# 7 A Recurring Theme

## 7.1 Recurrences

***Example 7.1: Binary Search.***
Consider the following recurrence relation $T(\circlearrowleft)$ defined below for all $n > 0$.[1]

$$T(1) := 1$$
$$T(2 \cdot n) := T(n) + 1$$

[1] This is one way of expressing the cost, in terms of *"significant operations,"* of running the *Binary Search* algorithm on a sorted list of length $n > 0$.

Suppose we are interested in finding a closed form[2] for this recurrence. To do so, we might try simply *unravelling* the definition by applying it several times until we notice a pattern. Here, we would be looking for a pattern *in terms of* $k$, a positive natural number which we will use to denote the number of times we have applied the definition during this process.[3]

$$\begin{aligned}
T(n) &= T\left(n/2\right) + 1 && (k = 1)\\
&= T\left(n/2^1\right) + 1\\
&= \left(T\left(n/2^1/2\right) + 1\right) + 1 && (k = 2)\\
&= T\left(n/2^2\right) + 2\\
&= \left(T\left(n/2^2/2\right) + 1\right) + 2 && (k = 3)\\
&= T\left(n/2^3\right) + 3\\
&\vdots\\
&= T\left(n/2^k\right) + k && (k \text{ applications of def.})\\
&\vdots\\
&= T\left(n/2^{\log_2(n)}\right) + \log_2(n) && \left(n/2^k = 1 \Leftrightarrow \log_2(n) = k\right)\\
&= T(1) + \log_2(n)\\
&= 1 + \log_2(n)
\end{aligned}$$

[2] A *"closed form"* for a *"function"* like $T(\circlearrowleft)$ is an expression that specifies $T(n)$ for each valid input $n$ without referring to $T$; in other words, a closed form for $T(\circlearrowleft)$ is a *non-recursive* way of expressing what $T(n)$ is for every $n$.

[3] *N.B.* We are not writing a proof right now. While everything we've studied so far certainly still applies, we are just exploring to see if we can discover a *conjecture*—a statement we *think* might be *true*. During this process, we might try whatever feels reasonable, relying on things we've learned throughout our entire lives. Once we *believe* we have something that might be *true*, we put our glasses back on and *state our conjecture* ***formally*** and attempt to ***prove*** *it*.

We can now *conjecture* that $T(n) = \log_2(n) + 1$… but this clearly can't hold for *every* natural number $n$. In particular, it's easy to see that the argument $n$ of $\log_2(\circlearrowleft)$ must be a power of 2; otherwise, $\log_2(n)$ would be *undefined!* We can therefore phrase our conjecture as follows.

***Conjecture 7.1.***
$\forall n(\exists m(n = 2^m) \Rightarrow T(n) = \log_2(n) + 1)$. ⌟

If we were to try to prove this by induction, our proof might go something like this. The statement is $\varphi(x) :=$ "$\exists m(x = 2^m) \Rightarrow T(x) = \log_2(x) + 1$." In our basis step, we would try to prove $\varphi(0)$, which is $\exists m(0 = 2^m) \Rightarrow T(0) = \log_2(0) + 1$.[4] This is an implication, so we begin our proof by assuming the premise.

[4] Notice that $\log_2(0)$ is undefined because 0 is not a power of 2!

*Basis Step:*
Assume that $\exists m(0 = 2^m)$. Recalling that $2 \neq 0$, we have a theorem that tells us

$\forall m(0 \neq 2^m)$, contradicting our original assumption. By *ex falso quodlibet*,[1] we then conclude $T(0) = \log_2(0) + 1$ as desired.

That finishes our basis step. Notice that, because we were proving an implication with a *false* premise, we never had to do any *real* work to derive our conclusion. All of the work happened in showing that our premise was, in fact, *false*. In particular, the fact that $T(0) = \log_2(0) + 1$ contains the undefined term $\log_2(0)$ doesn't end up mattering, since this formula is well-formed and contains no free variables (making it a proposition).[2]

Having finished the basis step, we move on to the inductive step. Let's write down what we need to prove—$\forall k(\varphi(k) \Rightarrow \varphi(k+1))$—in terms of our conjecture for all $k$.

$$\big(\exists m(k = 2^m) \Rightarrow T(k) = \log_2(k) + 1\big)$$
$$\Downarrow$$
$$\big(\exists m(k+1 = 2^m) \Rightarrow T(k+1) = \log_2(k+1) + 1\big)$$

Let's begin our inductive step then by introducing an arbitrary natural number $k$ and assuming our inductive hypothesis.

*Inductive Step:*
Let $k$ be a natural number. Assume $\exists m(k = 2^m) \Rightarrow T(k) = \log_2(k) + 1$. Now, we need to show that $\exists m(k+1 = 2^m) \Rightarrow T(k+1) = \log_2(k+1) + 1$.

Assume that there exists some natural number $m$ such that $k + 1 = 2^m$. Now, …

…

Well, we'd like to use our *inductive hypothesis*, which says $T(k) = \log_2(k) + 1$… but we can only assert that ***if k is a power of* 2**… What we know is that $k + 1$ is a power of 2. Therefore, …

…

Okay, what if we work backwards from the conclusion we want? We want to end our proof with the observation $T(k+1) = \log_2(k+1) + 1$. What do we know about $T(k+1)$? Well, by definition, we know $T(2 \cdot x) = T(x) + 1$ for every positive natural number $x$, and we also know $T(1) = 0$ by definition. Thankfully, we know $k + 1 = 2^m$, so we can take cases based on whether $m = 0$ or $m > 0$.[3]

Sketching this out:
If $m = 0$, then $k + 1 = 2^0 = 1$, and we can apply our definition for $T(1)$.
If $m > 0$, then $m = \tilde{m} + 1$, and then $k + 1 = 2^m = 2^{\tilde{m}+1} = 2^{\tilde{m}} \cdot 2$ so we can apply our recursive definition for $T$, and hopefully our *inductive hypothesis* helps here. Notice that this essentially corresponds to *an additional* ***basis step at* 1** that we may not have originally considered, but that should make sense based on how $T$ is recursively defined! $T$ doesn't have a base case at zero; its base case is at 1.

*Case 1:*
Suppose $m = 0$, so that $k + 1 = 2^0 = 1$ by definition of exponentiation. Then, recalling $T(1) = 1$ by definition, we can simply compute.

$$\begin{aligned} T(k+1) = T(1) = 1 = 0 + 1 \quad & \text{since } \forall x(0 + x = x) \\ = \log_2(1) + 1 \quad & \text{because } 2^0 = 1 \text{ by definition} \end{aligned}$$

[1] This is also known as the *principle of explosion*, but we better know this as ***False Elimination***, the rule of inference that asserts $\bot \vdash q$ for any proposition $q$. In English, this says that from a *false* premise (or a contradiction of the form $\neg p \wedge p$, from which we can obtain $\bot$ using ***False Introduction***), we can prove any proposition.

[2] If we're being technical, depending on our philosophical orientation, and depending on some behind-the-scenes decisions we would have been forced to make earlier when setting up the *Theory of Arithmetic*, it may actually be a formal problem that $\log_2(0)$ is undefined. But that problem is easily resolvable in several different technical ways, and none of them are at all interesting for the purposes of this course—but all of them are tedious. When we move on to our next *theory*, this will no longer be an issue.

[3] This corresponds to taking cases based on whether $k + 1$ is 1 or even, which is the question that the definition of $T$ asks about its input.

Therefore, $T(k+1) = \log_2(1) + 1 = \log_2(k+1) + 1$ as desired.

*Case 2:*
Suppose $m > 0$, so that $m = \tilde{m} + 1$ for some natural number $\tilde{m}$. Then, we can see that $k + 1 = 2^m = 2^{\tilde{m}+1} = 2^{\tilde{m}} \cdot 2 = 2 \cdot 2^{\tilde{m}}$ using the homomorphic property of exponentials and the commutativity of multiplication. We can now apply the definition of our recurrence relation and see the following.

$$T(k+1) = T\left(2 \cdot 2^{\tilde{m}}\right) = T\left(2^{\tilde{m}}\right) + 1$$

By the *inductive hypothesis*, we know $T\left(2^{\tilde{m}}\right) = \log_2\left(2^{\tilde{m}}\right) + 1$… *uhh… wait…* ***That's not what our inductive hypothesis said!*** The *inductive hypothesis* we assumed specifically stated $T(k) = \log_2(k) + 1$, but we don't know $k = 2^{\tilde{m}}$, so we don't know that we can apply our *inductive hypothesis* here! In fact, it is impossible for us to prove that $k = 2^{\tilde{m}}$ because that's not necessarily *true* in general—there are counterexamples! Now we are stuck…

If we *could* apply our *inductive hypothesis* to $2^{\tilde{m}}$, then that would let the rest of our proof fall out easily. But the *Axiom Schema of Weak Induction* doesn't give us the ability to do that…

Except that, *intuitively*, it definitely does. The whole idea behind induction is that we prove a statement $\varphi(\circlearrowleft)$ is *true* about all of the natural numbers by starting at zero and progressively working our way up the number line one-step-at-a-time. By the time we get to $\varphi(k+1)$, we should definitely have already verified $\varphi(k)$, which is why it should intuitively make sense to *"assume $\varphi(k)$"* at the beginning of your inductive step. However… not only have you already verified $\varphi(k)$… you've verified *everything* ***before k*** *as well!* In particular, you've verified ${}^{k+1}/_{2}$ when $k+1$ is even and positive—the exact situation here!

Instead of making a ***weak*** *assumption* as our *inductive hypothesis*, it would be nice if we could make a ***stronger*** *assumption* instead—assuming $\varphi(j)$ for every $j < k+1$. ⌟

Of course, we need *theorems* to verify our intuition and justify that our approach actually does correctly prove $\forall n(\varphi(n))$. As it turns out, there *are* such theorems[1] because our intuition *is* actually correct.[2]

***Theorem 7.1: Weak-Strong Induction.***
Let $\varphi$ we a *wff* with at most one free variable. The following equivalence then holds.

$$\Big(\varphi(0) \wedge \forall k\Big(\forall j\big(j \leqslant k \Rightarrow \varphi(j)\big) \Rightarrow \varphi\big(\mathfrak{suc}(k)\big)\Big)\Big) \Leftrightarrow \forall n\big(\varphi(n)\big)$$

⌟

***Theorem 7.2: Strict-Strong Induction.***
Let $\varphi$ we a *wff* with at most one free variable. The following equivalence then holds.

$$\Big(\varphi(0) \wedge \forall k\Big(\forall j\big(j < k \Rightarrow \varphi(j)\big) \Rightarrow \varphi(k)\Big)\Big) \Leftrightarrow \forall n\big(\varphi(n)\big)$$

⌟

***Theorem 7.3: Strong Induction.***
Let $\varphi$ we a *wff* with at most one free variable. The following equivalence then holds.

$$\forall k\Big(\forall j\big(j < k \Rightarrow \varphi(j)\big) \Rightarrow \varphi(k)\Big) \Leftrightarrow \forall n\big(\varphi(n)\big)$$

⌟

[1] Please note that the names of the three theorems that follow are ***not*** *standard.* In other places and in other contexts, different authors may use the phrase *"Strong Induction"* to refer to any (or all) of these statements. The names *"Weak-Strong Induction"* and *"Strict-Strong Induction"* are also totally made up and probably don't appear anywhere else.

[2] Don't just take this for granted. One should really try to understand *why* this is more than just intuition by thinking about how these theorems might be proved. These are not esoteric theorems; they are totally within our reach by this point.

It is important to note, for any given *wff* $\varphi$, that *Theorem 7.1*, *Theorem 7.2*, and *Theorem 7.3* are all *equivalent* to $\forall n(\varphi(n))$, and that $\forall n(\varphi(n))$ is *equivalent* to weak induction. As a result, all of these forms of induction are equivalent to each other. Formally speaking, there is no reason to prefer any of these schemata over any of the others… except for the problem we ran into with the previous example. For us, the ***humans*** *writing the* ***proofs***, it is typically much easier to write a proof using some form of strong induction than using the weak induction schema from *Axiom 4* because the *inductive hypothesis* that strong induction[1] allows you to assume is much easier to work with than the restrictive one from *Axiom 4*.

But… now we have three versions of *"strong induction"* to choose from. If we think back to the example we were working out about the Binary Search recurrence, the *"basis step"* in our induction required us to prove that our closed form worked at 0, despite the fact that we only asserted our closed form for powers of 2 and that 0 is not a power of 2. As it turns out, if we take a strong inductive approach (regardless which of the three we pick), then the *real "basis step"* shows up during the inductive step because we are forced to consider what happens with the input 1, the first actual power of 2. Proving our closed form works for 0 is a waste of time. In general, 0 may not be the actual basis step you're interested in proving, because your statement may be *"vacuously true"* about 0—but this doesn't mean you don't still have a *nontrivial* basis step hidden elsewhere that you *do* have to account for. How do you find this *"real"* basis step? How many of them do you have (it may be more than just one)? The answers to those questions all depend on the particular problem in front of you.

The most consistent approach—one that is guaranteed to work in any situation when you would like to use induction, but which avoids proving unnecessary bases that are irrelevant to the problem, and which inherently prompts you to think about what bases you actually need—is the from of strong induction given in *Theorem 7.3*, which we will simply call *"strong induction"* without qualification.

Now, equipped with these theorems, we can take another attempt at proving that our closed form for the Binary Search recurrence was actually correct.

***Lemma 7.1: Closed Form for the Binary Search Recurrence.***
Let $T$ be the recurrence relation defined below for any postitive natural number $n$.

$$T(1) \coloneqq 1$$
$$T(2 \cdot n) \coloneqq T(n) + 1$$

We then have $\exists m(n = 2^m) \Rightarrow T(n) = \log_2(n) + 1$ for every natural number $n$. ⌟

***Proof.*** We will prove $\forall n(\exists m(n = 2^m) \Rightarrow T(n) = \log_2(n) + 1)$ by *strong induction.*

Let $k$ be an arbitrary natural number. Assume the statement below.[2]

$$\forall j\Big(j < k \Rightarrow \big(\exists m(j = 2^m) \Rightarrow T(j) = \log_2(j) + 1\big)\Big)$$

Now, we need to show that $\exists m(k = 2^m) \Rightarrow T(k) = \log_2(k) + 1$, which happens to be a conditional statement. To proceed, assume there exists some natural number $x$ such that $k = 2^x$. We now consider two cases.

[1] Let $p_s$ and $p_w$ are two propositions. We typically say *"$p_s$ is* ***stronger*** *than $p_w$" iff* $p_s \Rightarrow p_w$ beacuse that means anything $p_w$ proves can also be proven from $p_s$. We say *"$p_w$ is* ***weaker*** *than $p_s$"* in this situation because there may be things $p_s$ can prove that can't be proven from $p_w$ alone. This is the sense in which we are using the words *"strong"* and *"weak"* when speaking about schemata for induction. Given an arbitrary natural number $k$, the schema of weak induction allows you to assume $\varphi(k)$. However, the schema of weak-strong induction allows you to assume $\forall j(j \leqslant k \Rightarrow \varphi(j))$; in particular, this means you know $\varphi(k)$, but also $\varphi(k-1)$, and $\varphi(k-2)$, *etc.* This inductive hypothesis can *prove* the inductive hypothesis from *Axiom 4*, making this one a *stronger assumption* and that one a *weaker assumption.*

[2] That is our *inductive hypothesis.*

*Case 1:*
Suppose $x = 0$. Then, $k = 2^x = 2^0 = 1$ by definition of exponentiation. We can then observe the following.

$$\begin{aligned} T(k) = T(1) &= 1 && \text{by definition of } T \\ &= 0 + 1 && \text{because } \forall z(0 + z = z) \\ &= \log_2(1) + 1 && \text{by definition of } \log_2(\circlearrowleft) \text{ since } 2^0 = 1 \\ &= \log_2(k) + 1 && \text{because } k = 2^x = 2^0 = 1 \end{aligned}$$

Therefore, we have $T(k) = \log_2(k) + 1$ as desired.

*Case 2:*
Suppose $x \neq 0$; this implies $x > 0$, so that there exists some natural number $y$ such that $x = y + 1$. Notice that, since $y + 1 = x$, we know $y < x$, which implies $2^y < 2^x$ because exponentiation is strictly increasing. This will allow us to apply our *inductive hypothesis* to $T(2^y)$, and we can get $T(2^y)$ by applying the definition of $T$ to $T(2^x)$. This means we are finally ready to observe…

$$\begin{aligned} T(k) &= T(2^x) && \text{because } k = 2^x \\ &= T\left(2^{y+1}\right) && \text{because } x = y + 1 \\ &= T(2^y \cdot 2) && \text{by definition of exponentiation} \\ &= T(2 \cdot 2^y) && \text{by commutativity of multiplication} \\ &= T(2^y) + 1 && \text{by definition of } T \\ &= \left(\log_2(2^y) + 1\right) + 1 && \text{by the } \textit{inductive hypothesis} \text{ because } 2^y < 2^x \\ &= \left(\log_2(2^y) + \log_2\left(2^1\right)\right) + 1 && \text{by definition of logarithms} \\ &= \log_2\left(2^y \cdot 2^1\right) + 1 && \text{because logarithms are homomorphic} \\ &= \log_2\left(2^{y+1}\right) + 1 && \text{because exponentiation is homomorphic} \\ &= \log_2(2^x) + 1 && \text{since } x = y + 1 \\ &= \log_2(k) + 1 && \text{since } k = 2^x \end{aligned}$$

This shows us $T(k) = \log_2(k) + 1$ just like we wanted.

Since these two cases exhaust all of the possibilities (either $x = 0$ or $x \neq 0$), and since we proved $T(k) = \log_2(k) + 1$ in both cases, we can finally conclude with the observation $\forall n(\exists m(n = 2^m) \Rightarrow T(n) = \log_2(n) + 1)$ by the theorem of *strong induction.*

QED

***Lemma 7.2: Closed Form for the Merge Sort Recurrence.***
henlo ⌟