15-455.) Here is an idea that *doesn't* work:

- Imagine we repeated the proof but using a decider B for $\textbf{BoundedAccepts}_{10\cdot 1.1^\bullet}$ (that is, with time bound $10 \cdot 1.1^{|w|}$ instead of $2^{|w|}$). Tracing through the proof, we'd get some language — call it L_{10} — that is still decidable in $O(3^n)$ time but is *not* decidable in $10 \cdot 1.1^n$ time. But $10 \cdot 1.1^n$ is still not as good as "any $O(1.1^n)$".

- Imagine we repeated the proof but using a decider B for $\textbf{BoundedAccepts}_{100\cdot 1.1^\bullet}$. We'd get a language — call it L_{100} — that is decidable in $O(3^n)$ time but not $100 \cdot 1.1^n$ time. But $100 \cdot 1.1^n$ is still not as good as "any $O(1.1^n)$".

- We could similarly produce L_{1000}, L_{10000}, etc.; in general, for any big number C we could get a language L_C that is decidable in $O(3^n)$ time but not $C \cdot 1.1^n$ time. But what we really want is *one* single language L decidable in $O(3^n)$ time that is not decidable in $C \cdot 1.1^n$ time *simultaneously for every* C.

To see why our proof of Theorem (D-tier quality Time Hierarchy Theorem) isn't good enough, imagine that for the language L it constructs, there *is* an M deciding L with running time $T_M(n) = 1000 \cdot 1.1^n$. Why doesn't the proof achieve a contradiction? The problem is that the proof only tries to show a *single* input w where $M(w)$ gives the wrong answer about $w \in L$, namely $w = \langle M \rangle$. The trouble is, for all we know, this w could be very short, like $|w| < 12$. Such a value for $|w|$ is small enough that the running time of $M(w) = M(\langle M \rangle)$, namely $1000 \cdot 1.1^{|w|}$, is *bigger* than $2^{|w|}$ (despite the fact that "intuitively", $1000 \cdot 1.1^n$ feels less than 2^n). Thus point (d) near the end of Theorem (D-tier quality Time Hierarchy Theorem)'s proof doesn't hold, and we don't get any contradiction.

The trick to improving the proof is to create "artificially longer" versions w' of the string $w = \langle M \rangle$ for which $M(w')$ is identical to $M(w)$. No matter what kind of $T_M(n) = O(1.1^n)$ running time M has, once $|w'|$ is large enough the inequality $T_M(|w'|) \le 2^{|w'|}$ will hold, the sticking point (d) of the proof will be okay, and we will get the required contradiction.

6 C-tier quality Time Hierarchy Theorem

Theorem (C-tier quality Time Hierarchy Theorem)**.** *The proof of Theorem (D-tier quality Time Hierarchy Theorem) can be upgraded to show that*

- $L \in \mathsf{TIME}(3^n)$;
- *there is no (standard-alphabet) TM M deciding L with running time $T_M(n) \le O(1.1^n)$.*

Proof. We just need to prove the second bullet point. Suppose for the sake of contradiction that there is some (standard-alphabet) TM M deciding L with running time $T_M(n) \le O(1.1^n)$. The idea is that this will always be smaller than 2^n for "large enough" n. Doing the math, $T_M(n) \le O(1.1^n)$ means that there exist $C, n_0 \ge 1$ such that $T_M(n) \le C \cdot 1.1^n$ for all $n \ge n_0$. Let n_1 be large enough so that $C \le (\frac{2}{1.1})^{n_1}$. Now setting $N = \max(n_0, n_1)$, we conclude

$$n \ge N \quad \implies \quad T_M(n) \le (\tfrac{2}{1.1})^n \cdot 1.1^n = 2^n.$$

Similar to the previous proof, for any input w we conclude:

(i) $M(w)$ gives the same answer as $D(w)$.

(ii') $M(w)$ gives this answer in at most $2^{|w|}$ steps provided $|w| \ge N$.

Let $w_0 = \langle M \rangle$. Since $|w_0|$ might not be at least N, we want to introduce "artificially longer" versions of w_0. For $i = 1, 2, 3, \ldots$, let M_i denote a TM that consists of taking M and adding i extra "useless" states at the end, meaning states that are never transitioned into. (For concreteness, you can assume that such a state q_j writes back the symbol it reads, moves it head right, and transitions back to q_j. It hardly matters, since no state transitions into q_j.) Let $w_i = \langle M_i \rangle$. The key points are:

- M_i behaves identically to M on every input;
- $|w_i| = |\langle M_i \rangle| = |\langle M \rangle| + c \cdot i = |w_0| + c \cdot i$ for some fixed constant c.

(The second point here depends a bit on our precise TM encoding format. For the format used in the Bevan universal TM, each extra useless state takes $c = 12$ extra symbols. Since we're technically encoding each symbol in Bevan's alphabet by 4 bits, c should be 48.)

Thus the strings w_i get longer and longer, but they all encode a TM that is "functionally equivalent" to M. Now there is some large enough value for i, say $i_* = \lceil N/c \rceil$, such that $|w_{i_*}| \ge N$. Thus item (ii') above implies that

$$M(w_{i_*}) \text{ takes at most } 2^{|w_{i_*}|} \text{ steps.} \qquad (\$)$$

Now we can get a contradiction almost identically to that in Theorem (D-tier quality Time Hierarchy Theorem), by considering the input $w_{i_*} = \langle M_{i_*} \rangle$:

(a) $M(\langle M_{i_*} \rangle) = D(\langle M_{i_*} \rangle)$, because of (i) above.

(b) $D(\langle M_{i_*} \rangle) \ne B(\langle M_{i_*}, \langle M_{i_*} \rangle \rangle)$, by definition of D.

(c) $B(\langle M_{i_*}, \langle M_{i_*} \rangle \rangle) = M_{i_*}(\langle M_{i_*} \rangle)$ provided $M_{i_*}(\langle M_{i_*} \rangle)$ runs in at most $2^{|\langle M_{i_*} \rangle|}$ steps, by definition of "B decides **BoundedAccepts$_{2^\bullet}$**".

(c') $M_{i_*}(\langle M_{i_*} \rangle) = M(\langle M_{i_*} \rangle)$, and both computations take the exact same number of steps, because M_{i_*} is functionally identical to M.

(d) $M_{i_*}(\langle M_{i_*} \rangle)$ runs in the same time $M(\langle M_{i_*} \rangle) = M(w_{i_*})$ does, and this is indeed at most $2^{|w_{i_*}|} = 2^{|\langle M_{i_*} \rangle|}$ steps, by (\$).

Putting together (a)–(d), we get $M(\langle M_{i_*} \rangle) \ne M(\langle M_{i_*} \rangle)$, a contradiction. □

7 B-tier quality Time Hierarchy Theorem

With this $O(\cdot)$ issue settled, we can move on to proving our B-tier Time Hierarchy Theorem, Theorem (B-tier quality Time Hierarchy Theorem), which we repeat here for reference.

Theorem (B-tier THT, restated)**.** *Let $f(n)$ be a clockable function with $n^2 \log n \leq O(f(n))$. Then there exists a language L such that*

- $L \in \mathsf{TIME}(n^2 \cdot f(n) \cdot \log f(n))$;
- $L \notin \mathsf{TIME}(\frac{f(n)}{\log f(n)})$.

Proof. We mostly just need to repeat the proof of Theorem (C-tier quality Time Hierarchy Theorem) and Theorem (D-tier quality Time Hierarchy Theorem), but with B being a decider for $\mathbf{BoundedAccepts}_{f(\bullet)}$ (rather than with the specific $f(\bullet) = 2^{\bullet}$ version). Since we are assuming $f(n)$ is clockable, Theorem (**??**) from the previous chapter tells us that B — and hence L — is in $\mathsf{TIME}(O(n^2 + \log f(n)) \cdot f(n))$. For simplicity we can be wasteful and say that $n^2 + \log f(n) \leq n^2 \cdot \log f(n)$, and therefore we have proven the first bullet point in the theorem, $L \in \mathsf{TIME}(n^2 \cdot f(n) \cdot \log f(n))$.

We move on to the second bullet point, $L \notin \mathsf{TIME}(\frac{f(n)}{\log f(n)})$. We can prove this almost exactly as in the proof of Theorem (C-tier quality Time Hierarchy Theorem); there we only used that any $O(1.1^n)$ function is smaller than 2^n for large enough n. In our general case here, we can use that any $O(\frac{f(n)}{\log f(n)})$ function is smaller than $f(n)$ for large enough n.

There is one more technical detail. Looking carefully at Theorem (C-tier quality Time Hierarchy Theorem), we find that we've only ruled that there is a *standard-alphabet* TM M solving L in $O(\frac{f(n)}{\log f(n)})$ time, whereas to truly say that $L \notin \mathsf{TIME}(\frac{f(n)}{\log f(n)})$, we need to rule out an M solving L in $O(\frac{f(n)}{\log f(n)})$ time using *any* (constant-sized) tape alphabet. But this is not hard; we saw in Proposition (**??**) that an arbitrary-tape-alphabet M running in time $O(\frac{f(n)}{\log f(n)})$ can be converted to a standard-alphabet M' deciding the same language and running in time $O(n^2 + \frac{f(n)}{\log f(n)})$. But using our assumption $n^2 \log n \leq O(f(n))$, we get that $O(n^2) \leq O(\frac{f(n)}{\log f(n)})$, so M' just has running time $O(\frac{f(n)}{\log f(n)})$. But we ruled out standard-alphabet TMs with this running time. □

8 A specific language satisfying the THT

As mentioned, all the Time Hierarchy Theorems we have proven only show that a language L satisfying the conditions of the theorem *exists*. It would be nicer to know a specific language L doing the job. For example, our C-tier quality Theorem (C-tier quality Time Hierarchy Theorem) was already enough to show

$$\exists L \in \mathsf{TIME}(3^n) \setminus \mathsf{TIME}(1.1^n).$$

As promised, we will show that the specific language $L = \mathbf{BoundedAccepts}_{2^\bullet}$ works. In general, the language $L = \mathbf{BoundedAccepts}_{f(\bullet)}$ will work for B-tier-type Time Hierarchy Theorems (though there are some parameter details that are not worth getting into here).

Since we already showed last chapter (Theorem (**??**)) that $\mathbf{BoundedAccepts}_{2^\bullet} \in \mathsf{TIME}(3^n)$, it remains to show:

Theorem. ***BoundedAccepts***$_{2^\bullet} \notin \mathsf{TIME}(1.1^n)$.

Proof. Consider the language L constructed in the proof of Theorem (D-tier quality Time Hierarchy Theorem). We showed that it satisfies $L \notin \mathsf{TIME}(1.1^n)$. But this was actually somewhat sloppy; we could have actually showed it is not in $\mathsf{TIME}(1.9^n)$, or even $\mathsf{TIME}(\frac{2^n}{n})$.

Recall also from the proof that L could be named something like

$$\mathbf{Doesn'tBoundedAcceptSelf}_{2^\bullet};$$

more carefully, we can see that

$$L = \{\langle M \rangle : M \text{ is a (standard-alphabet) TM such that}$$
$$M(\langle M \rangle) \text{ does } \textit{not} \text{ accept within } 2^{|\langle M \rangle|} \text{ steps}\}.$$

Suppose now by way of contradiction that there is a TM B deciding $\mathbf{BoundedAccepts}_{2^\bullet}$ with running time $O(1.1^n)$. We will use it to construct a TM R that solves L in time $O(1.21^n)$. This is indeed a contradiction, because as we said at the beginning of this proof, L is not even in $\mathsf{TIME}(1.9^n)$.

The algorithm R is straightforward: On input w of length n, algorithm R first interprets $w = \langle M \rangle$ for some (standard-alphabet) TM M. (Validating the input here surely takes at most $\mathrm{poly}(n)$ time, much less than $O(1.21^n)$ time.) Next, R copies the string $\langle M \rangle$, forming the input $x = \langle M, \langle M \rangle \rangle$ for B; again, this takes minimal time, say $O(n^2)$. More importantly, note that $|x| \approx 2n$. (Perhaps $2n + O(1)$ due to punctuation, but probably literally $2n$ if we dig into our encoding conventions.) Finally, R performs the code of B on x, and gives the opposite answer.

Overall, it is clear that R correctly decides L. Further, its running time is $\mathrm{poly}(n) + O(1.1^{2n})$. Here $O(1.1^{2n})$ is the running time of B on an input of length $2n$. Finally, $\mathrm{poly}(n) + O(1.1^{2n}) \leq O(1.21^n)$, as claimed, giving the desired contradiction. □