

A Relatively Small Turing Machine Whose Behavior Is Independent of Set Theory

Adam Yedidia

MIT

adamy@mit.edu

Scott Aaronson

MIT

aaronson@csail.mit.edu

March 28, 2016

Abstract

Since the definition of the Busy Beaver function by Radó in 1962, an interesting open question has been what the smallest value of n for which $BB(n)$ is independent of ZFC set theory. Is this n approximately 10, or closer to 1,000,000, or is it even larger? In this paper, we show that it is at most 8,013 by presenting an explicit description of a 8,013-state Turing machine Z with 1 tape and a 2-symbol alphabet that cannot be proved to run forever in ZFC (even though it presumably does), assuming ZFC is consistent. The machine is based on work of Harvey Friedman on independent statements involving order-invariant graphs. In doing so, we give the first known upper bound on the highest provable Busy Beaver number in ZFC. We also present an explicit description of a 4,888-state Turing machine G that halts if and only if there is a counterexample to Goldbach's conjecture, and an explicit description of a 5,372-state Turing machine R that halts if and only if the Riemann hypothesis is false. To create G , R , and Z , we develop and use a higher-level language, Laconic, which is much more convenient than direct state manipulation.

1 Introduction

1.1 Background and Motivation

Zermelo-Fraenkel set theory with the axiom of choice, more commonly known as ZFC, is an axiomatic system invented in the twentieth which has since been used as the foundation of most of modern mathematics. It encodes arithmetic by describing natural numbers as increasing sets of sets.

Like any axiomatic system capable of encoding arithmetic, ZFC is constrained by Gödel's two incompleteness theorems. The first incompleteness theorem states that if ZFC is *consistent* (it never proves both a statement and its opposite), then ZFC cannot also be *complete* (able to prove every true statement). The second incompleteness theorem states that if ZFC is consistent, then ZFC cannot prove its own consistency. Because we have built modern mathematics on top of ZFC, we can reasonably be said to have assumed ZFC's consistency. This means that we must also believe that ZFC cannot prove its own consistency. This fact carries with it certain surprising conclusions.

In particular, consider a Turing machine Z that enumerates, one after the other, each of the provable statements in ZFC. To describe how such a machine might be constructed, Z could iterate over the axioms and inference rules of ZFC, applying each in every possible way to each conclusion

or pair of conclusions that had been reached so far. We might ask Z to halt if it ever reaches a contradiction; in other words, Z will halt if and only if it ever finds a proof of $0 = 1$. Because we know that this machine will enumerate *every* provable statement in ZFC, we know that it will run forever if and only if ZFC is consistent.

It follows that Z is a Turing machine for which the question of its behavior (whether or not it halts when run indefinitely) is equivalent to the consistency of ZFC. While we will talk about ZFC throughout this paper, rather than simple Zermelo-Fraenkel set theory, this is simply convention brought about by the fact that ZFC is a more powerful and more commonly-used set of axioms. In more detail, for the purposes of this paper, the Axiom of Choice is irrelevant: the consistency of ZFC is equivalent to the consistency of simple ZF set theory, [8] and ZFC and ZF prove exactly the same arithmetical statements (which include, among other things, statements about whether Turing machines halt). [16] Therefore, just as ZFC cannot prove its own consistency (assuming ZFC is consistent), ZFC also cannot prove that Z will run forever.

This is interesting because, while the undecidability of the halting problem tells us that there cannot exist an algorithmic method for determining whether an *arbitrary* Turing machine loops or halts, Z is an example of a *specific* Turing machine whose behavior cannot be proven one way or the other using the foundation of modern mathematics. Mathematicians and computer scientists think of themselves as being able to determine how a given algorithm will behave if we are given enough time to stare at it; despite this intuition, Z is a machine whose behavior we can never prove without assuming axioms more powerful than those generally assumed in most of modern mathematics.

1.2 Turing Machines

There are many definitions for Turing machines, each differing slightly from the other. For example, some definitions allow the machine to have multiple tapes; others only allow it to have one. Some definitions allow an arbitrarily large alphabet, while others allow only two symbols. Some definitions allow the tape head to remain in place, while others require it to move at every time-step. In most research regarding Turing machines, mathematicians don't concern themselves with which of these models to use, because any one of them can simulate the others. However, because this work is concerned with upper-bounding the exact number of states required to perform certain tasks, it is important to define precisely what model of Turing machine is being used.

Formally, a k -state Turing machine is a 7-tuple $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$, where:

Q is the set of k states $\{q_0, q_1, \dots, q_{k-2}, q_{k-1}\}$

$\Gamma = \{a, b\}$ is the set of *tape alphabet symbols*

a is the *blank symbol*

Σ is the set of *input symbols*

$\delta = Q \times \Gamma \rightarrow (Q \cup F) \times \Gamma \times \{L, R\}$ is the *transition function*

q_0 is the *start state*

$F = \{\text{ACCEPT}, \text{REJECT}, \text{ERROR}\}$ is the set of *halting transitions*.

A Turing machine's *states* make up the Turing machine's easily-accessible, finite memory. The Turing machine's state is initialized to q_0 .

The *tape alphabet symbols* correspond to the symbols that can be written on the Turing machine's infinite tape.

In this work, all Turing machines discussed are run on the all-**a** input.

The *transition function* encodes the Turing machine's behavior. It takes two inputs: the current state of the Turing machine (an element of Q) and the symbol read off the tape (an element of Γ). It outputs three separate instructions: what state to enter (an element of Q), what symbol to write onto the tape (an element of Γ) and what direction to move the head in (an element of $\{L, R\}$). A transition function specifies the entire behavior of the Turing machine in all cases.

The *start state* is the state that the Turing machine is in at initialization.

A *halting transition* is a transition that causes the Turing machine to halt. While having three possible halting transitions is not necessary for our purposes, being able to differentiate between three different types of halting (ACCEPT, REJECT, and ERROR) is useful for testing.

1.3 The Busy Beaver Function

Consider the set of all Turing machines with k states, for some positive integer k . We call a Turing machine B a *k-state Busy Beaver* if when run on the empty tape as input, B halts, and also runs for at least as many steps before halting as all other halting k -state Turing machines. [15]

In other words, a Busy Beaver is a Turing machine that runs for at least as long as all other halting Turing machines with as many states as it. Another common definition for a Busy Beaver is a Turing machine that writes as many 1's on the tape as possible; because the number of 1's written is a somewhat arbitrary measure, it is not used in this work.

The *Busy Beaver function*, written $BB(k)$, equals the number of steps it takes for a k -state Busy Beaver to halt. The Busy Beaver function has many striking properties. To begin with, it is not *computable*; in other words, there does not exist an algorithm that takes k as input and returns $BB(k)$, for arbitrary values of k . This follows directly from the undecidability of the halting problem. Suppose an algorithm existed to compute the Busy Beaver function; then given a k -state Turing machine M as input, we could compute $BB(k)$ and run M for $BB(k)$ steps. If, after $BB(k)$ steps, M had not yet halted, we could safely conclude that M would never halt. Thus, we could solve the halting problem, which we know is impossible.

By the same argument, $BB(k)$ must grow faster than any computable function. (To check this, assume that some computable function $f(k)$ grows faster than $BB(k)$, and substitute $f(k)$ for $BB(k)$ in the rest of the proof.) In particular, the Busy Beaver grows even faster than (for instance) the Ackermann function, a well-known fast-growing function.

Because finding the value of $BB(k)$ for a given k requires so much work (one must fully explore the behavior of all k -state Turing machines), few explicit values of the Busy Beaver function are known. The known values are [10] [3]:

$$BB(1) = 1$$

$$BB(2) = 6$$

$$BB(3) = 21$$

$$BB(4) = 107$$

For $BB(5)$ and $BB(6)$, only lower bounds are known: $BB(5) \geq 47,176,870$, and $BB(6) \geq 7.4 \times 10^{36,534}$. Researchers are currently working on pinning down the value of $BB(5)$ exactly, and some consider it to be possibly within reach. A summary of the current state of human knowledge about Busy Beaver values can be found at [17].

Another way to discuss the Busy Beaver sequence is to say that modern mathematics has established a *lower bound* of 4 on the highest provable Busy Beaver value. In this paper, we prove the first known *upper bound* on the highest provable Busy Beaver value in ZFC; that is, we give a value of k , namely 8,013, such that the value of $BB(k)$ cannot be proven in ZFC.

Intuitively, one might expect that while no algorithm may exist to compute $BB(k)$ for *all* values of k , we could find the value of $BB(k)$ for any *specific* k using a procedure similar to the one we used to find the value of $BB(k)$ for $k \leq 4$. The reason this is not so is closely tied to the existence of a machine like the Gödelian machine Z , as described in Section 1.1. Suppose that Z has k states. Because Z 's behavior (whether it halts or loops) cannot be proven in ZFC, it follows that the value of $BB(k)$ also cannot be proven in ZFC; if it could, then a proof would exist of Z 's behavior in ZFC. Such a proof would consist of a *computation history* for Z , which is an explicit step-by-step description of Z 's behavior for a certain number of steps. If Z halts, a computation history leading up to Z 's halting would be the entire proof; if Z loops, then a computation history that takes $BB(k)$ steps, combined with a proof of the value of $BB(k)$, would constitute a proof that Z will run forever.

In this paper we construct a machine like Z , for which a proof that Z runs forever would imply that ZFC was consistent. In doing so, we give an explicit upper bound on the highest Busy Beaver value provable in ZFC assuming the consistency of a slightly stronger set theory. Our machine, which we shall refer to as Z hereafter, contains 8,013 states. Therefore, we will never be able to prove the value of $BB(8,013)$ without assuming more powerful axioms than those of ZFC. This upper bound is presumably very far from tight, but it is a first step.

Nontrivial ideas are needed to achieve such a small state count, without which the size of Z might range in the hundreds of thousands or even millions of states. We briefly introduce these ideas in the subsection following this one, and explore them in much greater detail later in the paper. These ideas constitute this paper's main technical contribution.

1.4 Parsimony

In most algorithmic study, efficiency is the primary concern. In designing Z , however, parsimony is the only thing that matters. One historical analogue is the practice of “code-golfing”: a recreational pursuit adopted by some programmers in which the goal is to produce a piece of code in a given programming language, using as few characters as possible. Many examples of code-golfing can be found at [18]. The goal of designing a Turing machine with as few states as possible to accomplish a certain task, without concern for the machine's efficiency or space usage, can be thought of as code-golfing with a particularly low-level programming language.

Part of the charm of Turing machines is that they give us a “standard reference point” for measuring complexity, unencumbered by the details of more sophisticated programming languages. Also, with Turing machines, there can be no suspicion that we engineered a programming formalism just for the purpose of code-golfing, or for making the concepts we want expressed artificially simple to describe. This is why we prefer Turing machines as a tool for measuring complexity; not because they are particularly special, but simply because they are so primitive and so minimal that their specifics will interfere minimally with what we mean by an algorithm being “complicated.”

In this paper, we use three ideas for generating parsimonious Turing machines: Harvey Friedman's mathematical statements, *on-tape processing*, and *introspective* Turing machines. The last of these ideas was proposed, under a different name and with some variations, by Ben-Amram and Petersen in 2002. [2] These three ideas are explained in more detail in Subsections 3.1, 8.1, and 8.3,

respectively, but we will summarize them very briefly here.

The first idea, that of using Friedman’s mathematical statements, is simply to use the research done by Friedman into finding simple-to-express statements that are equivalent to the consistency of various axiomatic systems. In particular, we make use of a statement discovered by Friedman to be equivalent to the consistency of a set theory known to be stronger than ZFC (and whose consistency, therefore, would imply the consistency of ZFC). [6]

The second idea, on-tape processing, is a way to encode high-level commands into a Turing machine parsimoniously. Instead of converting commands to groups of states directly, which incurs a multiplicative overhead based on how large these groups need to be, on-tape processing begins by writing the commands onto the tape, using as efficient an encoding as possible. Then, once the commands are on the tape, the commands are processed by a single group of states that understands how to interpret them.

The third idea, introspective Turing machines, is a way to write long strings onto the tape using as few states as possible. The idea is to encode information one of each state’s transitions, instead of encoding information in each state’s write field. This is advantageous because there are many choices for which state to point a transition to, but only two choices for what bit to write. Therefore, more information can be encoded in each state using this method.

1.5 Implementation Overview

To generate descriptions of Turing machines with nice mathematical properties entirely by hand is a daunting task. Rather than approach the problem directly, we created tools for generating parsimonious Turing machines while presenting an interface that is comfortably familiar to most programmers (and to us!)

We created two tools. At the top level is the Laconic programming language, whose syntax and capabilities are similar to those of most programming languages, such as Java or Python. Beneath it we created a lower-level language called Turing Machine Descriptor (TMD). TMD is quite unlike most programming languages, and is better thought of as a convenient way to describe a multi-tape, 3-symbol Turing machine plus a function stack. The style of multi-tape Turing machine used in TMD is the commonly used “one-tape-at-a-time” abstraction: only one tape at a time can be interacted with, for reading, writing, and moving the head. Laconic compiles down to a TMD program, and TMD compiles down to a description of a single-tape, 2-symbol Turing machine. This process is illustrated in Figure 1.

We recommend that programmers hoping to use our tools to generate their own encodings of mathematical statements or algorithms as Turing machines use Laconic. Laconic’s interface is perfect for somebody hoping to write in a “traditional” language. On the other hand, if the programmer wishes to improve upon Laconic’s compilation process, writing code directly in TMD is likely to be the better option.

2 Related Work

This paper is not the first to attempt to quantify the complexity of arithmetical statements. Calude and Calude [5] define a register machine of their own design, and provide quantifications of the complexity of Legendre’s conjecture, Fermat’s last theorem, Goldbach’s conjecture, Dyson’s conjecture,

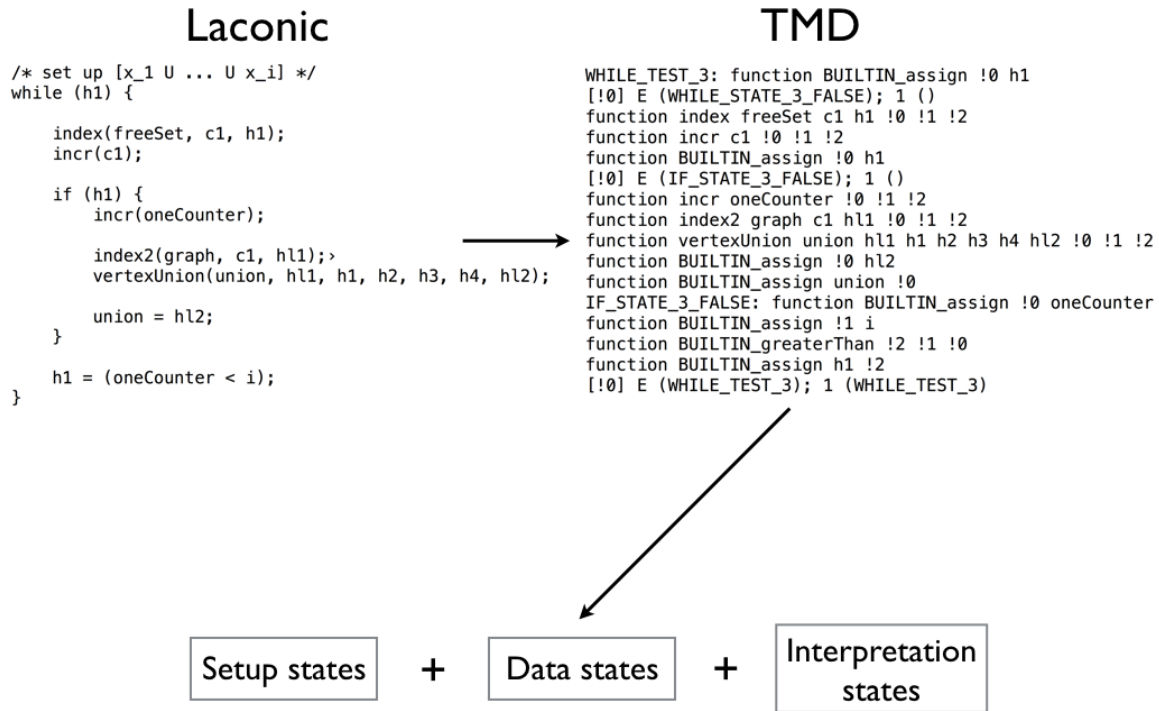


Figure 1: A visual overview of the compilation process.

the Riemann hypothesis, and the four color theorem.¹ In addition, Koza [9] and Pargellis [14] each invent instruction sets that are particularly well-suited to representing self-reproducing programs simply, and show that starting from a “primordial soup” of such instructions distributed about a large memory, along with an increasing number of program threads, a rich ecosystem of increasingly efficient self-reproducing programs start to dominate the “landscape.”

This paper differs from the previous work in two ways: firstly, it is the first to give explicit, relatively small machines whose behavior is provably independent of the standard axioms of modern mathematics. Secondly, to our knowledge, this paper is the first concrete study of parsimony to use Turing machines as the model of computation—rather than (for example) a new programming language proposed by the authors! We consider it important to use the weakest and most common model of computation for complexity comparisons across different mathematical statements. This is because the more powerful and complex the model of computation used, the more of the complexity of the algorithm can be “shunted” onto the model of computation, and the greater the potential distortion created by the choice of model. As a *reductio ad absurdum*, we could imagine a programming language that included “test the Riemann hypothesis” and “test the consistency of ZFC” as primitive operations. By using the “weakest” model of computation that is commonly known, and one which is generally accepted as the mathematical basis of algorithms, we hope to avoid this pitfall and make it easier to interpret our results in a model-independent way.

3 A Turing Machine that Cannot Be Shown to Run Forever Using ZFC

We present a 8,013-state Turing machine whose behavior is *independent of ZFC*; it would not be possible to prove that this machine would halt or wouldn’t halt using the axioms of ZFC, assuming a slightly stronger set theory is consistent. It is therefore impossible to prove the value of $BB(8,013)$ to be any given value without assuming axioms more powerful than ZFC, assuming that ZFC is consistent.

For an explicit listing of this machine, see Appendix C.

We call this machine Z . One way to build this machine would be to start with the axioms of ZFC and apply the inference rules of first-order logic repeatedly in each possible way so as to enumerate every statement ZFC could prove, and to halt if ever a contradiction was found. While this method is conceptually simple, to actually construct such a machine would lead to a huge number of states, because it would require writing a program to manipulate the axioms of ZFC and the inference rules of first-order logic, and then compiling that program all the way down to Turing machine states.

3.1 Friedman’s Mathematical Statement

Thankfully, a simpler method exists for creating Z . Friedman [6] was able to derive a graph-theoretic statement whose truth implies the consistency of ZFC, and which will be false if ZFC is inconsistent.² Here is Friedman’s statement (the notation will be explained in the rest of this

¹Because Fermat’s last theorem and the four color theorem have been proved, their complexity is now known to be 0.

²In fact, Friedman’s statement is equivalent to the consistency of SRP (“stationary Ramsey property”), which is a system of axioms more powerful than ZFC. Because SRP is strictly more powerful than ZFC (it in fact consists of

section):

Statement 1. *For all $k, n, r > 0$, every order invariant graph on $[\mathbb{Q}]^{\leq k}$ has a free $\{x_1, \dots, x_r, \text{ush}(x_1), \dots, \text{ush}(x_r)\}$ of complexity $\leq (8knr)!$, each $\{x_1, \dots, x_{(8kni)!}\}$ reducing $[x_1 \cup \dots \cup x_i \cup \{0, \dots, n\}]^{\leq k}$. [6]*

A number of *complexity* at most c refers to a number that can be written as a fraction a/b , where a and b are both integers less than or equal to c . A set has complexity at most c if all the numbers it contains have complexity at most c .

An *order invariant graph* is a graph containing a countably infinite number of nodes. In particular, it has one node for each finite set of rational numbers. The only numbers relevant to the statement are numbers of complexity $(8knr)!$ or smaller. In every description of nodes that follows, the term *node* refers both to the object in the order invariant graph and to the set of numbers that it represents.

In an order invariant graph, two nodes (a, b) have an edge between them if and only if each other pair of nodes (c, d) that is *order equivalent* with (a, b) has an edge between them. Two pairs of nodes (a, b) and (c, d) are *order equivalent* if a and c are the same size and b and d are the same size and if for all $1 \leq i \leq |a|$ and $1 \leq j \leq |b|$, the i -th element of a is less than the j -th element of b if and only if the i -th element of c is less than the j -th element of d .

To give some trivial examples of order invariant graphs: the graph with no edges is order invariant, as is the complete graph. A less trivial example is a graph on $[\mathbb{Q}]^2$, in which each node corresponds to a set of two rational numbers of a given complexity, and there is an edge between two nodes if and only if their corresponding sets a and b satisfy $a_1 < b_1 < a_2 < b_2$. (Because edges are undirected in order invariant graphs, such an edge will exist if *either* assignment of the vertices to a and b satisfies the inequality above).

The $\text{ush}()$ function takes as input a set and returns a copy of that set with all non-negative numbers in that set incremented by 1.

For vertices x and y , $x \leq_{\text{rlex}} y$ if and only if $x = y$ or $x_i < y_i$ where i is least such that $x_i \neq y_i$.

Finally, a set of vertices X *reduces* a set of vertices Y if and only if for all $y \in Y$, there exists $x \in X$ such that $x \leq_{\text{rlex}} y$ and an edge exists between x and y .

3.2 Implementation Methods

To create Z , we needed to design a Turing machine that halts if Statement 1 is false, and loops if Statement 1 is true. Such a Turing Machine's behavior would necessarily be independent of ZFC, because the truth or falsehood of Statement 1 is independent of ZFC, assuming the consistency of SRP, a slightly more powerful set theory. [6]

To design such a Turing machine, we wrote a Laconic program which encoded Friedman's statement, then compiled the program down to a description of a single-tape, 2-symbol Turing machine. What follows is an extremely brief description of the design of the Laconic program; for the documented Laconic code itself, along with a detailed explanation of the full compilation process, please see [19].

Our Laconic program begins by looping over all non-negative values for k , n , and r . For each trio (k, n, r) , our program generates a list N of all numbers of complexity at most $(8knr)!$. These

ZFC plus some additional axioms), the consistency of SRP implies the consistency of ZFC, and the inconsistency of ZFC implies the inconsistency of SRP.

numbers represent the vertices in our putative order invariant graph. Because Laconic does not support floating-point numbers, the list is entirely composed of integers; it is a list of all numbers that can be written in the form $((8knr)!)((8kni)!)/((8knj)!)$, where i and j are integers satisfying $-(8knr)! \leq i \leq (8knr)!$ and $1 \leq j \leq (8knr)!$. (Note that any number that can be expressed in this form is necessarily an integer, because of the large scaling factor in front.)

After we generate N , we generate the nodes in a potential order invariant graph by adding to N all possible lists of k or fewer numbers from N . We call this list of lists V .

We iterate over all binary lists of length $|V|^2$. Any such list E represents a possible set of edges in the graph. To be more precise, we say that an edge exists between node i and node j (represented by V_i and V_j respectively) if and only if $E_{i|V|+j}$ is 1.

For any graph (V, E) , we say that it is “valid” if the following three conditions hold:

1. No node has an edge to itself.
2. If an edge exists between node i and node j , an edge also exists between node j and node i .
3. The graph has a free $\{x_1, \dots, x_r, \text{ush}(x_1), \dots, \text{ush}(x_r)\}$, each $\{x_1, \dots, x_{(8kni)!}\}$ reducing $[x_1 \cup \dots \cup x_i \cup \{0, \dots, n\}]^{\leq k}$.

For each list of nodes V , we loop over every possible binary list E , and if no pair (V, E) yields a valid graph, we halt.

When verifying the validity of a graph, checking the first two conditions is trivial, but the third merits further explanation. In order to verify that a given graph (V, E) has a free $\{x_1, \dots, x_r, \text{ush}(x_1), \dots, \text{ush}(x_r)\}$, each $\{x_1, \dots, x_{(8kni)!}\}$ reducing $[x_1 \cup \dots \cup x_i \cup \{0, \dots, n\}]^{\leq k}$, we look at every possible subset of the nodes in V . For each subset, we verify that it has length r , that $\text{ush}(x_1), \dots, \text{ush}(x_r)$ all exist in V , and for each i such that $(8kni)! \leq r$, that $\{x_1, \dots, x_{(8kni)!}\}$ reduces $[x_1 \cup \dots \cup x_i \cup \{0, \dots, n\}]^{\leq k}$. Once we have found such a subset, we know that the third condition is satisfied.

I want to argue that the home for the paragraph from Appendix C about actually running the machine belongs in Appendix C instead of here. First, I think that the paragraph is not as technically “serious” as the rest of the material in this subsection. Second, I think that we’re likely to have two sorts of readers: “mathematicians” and “engineers.” This subsection definitely belongs to the mathematicians; Appendix C, with its goofy font and giant list of states, is the home of the engineers. Mathematicians won’t care what happens when you run the Turing machine, but engineers will be curious to hear if we tried it. And the paragraph fits nicely in Appendix C, since we’re talking about the nitty-gritty details of Z .

4 A Turing Machine that Encodes Goldbach’s Conjecture

We present a 4,888-state Turing machine that *encodes Goldbach’s conjecture*; in other words, to know whether this machine halts is to know whether Goldbach’s conjecture is true. It is therefore impossible to prove the value of $BB(4,888)$ without simultaneously proving or disproving Goldbach’s conjecture.

Recall that Goldbach’s conjecture is as follows:

Statement 2. Every even integer greater than 2 can be expressed as the sum of two primes.

Because Goldbach's conjecture is so simple to state, the Laconic program encoding the statement is also quite simple. It can be found in Appendix A. A detailed explanation of the compilation process, documentation for the Laconic language, and an explicit description of this Turing machine are available at [19].

5 A Turing Machine that Encodes Riemann's Hypothesis

We present a 5,372-state Turing machine that *encodes Riemann's hypothesis*; in other words, to know whether this machine halts is to know whether Riemann's hypothesis is true. An explicit description of this machine can be found at [19]

Riemann's hypothesis is traditionally stated as follows:

Statement 3. The Riemann zeta function has its zeros only at the negative even integers and the complex numbers with real part $1/2$.

5.1 Equivalent Statement

Instead of encoding the Riemann zeta function into a Laconic program, it is simpler to use the following statement, which was shown by Lagarias to be equivalent to the Riemann hypothesis [4]:

Statement 4. For all integers $n \geq 1$,

$$\left(\left(\sum_{k \leq \delta(n)} \frac{1}{k} \right) - \frac{n^2}{2} \right)^2 < 36n^3$$

The function $\delta(n)$ used in Statement 4 is defined as follows:

$$\begin{aligned} \eta(j) &= p \text{ if } j = p^k, p \text{ is prime, } k \text{ is a positive integer} \\ \eta(j) &= 1 \text{ otherwise} \\ \delta(x) &= \prod_{n < x} \prod_{j \leq n} \eta(j) \end{aligned}$$

5.2 Implementation Methods

This statement is equivalent to the following statement, which contains only positive integers³:

$$l(n) < r(n)$$

for all positive integers n , where

$$\begin{aligned} l(n) &= (a(n))^2 + (b(n))^2 \\ r(n) &= 36n^3(\delta(n)!)^2 + 2a(n)b(n) \end{aligned}$$

$$a(n) = \sum_{k \leq \delta(n)} \frac{\delta(n)!}{k}$$

³Although it is not immediately obvious at first glance, $\frac{\delta(n)!}{k}$ is necessarily an integer for all $k \leq \delta(n)$, and $\frac{\delta(n)!}{2}$ is an integer for all $n > 1$.

$$b(n) = n^2 \frac{\delta(n)!}{2}$$

To check the Riemann hypothesis, our program computes $a(n)$, $b(n)$, $l(n)$, and $r(n)$, in that order, for each possible value of n . If $l(n) \geq r(n)$, our program halts.

6 Laconic

Laconic is a programming language designed to be both user-friendly and easy to compile down to parsimonious Turing machine descriptions.

Laconic is a strongly-typed language that supports recursive functions. Laconic compiles to an intermediate language called TMD. TMD programs are spread across multiple files and grouped into directories. TMD directories are meant to represent a sequence of commands that could be given to a multi-tape, 3-symbol Turing machine, using the Turing machine abstraction that allows the machine to read and write from one head at a time.

For an example of a Laconic program, see Appendix A. For a visual illustration of the compilation process, see Figure 1.

7 TMD

TMD is a programming language designed to help the user describe the behavior of a multi-tape, 3-symbol Turing machine with a function stack. Each tape is infinite in one direction and supports three symbols: $_$, 1, and E. The blank symbol is $_$: that is, $_$ is the only symbol that can appear on the tape an infinite number of times. The tape must always have the form $_?(1|E)^+_$; in other words, each tape must always contain a string of 1's and E's of size at least 1, possibly preceded by a $_$ symbol, and necessarily followed by an infinite number of copies of the $_$ symbol.

What is the purpose of having a language like TMD as an intermediary between Laconic and a description of a single-tape machine? The concept of tapes in a multi-tape Turing machine and the concept of variables in standard imperative programming languages map to one another very nicely. The idea of the Laconic-to-TMD compiler is to encode the value of each variable on one tape. Then, each Laconic command that manipulates the value of one or more variables compiles down to a TMD function call that manipulates the tapes that correspond to those variables appropriately.

As an example, consider the following Laconic command:

```
a=b*c;
```

This Laconic command assigns the value of **a** to the value of **b*c**. It compiles down to the following TMD function call:

```
function BUILTIN_multiply a b c
```

This function call will result in **BUILTIN_multiply** being run on the three tapes **a**, **b**, and **c**. This will cause the symbols on tape **a** to take on a representation of an integer whose value is equal to bc .

In turn, the TMD code compiles directly to a string of bits that are written onto the tape at the start of the Turing machine's execution.

A TMD directory consists of three types of files:

1. The **functions** file. This file contains a list of the names of all the functions used by the TMD program. The top function in the file is pushed onto the stack at initialization. Moreover, when this top function returns, the Turing machine halts.
2. The **initvar** file. This file contains the non-`_` symbols that start in each register at initialization.
3. Any files used to describe TMD functions. These files all end in a `.tfn` extension and only have any relevance to the compiled program if they show up in the functions file.

8 Compilation and Processing

When discussing the layout of the tape symbols and patterns, there are two ways to think about it: one is with a 4-symbol alphabet (`{_, 1, H, E}`, blank symbol `_`), and one is with a 2-symbol alphabet (`{a, b}`, blank symbol `a`). The 2-symbol alphabet version is the one that's ultimately used for the results in this paper, since we advertised a Turing machine that used only two symbols. However, in nearly all parts of the Turing machine, the 2-symbol version of the machine is a direct translation of the 4-symbol version, according to the following mapping:

- `_` \leftrightarrow `aa`
- `1` \leftrightarrow `ab`
- `H` \leftrightarrow `ba`
- `E` \leftrightarrow `bb`

Additionally, the sections that follow may make reference to the **ERROR** state. Transitions to the **ERROR** state are stand-ins for transitions that should never be taken under any circumstances and are useful for debugging purposes.

8.1 Concept

A directory of TMD functions is converted at compilation time to a string of bits to be written onto the tape, along with other states designed to interpret these bits. The resulting Turing machine has three main components, or *submachines*:

1. The *initializer* sets up the basic structure of the variable registers and the function stack.
2. The *printer* writes down the binary string that corresponds to the compiled TMD code.
3. The *processor* interprets the compiled binary, modifying the variable registers and the function stack as necessary.

The Turing machine’s control flow proceeds from the initializer to the the printer to the interpreter. In other words, initializer states point only to initializer states or to printer states, printer states point only to printer states or to interpreter states, and interpreter states point only to interpreter states or the `HALT` state.

This division of labor, while seemingly straightforward, actually constitutes a very important and non-obvious idea. The problem of the compiler is to convert a higher-level representation—a machine with many tapes, a larger alphabet, and a function stack—to the lower-level representation of a machine with a single tape, a 2-symbol alphabet and no function stack. The immediately obvious solution, and the one taught in every computability theory class as a proof of the equivalence of different kinds of Turing machines, is to have every “state” in the higher-level machine compile down to many states in the lower-level machine.

While simple, this approach is suboptimal in terms of the number of states. As is nearly always true when designing systems to be parsimonious, the clue that improvement is possible lies in the presence of repetition. Each state transition in the higher-level machine is converted to a group of lower-level states with the same basic structure. Why not instead explain how to perform this conversion exactly once, and then apply the conversion many times?

This idea is at the core of the division of labor described previously. We begin by writing a description of the higher-level machine onto the tape, and then “run” the higher-level machine by reading what is on the tape with a set of states that understands how to interpret the encoded higher-level machine. We refer to this idea as *on-tape processing*.

In this paper, we use TMD as the representation of the higher-level machine.⁴ The printer writes the TMD program onto the tape, and the processor executes it. As a result of using this scheme, we incur a constant *additive* overhead—we have to include the processor in our final Turing machine—but we avoid the constant *multiplicative* overhead required for the naïve scheme.

Is this additive overhead small enough to be worth it? We found that it is. Our implementation of the processor requires 3,860 states. (See Section 8.5 for a detailed breakdown of the state cost by submachine.) In contrast to this additive overhead of 3,860, the naïve approach incurs a large multiplicative overhead that depends in part on how many states must be used to represent each higher-level state transition, and in part on how efficient an encoding scheme can be devised for the on-tape approach. The following table compares the performance of on-tape processing to the performance of an implementation that used the naïve approach. The comparison is shown for three kinds of machines: a machine that halts if and only if Goldbach’s conjecture is false, a machine that halts if and only if the Riemann hypothesis is false, and a machine whose behavior is independent of ZFC.

Program	States (Naïve)	States (On-Tape Processing)
Goldbach	7,902	4,888
Riemann	36,146	5,372
ZFC	340,943	8,013

⁴Note that instead of TMD, the on-tape processing scheme could be used for any language, assuming the designer provides both a processor and an encoding for that language. We chose TMD because it made the interpreter easy to write, but other minimalist languages, like Unlambda [11], Brainf*ck [13], or Iota and Jot [1], might be good candidates for parsimonious designs, with the additional advantage of being already known to some programmers! Thanks to Luke Schaeffer for this point. **I don’t think these languages need to be mentioned in the Related Work section—they have little to do with the Busy Beaver sequence or parsimony in general. They’re just languages with very few primitives.**

As can be seen from this table, on-tape interpretation results in huge gains, particularly in large and complex programs.

The subsections that follow describe each of the three submachines—the initializer, the printer, and the processor—in greater detail.

8.2 The Initializer

The initializer starts by writing a counter onto the tape which encodes how many registers there will be in the program. Using the value in that counter, it creates each register, with demarcation patterns between registers, and unique identifiers for each register. Each register’s value begins with the pattern of non- symbols laid out in the `initvar` file. The initializer also creates the program counter, which starts at 0, and the function stack, which starts out with only a single function call to the top function in the `functions` file.

Figure 2 is a detailed diagram describing the tape’s state when the initializer passes control to the printer.

8.3 The Printer

8.3.1 Specification

The printer writes down a long binary string which encodes the entirety of the TMD program onto the tape.

Figure 3 is a detailed diagram describing the tape’s state when the printer passes control to the processor.

8.3.2 Introspection

Writing down a long binary string onto a Turing machine tape in a parsimonious fashion is not as straightforward as it might initially appear. The first idea that comes to mind is simply to use one state per symbol, with each state pointing to the next, as shown in Figure 4.

Upon closer examination, however, it is apparent that this approach is quite wasteful for all but the smallest binary files. Every `a` transition points to the next state in the sequence, and none of the `b` transitions are used at all! Indeed, the only information-bearing part of the state is the single bit contained in the choice of which symbol to write. But in theory, far more information than that could be encoded with each state. In a machine that contains n states, each state could contain $2(\log(n) + 1)$ bits of information, because each of its two transitions could point to any of the n states, and write either an `a` or a `b` onto the tape. Of course, this is only in theory; in practice, to extract the information contained in the Turing machine’s states and translate it into bits on the tape is nontrivial.

What we propose here is a scheme originally conceived by Ben-Amram and Petersen [2] and refined further and suggested to us by Luke Schaeffer. It does not achieve the optimal theoretical encoding described above, but is relatively simple to implement and understand, and is within a factor of 2 of optimal for large binary strings. Schaeffer named Turing machines that use this idea *introspective*.

Introspection works as follows. If the binary string contains k bits, then let w be the *word size*. w takes the largest value it can such that $w2^w \leq k$. We can split the binary string into $n_w = \left\lceil \frac{k}{w} \right\rceil$ different *words* of size w bits each (we can pad the last word with copies of the blank symbol). In

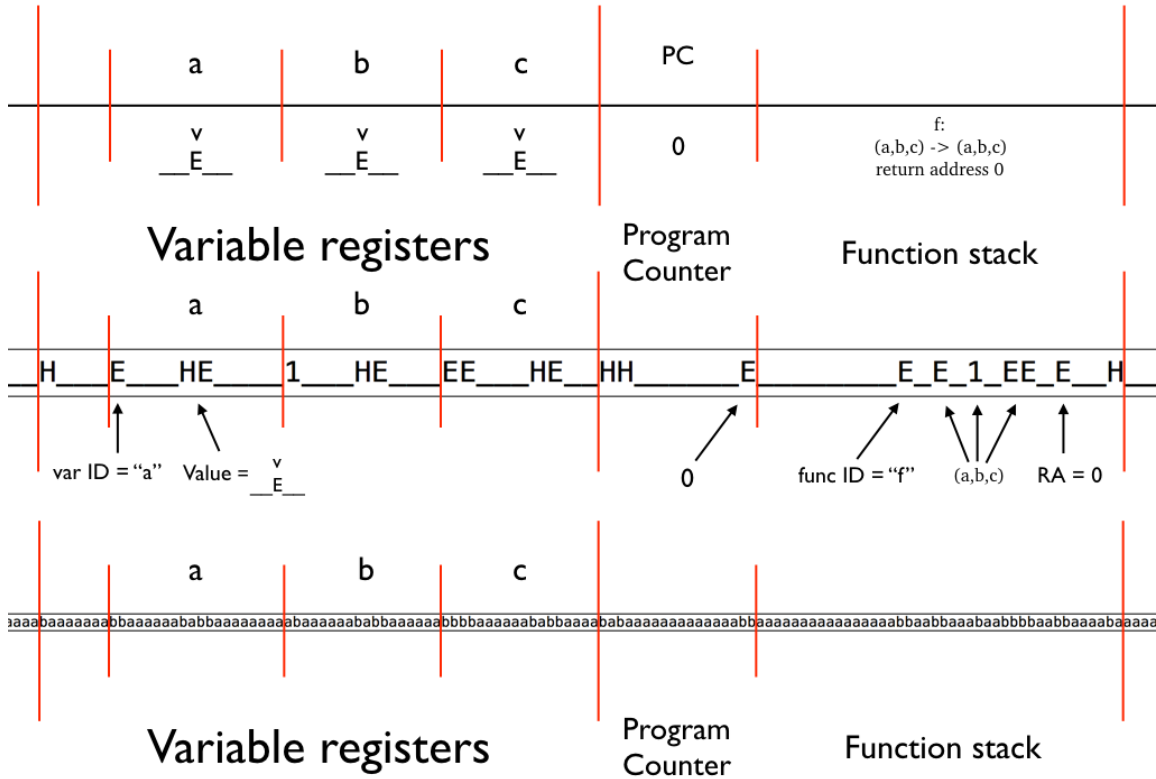


Figure 2: The state of the Turing machine tape after the initializer completes. The TMD program being expressed in Turing machine form is described in full in Appendix B. The top bar is a high-level description of what each part of the Turing machine tape represents. The middle bar is an encoding of the tape in the standard 4-symbol alphabet; the bottom bar is simply the translation of that tape into the 2-symbol alphabet. For a more detailed explanation of how to interpret the tape patterns, see [19].

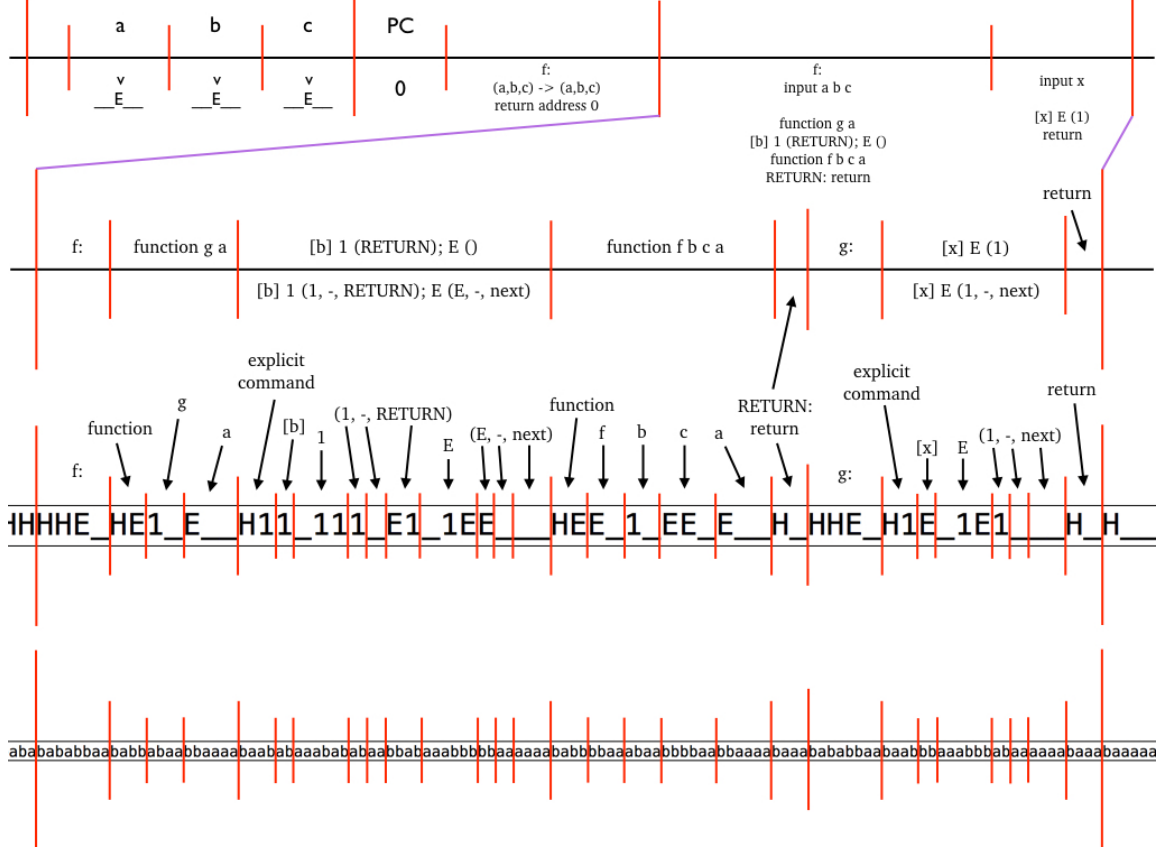


Figure 3: The state of the Turing machine tape after the printer completes. The TMD program being expressed in Turing machine form is described in full in Appendix B. The top bar is a high-level description of the entire tape; unfortunately, at this point there are so many symbols on the tape that it is impossible to see everything at once. For a detailed view of the first two-thirds of the tape (registers, program counter, and stack), see Figure 2. The bottom three bars show a zoomed-in view of the program binary. From the top, the second bar gives a high-level description of what each part of the program binary means; the third bar gives the direct correspondence between 4-symbol alphabet symbols on the tape and their meaning in TMD; the fourth and final bar gives the translation of the third bar into the 2-symbol alphabet. For a more detailed explanation of the encoding of TMD into tape symbols, see [19].

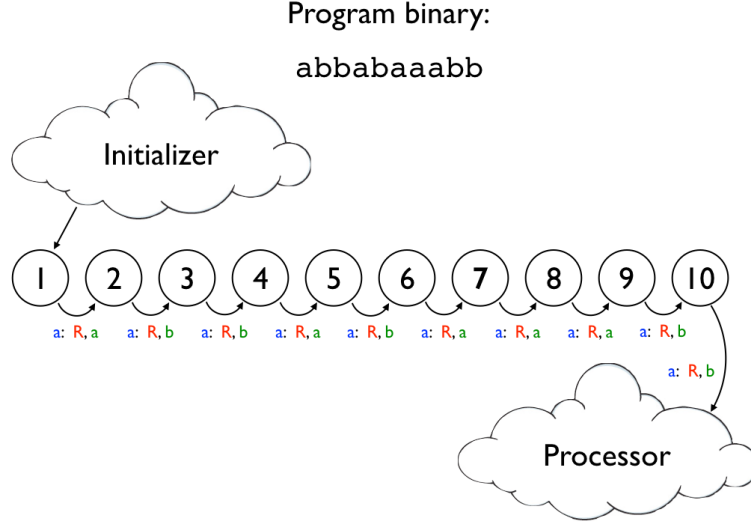


Figure 4: A naïve implementation of the printer. In this example, the hypothetical program is ten bits long, and the printer uses ten states, one for each bit. In the diagram, the blue symbol is the symbol that is read on a transition, the red letter indicates the direction the head moves, and the green symbol indicates the symbol that it written. Note the lack of transitions on reading a **b**; this is because in this implementation, the printer will only ever read the blank symbol, which is **a**, since the head is always proceeding to untouched parts of the tape. It therefore makes no difference what behavior the Turing machine adopts upon reading a **b** in states 1-10 (and therefore **b** transitions are presumed to lead to the **ERROR** state)

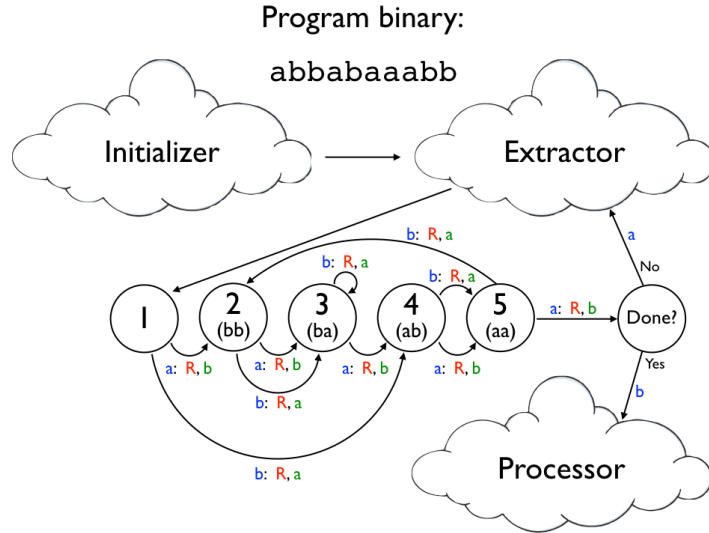


Figure 5: An introspective implementation of the printer. In this example, the hypothetical program is $k = 10$ bits long, and so the word size must be 2 (since $w = 2$ is the largest w such that $w2^w \leq 10$). There are therefore $n_w = \left\lceil \frac{k}{w} \right\rceil = 5$ data states, each encoding two bits. The **b** transitions carry the information about the encoding; note that each one only points to one of the last four data states. The last four data states have in parentheses what word we mean to encode if we point to them.

our scheme, each word in the bit-string will be represented by a *data state*. Each data state points to the state representing the next word in the sequence for its **a** transition, but which state the **b** transition points will encode the next word. Every **b** transition points to one of the last 2^w data states, thereby encoding w bits of information.

Of course, the encoding is useless until we specify how to extract the encoded bit-string from the data states. The extraction scheme works as follows. To query the i^{th} data state for the bits it encodes, we run the data states on the string $\mathbf{a}^{i-1}\mathbf{b}\mathbf{a}^\infty$ (a string of $i - 1$ **a**'s followed by a **b** in the i^{th} position). After running the data states on that string, what remains on the tape is the string $\mathbf{b}^{i-1}\mathbf{a}\mathbf{b}^r\mathbf{a}^\infty$, assuming that the i^{th} data state pointed to the r^{th} -to-last data state. Thus, what we are left with is essentially a unary encoding of the “value” of the word in binary. Thus, the job of the extractor is to set up a binary counter which removes one **b** at a time and increments the counter appropriately. Then, afterward, the extractor reverts the tape back to the form $\mathbf{a}^i\mathbf{b}\mathbf{a}^\infty$, shifts all symbols on the tape over by w bits, and repeats the process. Finally, when the state beyond the last data state sees a **b** on the tape, we know that the process has completed, and we can pass control to the processor. Figure 5 depicts the introspection algorithm.

How much have we gained by using an introspective technique for encoding the program binary, instead of the naïve approach? It depends on how large the program binary is. By using introspection, we incur an $O(\log k)$ *additive* overhead, because we have to include the extractor in our machine. (Our implementation of the extractor takes $10w + 17$ states.) But in return, we save a *multiplicative* factor of w (which scales with $\log k$) on the number of data states needed.

This is plainly not worth it for the 10-bit example binary shown in Figs. 4 and 5. For that binary, we require 69 additional states for the extractor in order to save 5 states on the data states. For real programs, however, it is worth it, as can be seen from the following table.

Program	Binary Size	w	n_w	Extractor Size	States (Naïve)	States (Introspective)
Example TMD	116	4	29	57	116	86
Goldbach	4,964	9	552	107	4,964	659
Riemann	9,532	10	1,024	117	9,532	1,141
ZFC	35,906	11	3,265	127	35,906	3,392

One minor detail concerns the numbers presented for the Riemann program. Ordinarily, with a binary of size 9,532, we would opt to split the program into 1,060 words of 9 bits each plus a 107-state extractor, since 9 is the greatest w such that $w2^w < 9,532$. But because 9,532 is so close to the “magic number” 10,240, it’s actually more parsimonious to pad the program with copies of the blank symbol until it’s 10,240 bits long, and split it into 1,024 words of 10 bits each plus a 117-state extractor. It’s okay to take the greatest w such that $w2^w < 9,532$, because we use $n_w = \left\lceil \frac{9532}{w} \right\rceil$ data states (and NOT 2^w data states, as I believe you are assuming, and which you correctly note would be insufficient to encode the whole bit-string). The scheme that I originally used was what you are suggesting (take the least w such that $w2^w \geq 9,532$, and use 2^w data states) but after I emailed as much to you and Luke, Luke pointed out that we could do better by a multiplicative factor between 1 and 2 by taking w to be one smaller and just having more some extra data states that can’t be pointed to; that way, you don’t have all that wasted space at the end! (In fact, Luke’s method is *almost* always better than the one I originally suggest here and that you suggested in your comments, so what I actually do is check which of the two methods uses less states and use that one; that’s what this paragraph is all about.)

8.4 The Processor

The processor's job is to interpret the code written onto the tape and modify the variable registers and function stack accordingly. The processor does this by the following sequence of steps:

START:

1. Find the function call at the top of the stack. Mark the function f in the code whose ID matches that of the top function call.
2. Read the current program counter. Mark the line of code l in f whose line number matches the program counter.
3. Read l . Depending on what type of command l is, carry out one of the following three lists of tasks.

IF l IS AN EXPLICIT TAPE COMMAND:

1. Read the variable name off l . Index the variable name into the list of variables in the top function on the stack. This list of variables corresponds to the mapping between the function's local variables and the register names.
2. Match the indexed variable to its corresponding register r . Mark r . Read the symbol s_r to the right of the head marker in that register.
3. Travel back to l , remembering the value of s_r using states. Find and mark the reaction x corresponding to the symbol. See what symbol s_w should be written in response to reading s_r .
4. Travel back to r , remembering the value of s_w using states. Replace s_r with s_w .
5. Travel back to x . See which direction d the head should move in response to reading s_r .
6. Travel back to r , remembering the value of d using states. Move the head marker accordingly.
7. Travel back to x . See if a jump is specified. If a jump is specified, copy the jump address onto the program counter. Otherwise, increment the program counter by 1.
8. Go back to START.

IF l IS A FUNCTION CALL:

1. Write the function's name to the top of the stack.
2. For each variable in the function call, index the variable name into the list of variables in the top function on the stack. This list of variables corresponds to the mapping between the function's local variables and the register names. Push the corresponding register names in the order that they correspond to the variables in the function call.
3. Copy the current program counter to the return address of the newborn function call at the top of the stack.

4. Replace the current program counter with 0 (meaning “read the first line of code”).
5. Go back to START.

IF l IS A RETURN STATEMENT:

1. Replace the current program counter with f ’s return address.
2. Increment the program counter by 1.
3. Erase the call to f from the top of the stack.
4. Check if the stack is now empty. If so, halt.
5. Go back to START.

8.5 Cost Analysis

It is worthwhile to analyze the relative contributions of the initializer, the printer, and the processor to the machine’s final state count. The following table lists the number of states in each submachine for each of the four different TMD programs under discussion.

Program	Initializer	Printer	Processor	Total
Example TMD	349	86	3,860	4,295
Goldbach	369	659	3,860	4,888
Riemann	371	1,141	3,860	5,372
ZFC	389	3,392	3,860	8,013

As can be seen from this table, the processor makes the largest contribution to all four programs. Improving the processor, therefore, is probably the best approach for improving upon the bounds we present. Equally clear, however, is that for programs more complicated than the ones presented here, the cost of the printer will grow almost linearly but the cost of the processor will stay the same. The cost of the initializer grows very slightly with the complexity of programs because of the need to initialize additional registers.

Improving the printer, and with it the TMD and Laconic languages, is probably the best approach for reducing state count for very large and complex programs.

9 Future Work

How much further can Z ’s state count be reduced? Without a significant new insight, further order-of-magnitude reductions seem unlikely via tweaks to our existing system: note that such tweaks would need to reduce the sizes of both the printer and the processor by an order of magnitude. However, improvement is possible by (for example) a redesigned processor, combined with a simpler independent mathematical statement than Friedman’s.

Other future work might involve further use of our Laconic language to upper-bound the ‘complexities’ of mathematical statements and algorithms, in as standardized and model-independent a way as possible. Perhaps Laconic could be used to measure the complexity of other well-known conjectures, or even to compare different algorithms for solving the same problem to each other (e.g. to try to quantify the notion that an insertion sort is simpler than a merge sort)!

10 Acknowledgements

We thank Prof. Harvey Friedman for having done the crucial theoretical work that made this project feasible. Prof. Friedman was endlessly available over email, and provided us with immediate and detailed clarifications when we needed them.

We thank Luke Schaeffer for his early help, as well as his help designing introspective Turing machines.

We thank Adam Hesterberg for his help explaining the meaning of various common mathematical terms used in Friedman’s work. *This sentence fits just fine in my thesis, but maybe it should be removed now that you are an author—I imagine that you would have understood the meaning of those mathematical terms just fine without asking Adam! :)*

We thank Alex Arkhipov for introducing us to the term “code golfing.”

Supported by an Alan T. Waterman Award from the National Science Foundation, under grant no. 1249349.

References

- [1] Barker, C. “Iota and Jot: the simplest languages?” <http://semarch.linguistics.fas.nyu.edu/barker/Iota/> [A website describing the Iota and Jot programming languages]
- [2] Ben-Amram, A., Petersen, H. “Improved Bounds for Functions Related to Busy Beavers” *Theory of Computing Systems* 35, 1-11 (2002)
- [3] Brady, A.H. “Solution of the Non-computable ‘Busy Beaver’ game for $k = 4$.” Abstracts for: ACM Computer Science Conference (Washington, DC, February 18-20, 1975), p. 27, ACM, 1975.
- [4] Browder, F. “Mathematical Developments Arising from Hilbert Problems.” American Mathematical Society. Volume 28, Part 1. *I tried to find a page number for this, but among all the papers I found that cited this, nobody else provided a page number for some reason. Perhaps Part 1 of Volume 28 of the AMS only has one page in it?*
- [5] Calude, C., Calude, E. “Evaluating the Complexity of Mathematical Problems: Part 1,” “Evaluating the Complexity of Mathematical Problems: Part 2.” *Complex Systems* 18, pp. 387-401. 2010.
- [6] Friedman, H. “Order Invariant Graphs and Finite Incompleteness.” <https://u.osu.edu/friedman.8/files/2014/01/FIiniteSeqInc062214a-v9w7q4.pdf>
- [7] Friedman, H. “Order Theoretic Equations, Maximality, and Incompleteness.” June 7, 2014. <http://u.osu.edu/friedman.8/foundational-adventures/downloadable-manuscripts/#78>.
- [8] Gödel, K. “The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis with the Axioms of Set Theory.” Published in 1940 by the Princeton University Press. *Annals of Mathematics Studies*.

- [9] Koza, J. “Spontaneous Emergence of Self-Replicating and Evolutionarily Self-Improving Computer Programs.” in *Artificial Life III* (SFI Studies in the Sciences of Complexity, vol. XVII), C. G. Langton, Ed. Reading, MA: Addison-Wesley. pp. 225-262. 1994.
- [10] Lin, S., Rado, T. “Computer Studies of Turing Machine Problems.” Published in *Journal of the ACM*, Volume 12, Issue 2, April 1965. Pages 196-212.
- [11] Madore, D. “The Unlambda Programming Language.” <http://www.madore.org/~david/programs/unlambda/> [A website describing the Unlambda programming language]
- [12] Marxen, H., Buntrock, J. “Attacking the Busy Beaver 5.” *Bull EATCS*, Vol. 40, pp. 247-251. 1990. You wrote: “Ref. [8] should probably have H. Marxen as the ‘author.’” I’m unsure what you mean—haven’t I already listed H. Marxen as the author here?
- [13] Müller, U. “Brainfuck.” <http://www.muppetlabs.com/~breadbox/bf/> [A website describing the Brainf*ck programming language]
- [14] Pargellis, A. “The Spontaneous Generation of Digital ‘Life.’” *Physica D*, 91, 86-96. 1996.
- [15] Rado, T. “On Non-Computable Functions.” *Bell System Technical Journal*, 41: 3. May 1962 pp 877-884.
- [16] Schoenfield, J. “The Problem of Predicativity.” *Essays on the foundations of mathematics*, Y. Bar-Hillel et al., eds., pp. 132-142. 1961.
- [17] <http://www.drb.insel.de/~heiner/BB/> [A list of the known busy beaver values]
- [18] <http://codegolf.stackexchange.com/> [A place where programmers go for recreational code golfing]
- [19] <https://github.com/adamyedidia/parsimony> A link to a GitHub repository containing all programs, Turing machines related to this paper, along with related documentation.

Appendices

A Example Laconic Program: Goldbach’s Conjecture

The following is an example Laconic program, which compiles down to the Turing machine G mentioned in Section 4 (which halts if and only Goldbach’s Conjecture is false).

```
func zero(x) {
    x = 0;
    return;
}

func one(x) {
    x = 1;
    return;
}

func incr(x) {
    x = x + 1;
}
```

```

    return;
}

/* Computes x modulo y */
func modulus(x, y, out) {
    out = x;

    while (out >= y) {
        out = out - y;
    }

    return;
}

func assignXtoYminusX(x, y) {
    x = y - x;
    return;
}

/* Figures out if x is prime, and puts the output in y */
/* Does not modify x, modifies y */
func isPrime(x, h, y) {
    if (x == 1) {
        zero(y);
        return;
    }

    y = 2;

    while (x > y) {
        modulus(x, y, h);

        if (h == 0) {
            zero(y);
            return;
        }
        incr(y);
    }

    return;
}

int evenNumber;
int primeCounter;
int isThisOnePrime;
int foundSum;
int h;

evenNumber = 2;
one(foundSum);

while (foundSum) {
    zero(foundSum);
    evenNumber = evenNumber + 2;
    one(primeCounter);

    while (primeCounter < evenNumber) {
        isPrime(primeCounter, h, isThisOnePrime);

        if (isThisOnePrime) {
            assignXtoYminusX(primeCounter, evenNumber);
            isPrime(primeCounter, h, isThisOnePrime);
            assignXtoYminusX(primeCounter, evenNumber);

            if (isThisOnePrime) {

```

```

        print evenNumber;
        print primeCounter;

        one(foundSum);
    }
}

    incr(primeCounter);
}

halt;

```

For detailed documentation of the Laconic programming language, see [19]. To find this file specifically, navigate to `parsimony/src/laconic/laconic_files/goldbach.lac` at [19].

B Example TMD Program

The following is an example TMD directory, which compiles down to a binary string to be written on a Turing machine tape. It is the example that is used in illustrations throughout this paper, most notably in the example compilation shown in Figs. 2 and 3. The program calls itself recursively three times until the starting symbol on each tape, **E**, is replaced with a 1, at which point the program halts.

This TMD directory is called `example_tmd_dir`, and contains four files: `f.tmd`, `g.tmd`, `initvar`, and `functions`.

```

f.tmd:
input a b c

// Recursively writes a 1 on every tape.

function g a
[b] 1 (RETURN); E ()
function f b c a
RETURN: return

g.tmd:
input x

// Writes a 1 on the input tape.

[x] E (1)
return

functions:
f
g

initvar:
E

```

For detailed documentation of the TMD programming language, see [19]. To find this directory specifically, navigate to `parsimony/src/tmd/tmd_dirs/example_tmd_dir/` at [19].

C Explicit Listing of Z

We present below an explicit listing of Z. For a more easily readable version of Z, complete with descriptive state names, see [19].

We ran this Turing Machine for 10,000,000,000 steps (more than half a day on our simulators) and within that time it did not halt. We note, however, that it was designed for parsimony and not for efficiency, and that this “experiment” is of little consequence! We similarly designed and ran a Turing machine designed to test the conjecture that all perfect squares are less than 5, and it ran for 2,895,083,899 steps (a couple hours on our simulators) before it found the counterexample 9 and halted.

See Section 3.2 for why I kept the paragraph here

Figure 6 presents useful information for how to interpret the description shown below. In addition, note the following:

1. The tape has a 2-symbol alphabet, with tape symbols {a, b} and blank symbol a (in other words, a is the only symbol that can appear an infinite times on the tape).
2. The start state of Z is 0000.
3. Z will never transition to the ERROR state. Any transition to the ERROR state could be replaced by a transition to any other state (including HALT) and the Turing machine’s behavior would remain identical.
4. Z contains only one transition to the HALT state, out of state 7593.

```
0000(0001bR|ERROR-) 0001(0004bR|ERROR-) 0002(0003bR|ERROR-) 0003(0012aR|0012bR) 0004(0005bR|ERROR-) 0005(0006bR|ERROR-) 0006(0007aR|ERROR-) 0007(0008bR|ERROR-) 0008(0009aR|ERROR-) 0009(0010bR|ERROR-)
0010(0011aR|ERROR-) 0011(0022aR|ERROR-) 0012(0013aR|ERROR-) 0013(0014aR|0014bR) 0014(0015aR|ERROR-) 0015(0057aR|0057bR) 0016(0017bR|ERROR-) 0017(0018bR|ERROR-) 0018(0019aR|ERROR-) 0019(0020aR|0020bR)
0020(0021aR|ERROR-) 0021(0022aR|0022bR) 0022(0023aR|0023bR) 0023(0024aR|0024bR) 0024(0025aR|ERROR-) 0025(0067aR|0067bR) 0026(0027aR|0027bR) 0027(0028aR|0028bR) 0028(0029aR|0029bR) 0029(0030aR|0030bR)
0030(0031aR|0031bL) 0031(0026aL|0026bL) 0032(0033aL|0035bL) 0033(0034aL|0034bL) 0034(0037aL|0037bL) 0035(0036aL|0036bL) 0036(0037aL|0037bL) 0037(0038aL|0041bR) 0038(0044aR|0039bL) 0039(0040bR|0040bR)
0040(0041aR|0049bL) 0041(ERROR|0042bL) 0042(0043aL|0043aL) 0043(0037aL|0037bL) 0044(0045aR|0048bR) 0045(ERROR|0046aL) 0046(0047aR|0047aR) 0047(0049aL|0049bL) 0048(0071aR|ERROR-) 0049(0050aR|0051bR)
0050(0049aR|0049aR) 0051(0052aR|0049aR) 0052(0053aR|0053aR) 0053(0052aR|0052bR) 0054(0055aL|0055bR) 0055(0056aR|0056aR) 0056(0012aR|0012bR) 0057(0058aR|ERROR-) 0058(0059aR|0059aR) 0059(0060aR|ERROR-)
0060(0061aR|0061bR) 0061(0062aR|ERROR-) 0062(0063aR|0063bR) 0063(0064aR|0065bR) 0064(0065aR|0065bR) 0065(0066aR|ERROR-) 0066(0016aR|0016bR) 0067(0068bR|ERROR-) 0068(0069bR|ERROR-) 0069(0070bL|ERROR-)
0070(0026aL|0026bL) 0071(0072aR|0075bR) 0072(0071aR|0073bL) 0073(0074aR|0074bR) 0074(0075aR|0078bL) 0075(ERROR|0076bL) 0076(0077aR|0077bR) 0077(0078aL|0078bL) 0078(0079aR|0082bR) 0079(0080aR|0078bR)
0080(0081bR|0081bR) 0081(0083aR|0083bR) 0082(0079aR|0079bR) 0083(0084aR|0085bR) 0084(0085aR|0083bR) 0085(0095aL|0093bR) 0086(0087aL|0087bL) 0087(0088aL|0088bL) 0088(0089aR|0094bR) 0089(0090aL|0092bL)
0090(0091aR|0091bR) 0091(0099aL|0099bL) 0092(0093aL|0093bL) 0093(0088aL|0088bL) 0094(0095aL|0097bL) 0095(0096aL|0096bL) 0096(0098aL|0098bL) 0097(0098aL|0098bL) 0098(0099aR|0099bR) 0099(0100aR|0105bR)
0100(0101aL|0103bL) 0101(0102aL|0102bL) 0102(0099aL|0099bL) 0103(0104aR|0104bR) 0104(0101aL|0110bL) 0105(0106aL|0108bL) 0106(0107aR|0107aR) 0107(0139aL|0139bL) 0108(0109aR|0109bR) 0109(0110aL|0110bL)
0110(0111aR|0113bR) 0111(0112aL|0114bL) 0112(0113aL|0115bL) 0113(0114aR|0115bL) 0114(0115aL|0119bL) 0115(0116aL|0128bL) 0116(0117aR|0127bL) 0117(0118aR|0118aR) 0118(0119aR|0119aR) 0119(0120aR|0126bR)
0120(0121aL|0123bL) 0121(0122aR|0130bL) 0122(0130aL|0130bL) 0123(0124aL|0124bL) 0124(0119aL|0119bL) 0125(0126aL|0128bL) 0126(0127aL|0127bL) 0127(0119aL|0127bL) 0128(0129aL|0129bL) 0129(0119aL|0119bL)
0130(0131aR|0136bR) 0131(0132aL|0134bL) 0132(0133aL|0133bL) 0133(0130aL|0130bL) 0134(0135aR|0135bR) 0135(0088aL|0088bL) 0136(ERROR|0137bL) 0137(0138aR|0138bR) 0138(0088aL|0088bL) 0139(0140aR|0143bR)
0140(0139aR|0141bL) 0141(0142aR|0142bR) 0142(0146aL|0146bL) 0143(ERROR|0144bL) 0144(0145aR|0145bR) 0145(0146aL|0146bL) 0146(0147aR|0150bR) 0147(0148aL|0148aR) 0148(0149aR|0149bR) 0149(0071aL|0151bL)
0150(0151aL|0146bR) 0151(0152aR|0153aR) 0152(0153aR|0153bR) 0153(0154aR|0154bR) 0154(0155aR|0155bR) 0155(0156aR|0156bR) 0156(0157aL|0157bL) 0157(0158aR|0163bR) 0158(0159aL|0161bL) 0159(0160aL|0160bL)
0160(0157aL|0157bL) 0161(0162aR|0162bR) 0162(0166aL|0166bL) 0163(0197aR|0164bL) 0164(0165aR|0165bR) 0165(0166aL|0166bL) 0166(0167aR|0172bR) 0167(0168aL|0170bL) 0168(0169bL|0169bL) 0169(0177aL|0177bL)
0170(0171aL|0171bL) 0171(0166aL|0166bL) 0172(0173aL|0175bL) 0173(0174aL|0174bL) 0174(0165aR|0166bL) 0175(0176aL|0176bL) 0176(0166aL|0166bL) 0177(0178aR|0183bR) 0178(0179aL|0181bL) 0179(0180aL|0180bL)
0180(0177aL|0177bL) 0181(0182aR|0182bR) 0182(0186aL|0186bL) 0183(ERROR|0184bL) 0184(0185aR|0185bR) 0185(0186aL|0186bL) 0186(0187aR|0192bR) 0187(0188aL|0190bL) 0188(0189aR|0189bR) 0189(0157aL|0157bL)
0190(0191aL|0191bL) 0191(0186aL|0186bL) 0192(0193aL|0195bL) 0193(0194aL|0194bL) 0194(0186aL|0186bL) 0195(0186aL|0196bL) 0196(0186aL|0186bL) 0197(0198aR|0199bR) 0198(0197aR|0197bR) 0199(0200aR|0197bR)
0200(0201aR|0201bR) 0201(0202aR|0203bR) 0202(0203aR|0203bR) 0203(0205aR|0205bR) 0204(0206aR|0197bR) 0205(0206aR|0197bR) 0206(0207aR|0207bR) 0207(0208aR|0209bR) 0208(0209aR|0209bR) 0209(0210aR|0210bR)
0210(0211aR|0219bR) 0211(0212bR|ERROR-) 0212(0213bR|ERROR-) 0213(0214aR|0213bR) 0214(0233aR|0233bR) 0215(0216aR|ERROR-) 0216(0217bR|ERROR-) 0217(0218bL|ERROR-) 0218(0219aR|0219bR) 0219(0220aR|0263bL)
0220(0221aR|0221bR) 0221(0222aR|ERROR-) 0222(0223aR|0223bR) 0223(0224aR|ERROR-) 0224(0225aR|0225bR) 0225(0226aR|ERROR-) 0226(0227aR|0227bR) 0227(0228aR|ERROR-) 0228(0229aR|0229bR) 0229(0230aR|ERROR-)
0230(0231aR|0231bR) 0231(0232aR|ERROR-) 0232(0211aR|0211bR) 0233(0234aR|0234bR) 0234(0235aR|ERROR-) 0235(0236aR|ERROR-) 0236(0237aR|ERROR-) 0237(0238aR|ERROR-) 0238(0239aR|ERROR-) 0239(0240aR|ERROR-)
0240(0241aR|ERROR-) 0241(0242bR|ERROR-) 0242(0243bR|ERROR-) 0243(0244aR|ERROR-) 0244(0245aR|ERROR-) 0245(0246aR|ERROR-) 0246(0247aR|0247bR) 0247(0248aR|ERROR-) 0248(0249aR|ERROR-) 0249(0250aR|ERROR-)
0250(0251aR|0251bR) 0251(0252aR|ERROR-) 0252(0253aR|0253bR) 0253(0254aR|0254bR) 0254(0255aR|0255bR) 0255(0256aR|0256bR) 0256(0257aR|0257bR) 0257(0258aR|ERROR-) 0258(0259aR|0259bR) 0259(0260aR|ERROR-)
0260(0261aR|0261bR) 0261(0262aR|ERROR-) 0262(0215aR|0215bR) 0263(0264aR|0269bR) 0264(0265aL|0267bL) 0265(0266aL|0266bL) 0266(0263aL|0263bL) 0267(0268aL|0268bL) 0268(0269aR|0269bR) 0269(0270aL|0272bL)
0270(0271aL|0271bL) 0271(0272aL|0273bL) 0272(0273aL|0273bL) 0273(0263aL|0263bL) 0274(0275aR|0275bR) 0275(0276aR|0276bL) 0276(0277aR|0277bL) 0277(0288aL|0288bL) 0278(0289aR|0290bR) 0279(0290aR|0288bR)
0280(0274aL|0274bL) 0281(0282aR|0285bR) 0282(ERROR|0283aL) 0283(0284aR|0284bR) 0284(0285aR|0288bL) 0285(0286aR|ERROR-) 0286(0287aR|0287aR) 0287(0300aR|0300bR) 0288(0289aR|0301bR) 0289(0288aR|0288bR)
0290(0291aR|0289bR) 0291(0292aR|0293bR) 0292(0291aR|0291bR) 0293(0294aL|0291bR) 0294(0295aR|0295bR) 0295(0296aR|0296bR) 0296(0297aR|ERROR-) 0297(0298aR|ERROR-) 0298(0299bL|ERROR-) 0299(0283aL|0283bL)
0300(0301aR|0302bR) 0301(0300aR|0300bR) 0302(0300aR|0303bR) 0303(0304aR|0305bR) 0304(0308aR|0386bR) 0305(0308aR|0386bR) 0306(0307aR|0386bR) 0307(0308aR|0308bR) 0308(0309aR|0310bR) 0309(0308aR|0308bR)
0310(0311aL|0306bR) 0311(0312aL|0312bL) 0312(0313aL|0313bL) 0313(0314aR|0317bR) 0314(0339aR|0315bL) 0315(0316aL|0316bL) 0316(0313aL|0313bL) 0317(0338aR|0318bL) 0318(0319aR|0319aR) 0319(0360aL|0360bL)
0320(0321aR|0324bR) 0321(0320aR|0322aL) 0322(0323aR|0323bR) 0323(0324aR|0329bR) 0324(0325aL|0327bL) 0325(0326aR|0326aR) 0326(0347aR|0347bR) 0327(0328aR|0328aR) 0328(0338aR|0338bR) 0329(0339aR|0339bR)
0330(0331aL|0329bR) 0331(0332aR|0332bR) 0332(0333aR|0336bL) 0333(0334aR|0336bL) 0334(0335aR|0335aR) 0335(0336aR|0340bR) 0336(0337aR|0337aR) 0337(0338aR|0338bR) 0338(0339aR|0343bR) 0339(0340bL|0342bL)
0340(0341aR|0341bR) 0341(0320aR|0320bR) 0342(0343bR|0343bR) 0343(0329aR|0329bR) 0344(0345bL|0338bR) 0345(0346aR|0346aR) 0346(0347aR|0347aR) 0347(0348bL|0349bL) 0348(0349aR|0349bR) 0349(0350aR|0350bR)
0350(0351aL|0353bL) 0351(0352aL|0352bL) 0352(0349aL|0349bL) 0353(0354aR|0354bR) 0354(0351aL|0351bL) 0355(0356aL|0358bL) 0356(0357aR|0357aR) 0357(0371aR|0371bR) 0358(0369aL|0369bL) 0359(0369aL|0369bL)
0360(0361aR|0366bR) 0361(0362aL|0364bL) 0362(0363aL|0363bL) 0363(0313aL|0313bL) 0364(0365aL|0365bL) 0365(0366aL|0366bL) 0366(0367aL|0369bL) 0367(0368aR|0368aR) 0368(0371aR|0371bR) 0369(0370aL|0355bR)
0370(0360aL|0360bL) 0371(0372aR|0375bR) 0372(0373aL|0371bR) 0373(0374bR|0374bR) 0374(0380aR|0380bR) 0375(0376aL|0371bR) 0376(0377aR|0377aR) 0377(0378aR|0378bR) 0378(0379bR|ERROR-) 0379(0380bR|ERROR-)
0380(0381aR|ERROR-) 0381(0382aR|0382bR) 0382(0383aR|ERROR-) 0383(0384aR|0384bR) 0384(0385bR|ERROR-) 0385(0389aR|0389bR) 0386(0387aR|0389bR) 0387(0386aR|0386bR) 0388(0390aR|0390bR) 0389(0390aR|ERROR-)
0390(0391aR|ERROR-) 0391(0392aR|0392bR) 0392(0393aL|0392aL) 0393(0393aL|0394bR) 0394(0415aR|ERROR-) 0395(0396bR|0262aR) 0396(0397bR|0318aR) 0397(0398bR|0357aR) 0398(0399bR|0296aR) 0399(0400bR|0310aR)
0400(0401bR|0260aR) 0401(0402bR|0395aR) 0402(0403bR|0242aR) 0403(0404bR|0208aR) 0404(0405bR|1998aR) 0405(0406bR|0390aR) 0406(0407bR|0319aR) 0407(0408bR|0237aR) 0408(0409bR|0217aR) 0409(0410bR|0264aR)
0410(0411bR|0212aR) 0411(0412bR|0327aR) 0412(0413bR|0372aR) 0413(0414bR|0340aR) 0414(0415bR|0276aR) 0415(0416bR|0261aR) 0416(0417bR|0305aR) 0417(0418bR|0340aR) 0418(0419bR|0276aR) 0419(0420bR|0350aR)
0420(0421bR|0255aR) 0421(0422bR|0307aR) 0422(0423bR|0263aR) 0423(0424bR|0339aR) 0424(0425bR|0318aR) 0425(0426bR|0207aR) 0426(0427bR|0243aR) 0427(0428bR|0251aR) 0428(0429bR|0251aR) 0429(0430bR|0251aR)
0430(0431bR|0296aR) 0431(0432bR|0310aR) 0432(0433bR|0244aR) 0433(0434bR|0394aR) 0434(0435bR|0217aR) 0435(0436bR|0361aR) 0436(0437bR|0244aR) 0437(0438bR|0310aR) 0438(0439bR|0353aR) 0439(0440bR|0345aR)
0440(0441bR|0360aR) 0441(0442bR|0385aR) 0442(0443bR|0205aR) 0443(0444bR|0205aR) 0444(0445bR|0304aR) 0445(0446bR|0393aR) 0446(0447bR|0217aR) 0447(0448bR|0236aR) 0448(0449bR|0265aR) 0449(0450bR|0204aR)
0450(0451bR|0217aR) 0451(0452bR|0210aR) 0452(0453bR|0258aR) 0453(0454bR|0232aR) 0454(0455bR|0214aR) 0455(0456bR|0203aR) 0456(0457bR|0387aR) 0457(0458bR|0387aR) 0458(0459bR|0312aR) 0459(0460bR|0325aR)
0460(0461bR|0299aR) 0461(0462bR|0191aR) 0462(0463bR|0254aR) 0463(0464bR|0206aR) 0464(0465bR|0207aR) 0465(0466bR|0365aR) 0466(0467bR|0357aR) 0467(0468bR|0239aR) 0468(0469bR|0353aR) 0469(0470bR|0301aR)
0470(0471bR|0267aR) 0471(0472bR|0310aR) 0472(0473bR|0272aR) 0473(0474bR|0324aR) 0474(0475bR|0260aR) 0475(0476bR|0339aR) 0476(0477bR|0217aR) 0477(0478bR|0396aR) 0478(0479bR|0396aR) 0479(0480bR|0243aR)
0480(0481bR|0370aR) 0481(0482bR|0339aR) 0482(0483bR|0269aR) 0483(0484bR|0210aR) 0484(0485bR|0285aR) 0485(0486bR|0388aR) 0486(0487bR|0279aR) 0487(0488bR|0226aR) 0488(0489bR|0226aR) 0489(0490bR|0357aR)
0490(0491bR|0279aR) 0491(0492bR|0257aR) 0492(0493bR|0269aR) 0493(0494bR|0310aR) 0494(0495bR|0217aR) 0495(0496bR|0302aR) 0496(0497bR|0283aR) 0497(0498bR|0326aR) 0498(0499bR|0295aR) 0499(0500bR|0364aR)
0500(0501bR|0242aR) 0501(0502bR|0391aR) 0502(0503bR|0319aR) 0503(0504bR|0358aR) 0504(0505bR|0272aR) 0505(0506bR|0306aR) 0506(0507bR|0236aR) 0507(0508bR|0227aR) 0508(0509bR|0372aR) 0509(0510bR|0355aR)
0510(0511bR|0279aR) 0511(0512bR|0366aR) 0512(0513bR|0216aR) 0513(0514bR|0288aR) 0514(0515bR|0243aR) 0515(0516bR|0275aR) 0516(0517bR|0220aR) 0517(0518bR|0293aR) 0518(0519bR|0292aR) 0519(0520bR|0369aR)
0520(0521bR|0249aR) 0521(0522bR|0200aR) 0522(0523bR|0364aR) 0523(0524bR|0415aR) 0524(0525bR|0202aR) 0525(0526bR|0088aR) 0526(0527bR|0240aR) 0527(0528bR|0210aR) 0528(0529bR|0255aR) 0529(0530bR|0362aR)
0530(0531bR|0276aR) 0531(0532bR|0341aR) 0532(0533bR|0304aR) 0533(0534bR|0293aR) 0534(0535bR|0210aR) 0535(0536bR|0340aR) 0536(0537bR|0278aR) 0537(0538bR|0352aR) 0538(0539bR|0352aR) 0539(0540bR|0352aR)
0540(0541bR|0327aR) 0541(0542bR|0386aR) 0542(0543bR|0267aR) 0543(0544bR|0249aR) 0544(0545bR|0204aR) 0545(0546bR|0081aR) 0546(0547bR|0244aR) 0547(0548bR|0301aR) 0548(0549bR|0216aR) 0549(0550bR|0327aR)
0550(0551bR|0327aR) 0551(0552bR|0353aR) 0552(0553bR|0289aR) 0553(0554bR|0257aR) 0554(0555bR|0269aR) 0555(0556bR|0310aR) 0556(0557bR|0217aR) 0557(0558bR|0302aR) 0558(0559bR|0283aR) 0559(0560bR|0307aR)
0560(0561bR|0360aR) 0561(0562bR|0364aR) 0562(0563bR|0215aR) 0563(0564bR|0201aR) 0564(0565bR|0223aR) 0565(0566bR|03457aR) 0566(0567bR|0280aR) 0567(0568bR|0360aR) 0568(0569bR|03171aR) 0569(0570bR|0385aR)
0570(0571bR|0357aR) 0571(0572bR|0239aR) 0572(0573bR|0302aR) 0573(0574bR|0234aR) 0574(0575bR|0289aR) 0575(0576bR|0381aR) 0576(0577bR|0304aR) 0577(0578bR|0385aR) 0578(0579bR|0308aR) 0579(0580bR|0381aR)
0580(0581bR|0304aR) 0581(0582bR|0354aR) 0582(0583bR|0276aR) 0583(0584bR|0408aR) 0584(0585bR|0225aR) 0585(0586bR|0224aR) 0586(0587bR|0242aR) 0587(0588bR|0358aR) 0588(0589bR|0358aR) 0589(0590bR|0367aR)
0590(0591bR|0317aR) 0591(0592bR|0385aR) 0592(0593bR|0382aR) 0593(0594bR|0251aR) 0594(0595bR|0198aR) 0595(0596bR|0268aR) 0596(0597bR|0278aR) 0597(0598bR|0312aR) 0598(0599bR|0369aR) 0599(0600bR|0369aR)
0600(0601bR|0301aR) 0601(0602bR|0366aR) 0602(0603bR|0289aR) 0603(0604bR|0310aR) 0604(0605bR|0275aR) 0605(0606bR|0233aR) 0606(0607bR|0374aR) 0607(0608bR|0345aR) 0608(0609bR|0328aR) 0609(0610bR|0
```

A description of a single state in Z

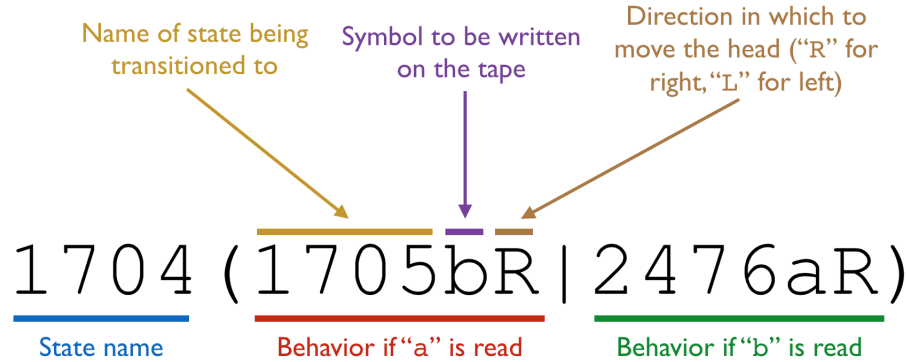


Figure 6: This figure explains how to read a description of a single state. Note that “ERROR--” or “HALT--” denote transitions to the ERROR or HALT states, respectively (no further information is provided because what symbol is written and which direction the head moves are at that point irrelevant).

0750 (0751bR 2191aR)	0751 (0752bR 2501aR)	0752 (0753bR 2556aR)	0753 (0754bR 2651aR)	0754 (0755bR 2210aR)	0755 (0756bR 2102aR)	0756 (0757bR 2999aR)	0757 (0758bR 2651aR)	0758 (0759bR 2218aR)	0759 (0760bR 3120aR)
0760 (0761bR 2555aR)	0761 (0762bR 3643aR)	0762 (0763bR 3341aR)	0763 (0764bR 2352aR)	0764 (0765bR 3883aR)	0765 (0766bR 3335aR)	0766 (0767bR 1986aR)	0767 (0768bR 3901aR)	0768 (0769bR 2230aR)	0769 (0770bR 2450aR)
0770 (0771bR 2722aR)	0771 (0772bR 2363aR)	0772 (0773bR 2220aR)	0773 (0774bR 2352aR)	0774 (0775bR 3298aR)	0775 (0776bR 3617aR)	0776 (0777bR 3764aR)	0777 (0778bR 3353aR)	0778 (0779bR 3893aR)	0779 (0780bR 2401aR)
0780 (0781bR 2028aR)	0781 (0782bR 3936aR)	0782 (0783bR 2735aR)	0783 (0784bR 2521aR)	0784 (0785bR 3438aR)	0785 (0786bR 3997aR)	0786 (0787bR 2744aR)	0787 (0788bR 2491aR)	0788 (0789bR 2040aR)	0789 (0800bR 2101aR)
0790 (0791bR 2813aR)	0791 (0792bR 3376aR)	0792 (0793bR 2215aR)	0793 (0794bR 3607aR)	0794 (0795bR 3291aR)	0795 (0796bR 3877aR)	0796 (0797bR 3532aR)	0797 (0798bR 2357aR)	0798 (0799bR 2996aR)	0799 (0800bR 3289aR)
0800 (0801bR 2891aR)	0801 (0802bR 3713aR)	0802 (0803bR 3771aR)	0803 (0804bR 3359aR)	0804 (0805bR 2047aR)	0805 (0806bR 2577aR)	0806 (0807bR 3203aR)	0807 (0808bR 3500aR)	0808 (0809bR 2923aR)	0809 (0810bR 3100aR)
0810 (0811bR 2210aR)	0811 (0812bR 3394aR)	0812 (0813bR 2813aR)	0813 (0814bR 3125aR)	0814 (0815bR 2891aR)	0815 (0816bR 3713aR)	0816 (0817bR 3532aR)	0817 (0818bR 2357aR)	0818 (0819bR 2996aR)	0819 (0820bR 3289aR)
0820 (0821bR 3715aR)	0821 (0822bR 3653aR)	0822 (0823bR 2928aR)	0823 (0824bR 3962aR)	0824 (0825bR 3226aR)	0825 (0826bR 3857aR)	0826 (0827bR 3532aR)	0827 (0828bR 3500aR)	0828 (0829bR 2218aR)	0829 (0830bR 2096aR)
0830 (0831bR 2996aR)	0831 (0832bR 3367aR)	0832 (0833bR 3476aR)	0833 (0834bR 3719aR)	0834 (0835bR 2891aR)	0835 (0836bR 3033aR)	0836 (0837bR 2960aR)	0837 (0838bR 3422aR)	0838 (0839bR 3964aR)	0839 (0840bR 3422aR)
0840 (0841bR 3180aR)	0841 (0842bR 3962aR)	0842 (0843bR 3216aR)	0843 (0844bR 2072aR)	0844 (0845bR 3173aR)	0845 (0846bR 2054aR)	0846 (0847bR 2124aR)	0847 (0848bR 3940aR)	0848 (0849bR 2032aR)	0849 (0850bR 3360aR)
0850 (0851bR 2690aR)	0851 (0852bR 2477aR)	0852 (0853bR 2544aR)	0853 (0854bR 2359aR)	0854 (0855bR 3296aR)	0855 (0856bR 2372aR)	0856 (0857bR 2762aR)	0857 (0858bR 3525aR)	0858 (0859bR 3824aR)	0859 (0860bR 2437aR)
0860 (0861bR 3043aR)	0861 (0862bR 3652aR)	0862 (0863bR 2754aR)	0863 (0864bR 3799aR)	0864 (0865bR 2172aR)	0865 (0866bR 3537aR)	0866 (0867bR 3069aR)	0867 (0868bR 2241aR)	0868 (0869bR 2760aR)	0869 (0870bR 3541aR)
0870 (0871bR 2127aR)	0871 (0872bR 2515aR)	0872 (0873bR 3660aR)	0873 (0874bR 2334aR)	0874 (0875bR 3008aR)	0875 (0876bR 2449aR)	0876 (0877bR 3696aR)	0877 (0878bR 2599aR)	0878 (0879bR 2760aR)	0879 (0880bR 3541aR)
0880 (0881bR 3842aR)	0881 (0882bR 2369aR)	0882 (0883bR 3660aR)	0883 (0884bR 3649aR)	0884 (0885bR 2672aR)	0885 (0886bR 3104aR)	0886 (0887bR 2740aR)	0887 (0888bR 2955aR)	0888 (0889bR 2702aR)	0889 (0890bR 2513aR)
0890 (0891bR 1996aR)	0891 (0892bR 2101aR)	0892 (0893bR 3660aR)	0893 (0894bR 2072aR)	0894 (0895bR 2544aR)	0895 (0896bR 3386aR)	0896 (0897bR 2112aR)	0897 (0898bR 3590aR)	0898 (0899bR 2031aR)	0899 (0900bR 2513aR)
0900 (0901bR 3019aR)	0901 (0902bR 2362aR)	0902 (0903bR 2124aR)	0903 (0904bR 2264aR)	0904 (0905bR 2228aR)	0905 (0906bR 3096aR)	0906 (0907bR 2672aR)	0907 (0908bR 3104aR)	0908 (0909bR 2742aR)	0909 (0910bR 2071aR)
0910 (0911bR 2032aR)	0911 (0912bR 3101aR)	0912 (0913bR 2749aR)	0913 (0914bR 2353aR)	0914 (0915bR 3019aR)	0915 (0916bR 3652aR)	0916 (0917bR 2754aR)	0917 (0918bR 3806aR)	0918 (0919bR 3043aR)	0919 (0920bR 2054aR)
0920 (0921bR 2045aR)	0921 (0922bR 2262aR)	0922 (0923bR 2156aR)	0923 (0924bR 2264aR)	0924 (0925bR 2173aR)	0925 (0926bR 3029aR)	0926 (0927bR 3532aR)	0927 (0928bR 2071aR)	0928 (0929bR 2687aR)	0929 (0930bR 2513aR)
0930 (0931bR 3019aR)	0931 (0932bR 2362aR)	0932 (0933bR 2124aR)	0933 (0934bR 2269aR)	0934 (0935bR 2228aR)	0935 (0936bR 3014aR)	0936 (0937bR 2156aR)	0937 (0938bR 2072aR)	0938 (0939bR 2557aR)	0939 (0940bR 2054aR)
0940 (0941bR 2030aR)	0941 (0942bR 2477aR)	0942 (0943bR 2531aR)	0943 (0944bR 2054aR)	0944 (0945bR 2156aR)	0945 (0946bR 2078aR)	0946 (0947bR 3043aR)	0947 (0948bR 3361aR)	0948 (0949bR 2228aR)	0949 (0950bR 2261aR)
0950 (0951bR 3150aR)	0951 (0952bR 3462aR)	0952 (0953bR 2028aR)	0953 (0954bR 2264aR)	0954 (0955bR 2685aR)	0955 (0956bR 3013aR)	0956 (0957bR 3532aR)	0957 (0958bR 3357aR)	0958 (0959bR 2698aR)	0959 (0960bR 3454aR)
0960 (0961bR 2027aR)	0961 (0962bR 2336aR)	0962 (0963bR 2690aR)	0963 (0964bR 2373aR)	0964 (0965bR 2968aR)	0965 (0966bR 2076aR)	0966 (0967bR 2178aR)	0967 (0968bR 2453aR)	0968 (0969bR 3043aR)	0969 (0970bR 3653aR)
0970 (0971bR 2674aR)	0971 (0972bR 2393aR)	0972 (0973bR 2720aR)	0973 (0974bR 3864aR)	0974 (0975bR 2178aR)	0975 (0976bR 3862aR)	0976 (0977bR 2723aR)	0977 (0978bR 1990aR)	0978 (0979bR 2028aR)	0979 (0980bR 2264aR)
0980 (0981bR 2685aR)	0981 (0982bR 2054aR)	0982 (0983bR 2028aR)	0983 (0984bR 2014aR)	0984 (0985bR 3008aR)	0985 (0986bR 3361aR)	0986 (0987bR 2176aR)	0987 (0988bR 3496aR)	0988 (0989bR 2554aR)	0989 (0990bR 3396aR)
0990 (0991bR 2973aR)	0991 (0992bR 2600aR)	0992 (0993bR 2124aR)	0993 (0994bR 3941aR)	0994 (0995bR 2813aR)	0995 (0996bR 3696aR)	0996 (0997bR 3018aR)	0997 (0998bR 3624aR)	0998 (0999bR 2675aR)	0999 (1000bR 3754aR)
1000 (1001bR 2819aR)	1001 (1002bR 2097aR)	1002 (1003bR 3307aR)	1003 (1004bR 2478aR)	1004 (1005bR 3316aR)	1005 (1006bR 2252aR)	1006 (1007bR 2962aR)	1007 (1008bR 3422aR)	1008 (1009bR 3859aR)	1009 (1010bR 2516aR)
1010 (1011bR 2813aR)	1011 (1012bR 2329aR)	1012 (1013bR 3893aR)	1013 (1014bR 2401aR)	1014 (1015bR 2675aR)	1015 (1016bR 3669aR)	1016 (1017bR 3255aR)	1017 (1018bR 3624aR)	1018 (1019bR 2035aR)	1019 (1020bR 3669aR)
1020 (1021bR 3255aR)	1021 (1022bR 2650aR)	1022 (1023bR 2163aR)	1023 (1024bR 3664aR)	1024 (1025bR 2732aR)	1025 (1026bR 2416aR)	1026 (1027bR 2883aR)	1027 (1028bR 3130aR)	1028 (1029bR 2531aR)	1029 (1030bR 2376aR)
1030 (1031bR 3430aR)	1031 (1032bR 3269aR)	1032 (1033bR 3535aR)	1033 (1034bR 3668aR)	1034 (1035bR 2702aR)	1035 (1036bR 3550aR)	1036 (1037bR 3176aR)	1037 (1038bR 3638aR)	1038 (1039bR 2231aR)	1039 (1040bR 2651aR)
1040 (1041bR 2200aR)	1041 (1042bR 2453aR)	1042 (1043bR 3069aR)	1043 (1044bR 3127aR)	1044 (1045bR 3754aR)	1045 (1046bR 4013aR)	1046 (1047bR 2880aR)	1047 (1048bR 3863aR)	1048 (1049bR 2230aR)	1049 (1050bR 3487aR)
1050 (1051bR 2922aR)	1051 (1052bR 3962aR)	1052 (1053bR 3218aR)	1053 (1054bR 2998aR)	1054 (1055bR 3146aR)	1055 (1056bR 3639aR)	1056 (1057bR 3304aR)	1057 (1058bR 2398aR)	1058 (1059bR 3859aR)	1059 (1060bR 2845aR)
1060 (1061bR 2552aR)	1061 (1062bR 3307aR)	1062 (1063bR 3406aR)	1063 (1064bR 2263aR)	1064 (1065bR 2803aR)	1065 (1066bR 3668aR)	1066 (1067bR 2223aR)	1067 (1068bR 2921aR)	1068 (1069bR 4206aR)	1069 (1070bR 2078aR)
1070 (1071bR 3875aR)	1071 (1072bR 3357aR)	1072 (1073bR 3832aR)	1073 (1074bR 2107aR)	1074 (1075bR 3341aR)	1075 (1076bR 2070aR)	1076 (1077bR 2941aR)	1077 (1078bR 2069aR)	1078 (1079bR 2231aR)	1079 (1080bR 2651aR)
1080 (1081bR 2730aR)	1081 (1082bR 3548aR)	1082 (1083bR 2237aR)	1083 (1084bR 2325aR)	1084 (1085bR 3264aR)	1085 (1086bR 2101aR)	1086 (1087bR 2894aR)	1087 (1088bR 2110aR)	1088 (1089bR 3178aR)	1089 (1090bR 3643aR)
1090 (1091bR 3341aR)	1091 (1092bR 3124aR)	1092 (1093bR 3614aR)	1093 (1094bR 3614aR)	1094 (1095bR 3875aR)	1095 (1096bR 3100aR)	1096 (1097bR 2557aR)	1097 (1098bR 3126aR)	1098 (1099bR 2927aR)	1099 (1100bR 2781aR)
1100 (1101bR 2607aR)	1101 (1102bR 3127aR)	1102 (1103bR 2991aR)	1103 (1104bR 2781aR)	1104 (1105bR 2237aR)	1105 (1106bR 2069aR)	1106 (1107bR 2973aR)	1107 (1108bR 3696aR)	1108 (1109bR 3199aR)	1109 (1110bR 3399aR)
1110 (1111bR 2740aR)	1111 (1112bR 2285aR)	1112 (1113bR 2941aR)	1113 (1114bR 2055aR)	1114 (1115bR 2191aR)	1115 (1116bR 3656aR)	1116 (1117bR 3704aR)	1117 (1118bR 3614aR)	1118 (1119bR 3694aR)	1119 (1120bR 2097aR)
1120 (1121bR 2923aR)	1121 (1122bR 3936aR)	1122 (1123bR 2158aR)	1123 (1124bR 3612aR)	1124 (1125bR 2927aR)	1125 (1126bR 2513aR)	1126 (1127bR 3184aR)	1127 (1128bR 3937aR)	1128 (1129bR 2811aR)	1129 (1130bR 3497aR)
1130 (1131bR 1999aR)	1131 (1132bR 3415aR)	1132 (1133bR 2228aR)	1133 (1134bR 2264aR)	1134 (1135bR 2927aR)	1135 (1136bR 2775aR)	1136 (1137bR 2043aR)	1137 (1138bR 3937aR)	1138 (1139bR 3993aR)	1139 (1140bR 2400aR)
1140 (1141bR 2124aR)	1141 (1142bR 3550aR)	1142 (1143bR 3195aR)	1143 (1144bR 2107aR)	1144 (1145bR 3341aR)	1145 (1146bR 2077aR)	1146 (1147bR 3842aR)	1147 (1148bR 2397aR)	1148 (1149bR 3893aR)	1149 (1150bR 2401aR)
1150 (1151bR 2220aR)	1151 (1152bR 3936aR)	1152 (1153bR 2767aR)	1153 (1154bR 2500aR)	1154 (1155bR 2187aR)	1155 (1156bR 3399aR)	1156 (1157bR 3272aR)	1157 (1158bR 3621aR)	1158 (1159bR 3893aR)	1159 (1160bR 2401aR)
1160 (1161bR 2813aR)	1161 (1162bR 3381aR)	1162 (1163bR 2754aR)	1163 (1164bR 3777aR)	1164 (1165bR 3811aR)	1165 (1166bR 3511aR)	1166 (1167bR 2046aR)	1167 (1168bR 3399aR)	1168 (1169bR 3434aR)	1169 (1170bR 3364aR)
1170 (1171bR 2880aR)	1171 (1172bR 3952aR)	1172 (1173bR 3783aR)	1173 (1174bR 3783aR)	1174 (1175bR 3696aR)	1175 (1176bR 3640aR)	1176 (1177bR 2163aR)	1177 (1178bR 3665aR)	1178 (1179bR 2702aR)	1179 (1180bR 3953aR)
1180 (1181bR 3534aR)	1181 (1182bR 3386aR)	1182 (1183bR 2024aR)	1183 (1184bR 3873aR)	1184 (1185bR 3883aR)	1185 (1186bR 3359aR)	1186 (1187bR 2042aR)	1187 (1188bR 3496aR)	1188 (1189bR 2123aR)	1189 (1190bR 2473aR)
1190 (1191bR 2156aR)	1191 (1192bR 3899aR)	1192 (1193bR 2208aR)	1193 (1194bR 3605aR)	1194 (1195bR 2156aR)	1195 (1196bR 3662aR)	1196 (1197bR 2230aR)	1197 (1198bR 2433aR)	1198 (1199bR 3179aR)	1199 (1200bR 3713aR)
1200 (1201bR 3692aR)	1201 (1202bR 2096aR)	1202 (1203bR 2230aR)	1203 (1204bR 2103aR)	1204 (1205bR 3744aR)	1205 (1206bR 3999aR)	1206 (1207bR 3994aR)	1207 (1208bR 2311aR)	1208 (1209bR 2767aR)	1209 (1210bR 2845aR)
1210 (1211bR 2813aR)	1211 (1212bR 3783aR)	1212 (1213bR 3840aR)	1213 (1214bR 3097aR)	1214 (1215bR 3875aR)	1215 (1216bR 3037aR)	1216 (1217bR 2740aR)	1217 (1218bR 3286aR)	1218 (1219bR 2928aR)	1219 (1220bR 3962aR)
1220 (1221bR 3224aR)	1221 (1222bR 3777aR)	1222 (1223bR 3875aR)	1223 (1224bR 3423aR)	1224 (1225bR 3476aR)	1225 (1226bR 3719aR)	1226 (1227bR 2638aR)	1227 (1228bR 2085aR)	1228 (1229bR 3883aR)	1229 (1230bR 3329aR)
1230 (1231bR 3630aR)	1231 (1232bR 3714aR)	1232 (1233bR 2670aR)	1233 (1234bR 2078aR)	1234 (1235bR 3859aR)	1235 (1236bR 2513aR)	1236 (1237bR 3247aR)	1237 (1238bR 2581aR)	1238 (1239bR 2813aR)	1239 (1240bR 3686aR)
1240 (1241bR 3040aR)	1241 (1242bR 3486aR)	1242 (1243bR 3043aR)	1243 (1244bR 2462aR						

1490 (1491br|2802ar) 1491 (1492br|3805ar) 1492 (1493br|2962ar) 1493 (1494br|2262ar) 1494 (1495br|2928ar) 1495 (1496br|3612ar) 1496 (1497br|2927ar) 1497 (1498br|2921ar) 1498 (1499br|3272ar) 1499 (1500br|3639ar) 1500 (1501br|3306ar) 1501 (1502br|3643ar) 1502 (1503br|3341ar) 1503 (1504br|2245ar) 1504 (1505br|2769ar) 1505 (1506br|3612ar) 1506 (1507br|2927ar) 1507 (1508br|2081ar) 1508 (1509br|3964ar) 1509 (1510br|3125ar) 1510 (1511br|3191ar) 1511 (1512br|2963ar) 1512 (1513br|3896ar) 1513 (1514br|3178ar) 1514 (1515br|2219ar) 1515 (1516br|3639ar) 1516 (1517br|2927ar) 1517 (1518br|3643ar) 1518 (1519br|3341ar) 1519 (1520br|2245ar) 1520 (1521br|2769ar) 1521 (1522br|2680ar) 1522 (1523br|2385ar) 1523 (1524br|3285ar) 1524 (1525br|3612ar) 1525 (1526br|3964ar) 1526 (1527br|2163ar) 1527 (1528br|3191ar) 1528 (1529br|3125ar) 1529 (1530br|3639ar) 1530 (1531br|2173ar) 1531 (1532br|3366ar) 1532 (1533br|2186ar) 1533 (1534br|3862ar) 1534 (1535br|2869ar) 1535 (1536br|3489ar) 1536 (1537br|2766ar) 1537 (1538br|3393ar) 1538 (1539br|3055ar) 1539 (1540br|2664ar) 1540 (1541br|2691ar) 1541 (1542br|2999ar) 1542 (1543br|3592ar) 1543 (1544br|3922ar) 1544 (1545br|3643ar) 1545 (1546br|3178ar) 1546 (1547br|3643ar) 1547 (1548br|3964ar) 1548 (1549br|3125ar) 1549 (1550br|3639ar) 1550 (1551br|3252ar) 1551 (1552br|2268ar) 1552 (1553br|2941ar) 1553 (1554br|3862ar) 1554 (1555br|2924ar) 1555 (1556br|3463ar) 1556 (1557br|3372ar) 1557 (1558br|3393ar) 1558 (1559br|3055ar) 1559 (1560br|2664ar) 1560 (1561br|2703ar) 1561 (1562br|2844ar) 1562 (1563br|2045ar) 1563 (1564br|3037ar) 1564 (1565br|2552ar) 1565 (1566br|2455ar) 1566 (1567br|3438ar) 1567 (1568br|3292ar) 1568 (1569br|3046ar) 1569 (1570br|3381ar) 1570 (1571br|2173ar) 1571 (1572br|3366ar) 1572 (1573br|2186ar) 1573 (1574br|3862ar) 1574 (1575br|2869ar) 1575 (1576br|3489ar) 1576 (1577br|2766ar) 1577 (1578br|3393ar) 1578 (1579br|3055ar) 1579 (1580br|2664ar) 1580 (1581br|2703ar) 1581 (1582br|2844ar) 1582 (1583br|2045ar) 1583 (1584br|3037ar) 1584 (1585br|2552ar) 1585 (1586br|2455ar) 1586 (1587br|3438ar) 1587 (1588br|3292ar) 1588 (1589br|3046ar) 1589 (1590br|3381ar) 1590 (1591br|2173ar) 1591 (1592br|3366ar) 1592 (1593br|2186ar) 1593 (1594br|3862ar) 1594 (1595br|2869ar) 1595 (1596br|3489ar) 1596 (1597br|2766ar) 1597 (1598br|3393ar) 1598 (1599br|3055ar) 1599 (1600br|2651ar) 1600 (1601br|2691ar) 1601 (1602br|2999ar) 1602 (1603br|3592ar) 1603 (1604br|3922ar) 1604 (1605br|3643ar) 1605 (1606br|3178ar) 1606 (1607br|3643ar) 1607 (1608br|3964ar) 1608 (1609br|3125ar) 1609 (1610br|3639ar) 1610 (1611br|3252ar) 1611 (1612br|2268ar) 1612 (1613br|2941ar) 1613 (1614br|3862ar) 1614 (1615br|2924ar) 1615 (1616br|3463ar) 1616 (1617br|3372ar) 1617 (1618br|3393ar) 1618 (1619br|3055ar) 1619 (1620br|2664ar) 1620 (1621br|2703ar) 1621 (1622br|2844ar) 1622 (1623br|2045ar) 1623 (1624br|3037ar) 1624 (1625br|2552ar) 1625 (1626br|2455ar) 1626 (1627br|3438ar) 1627 (1628br|3292ar) 1628 (1629br|3046ar) 1629 (1630br|3381ar) 1630 (1631br|2173ar) 1631 (1632br|3366ar) 1632 (1633br|2186ar) 1633 (1634br|3862ar) 1634 (1635br|2869ar) 1635 (1636br|3489ar) 1636 (1637br|2766ar) 1637 (1638br|3393ar) 1638 (1639br|3055ar) 1639 (1640br|2664ar) 1640 (1641br|2703ar) 1641 (1642br|2844ar) 1642 (1643br|2045ar) 1643 (1644br|3037ar) 1644 (1645br|2552ar) 1645 (1646br|2455ar) 1646 (1647br|3438ar) 1647 (1648br|3292ar) 1648 (1649br|3046ar) 1649 (1650br|3381ar) 1650 (1651br|2173ar) 1651 (1652br|3366ar) 1652 (1653br|2186ar) 1653 (1654br|3862ar) 1654 (1655br|2869ar) 1655 (1656br|3489ar) 1656 (1657br|2766ar) 1657 (1658br|3393ar) 1658 (1659br|3055ar) 1659 (1660br|2651ar) 1660 (1661br|2691ar) 1661 (1662br|2999ar) 1662 (1663br|3592ar) 1663 (1664br|3922ar) 1664 (1665br|3643ar) 1665 (1666br|3178ar) 1666 (1667br|3643ar) 1667 (1668br|3964ar) 1668 (1669br|3125ar) 1669 (1670br|3639ar) 1670 (1671br|3252ar) 1671 (1672br|2268ar) 1672 (1673br|2941ar) 1673 (1674br|3862ar) 1674 (1675br|2924ar) 1675 (1676br|3463ar) 1676 (1677br|3372ar) 1677 (1678br|3393ar) 1678 (1679br|3055ar) 1679 (1680br|2664ar) 1680 (1681br|2703ar) 1681 (1682br|2844ar) 1682 (1683br|2045ar) 1683 (1684br|3037ar) 1684 (1685br|2552ar) 1685 (1686br|2455ar) 1686 (1687br|3438ar) 1687 (1688br|3292ar) 1688 (1689br|3046ar) 1689 (1690br|3381ar) 1690 (1691br|2173ar) 1691 (1692br|3366ar) 1692 (1693br|2186ar) 1693 (1694br|3862ar) 1694 (1695br|2869ar) 1695 (1696br|3489ar) 1696 (1697br|2766ar) 1697 (1698br|3393ar) 1698 (1699br|3055ar) 1699 (1700br|3940ar) 1700 (1701br|2186ar) 1701 (1702br|2462ar) 1702 (1703br|3859ar) 1703 (1704br|2518ar) 1704 (1705br|2970ar) 1705 (1706br|3887ar) 1706 (1707br|2880ar) 1707 (1708br|3957ar) 1708 (1709br|3628ar) 1709 (1710br|3777ar) 1710 (1711br|3770ar) 1711 (1712br|3496ar) 1712 (1713br|2035ar) 1713 (1714br|3719ar) 1714 (1715br|2043ar) 1715 (1716br|3653ar) 1716 (1717br|2811ar) 1717 (1718br|3754ar) 1718 (1719br|2824ar) 1719 (1720br|3037ar) 1720 (1721br|3058ar) 1721 (1722br|3937ar) 1722 (1723br|2187ar) 1723 (1724br|3496ar) 1724 (1725br|2552ar) 1725 (1726br|3642ar) 1726 (1727br|2542ar) 1727 (1728br|3462ar) 1728 (1729br|2742ar) 1729 (1730br|3376ar) 1730 (1731br|2891ar) 1731 (1732br|3359ar) 1732 (1733br|3272ar) 1733 (1734br|3608ar) 1734 (1735br|2968ar) 1735 (1736br|3463ar) 1736 (1737br|3274ar) 1737 (1738br|3192ar) 1738 (1739br|2924ar) 1739 (1740br|3937ar) 1740 (1741br|2179ar) 1741 (1742br|3386ar) 1742 (1743br|1994ar) 1743 (1744br|3877ar) 1744 (1745br|3578ar) 1745 (1746br|3963ar) 1746 (1747br|2883ar) 1747 (1748br|3477ar) 1748 (1749br|3767ar) 1749 (1750br|2604ar) 1750 (1751br|1994ar) 1751 (1752br|2457ar) 1752 (1753br|3184ar) 1753 (1754br|2373ar) 1754 (1755br|3151ar) 1755 (1756br|2513ar) 1756 (1757br|3020ar) 1757 (1758br|3500ar) 1758 (1759br|2040ar) 1759 (1760br|2432ar) 1760 (1761br|2923ar) 1761 (1762br|3941ar) 1762 (1763br|2156ar) 1763 (1764br|3877ar) 1764 (1765br|2968ar) 1765 (1766br|3963ar) 1766 (1767br|2960ar) 1767 (1768br|3077ar) 1768 (1769br|3559ar) 1769 (1770br|3104ar) 1770 (1771br|2763ar) 1771 (1772br|2270ar) 1772 (1773br|3043ar) 1773 (1774br|2012ar) 1774 (1775br|2968ar) 1775 (1776br|2968ar) 1776 (1777br|3842ar) 1777 (1778br|3863ar) 1778 (1779br|3863ar) 1779 (1780br|2299ar) 1780 (1781br|2892ar) 1781 (1782br|2081ar) 1782 (1783br|2162ar) 1783 (1784br|3668ar) 1784 (1785br|2440ar) 1785 (1786br|3877ar) 1786 (1787br|3043ar) 1787 (1788br|2722ar) 1788 (1789br|2960ar) 1789 (1790br|3179ar) 1790 (1791br|3148ar) 1791 (1792br|2081ar) 1792 (1793br|2968ar) 1793 (1794br|3120ar) 1794 (1795br|2968ar) 1795 (1796br|3863ar) 1796 (1797br|3842ar) 1797 (1798br|3729ar) 1798 (1799br|3179ar) 1799 (1800br|2054ar) 1800 (1801br|2168ar) 1801 (1802br|3936ar) 1802 (1803br|2019ar) 1803 (1804br|2264ar) 1804 (1805br|2178ar) 1805 (1806br|3593ar) 1806 (1807br|2208ar) 1807 (1808br|3858ar) 1808 (1809br|3695ar) 1809 (1810br|3671ar) 1810 (1811br|2680ar) 1811 (1812br|3066ar) 1812 (1813br|3862ar) 1813 (1814br|3862ar) 1814 (1815br|2992ar) 1815 (1816br|3784ar) 1816 (1817br|3712ar) 1817 (1818br|3172ar) 1818 (1819br|3712ar) 1819 (1820br|3172ar) 1820 (1821br|2927ar) 1821 (1822br|2567ar) 1822 (1823br|2220ar) 1823 (1824br|3608ar) 1824 (1825br|2178ar) 1825 (1826br|3591ar) 1826 (1827br|3329ar) 1827 (1828br|2634ar) 1828 (1829br|3928ar) 1829 (1830br|3496ar) 1830 (1831br|2019ar) 1831 (1832br|3360ar) 1832 (1833br|2973ar) 1833 (1834br|3862ar) 1834 (1835br|2019ar) 1835 (1836br|2391ar) 1836 (1837br|3298ar) 1837 (1838br|2343ar) 1838 (1839br|2167ar) 1839 (1840br|3463ar) 1840 (1841br|2743ar) 1841 (1842br|3366ar) 1842 (1843br|3964ar) 1843 (1844br|3964ar) 1844 (1845br|2973ar) 1845 (1846br|3628ar) 1846 (1847br|2973ar) 1847 (1848br|2601ar) 1848 (1849br|1987ar) 1849 (1850br|3914ar) 1850 (1851br|2763ar) 1851 (1852br|3646ar) 1852 (1853br|2636ar) 1853 (1854br|2457ar) 1854 (1855br|2755ar) 1855 (1856br|2391ar) 1856 (1857br|2228ar) 1857 (1858br|3499ar) 1858 (1859br|2184ar) 1859 (1860br|3712ar) 1860 (1861br|2885ar) 1861 (1862br|2269ar) 1862 (1863br|2749ar) 1863 (1864br|2000ar) 1864 (1865br|2700ar) 1865 (1866br|2438ar) 1866 (1867br|2032ar) 1867 (1868br|3499ar) 1868 (1869br|3850ar) 1869 (1870br|3254ar) 1870 (1871br|2892ar) 1871 (1872br|2081ar) 1872 (1873br|2968ar) 1873 (1874br|3668ar) 1874 (1875br|2032ar) 1875 (1876br|3877ar) 1876 (1877br|3043ar) 1877 (1878br|2722ar) 1878 (1879br|2960ar) 1879 (1880br|3179ar) 1880 (1881br|2766ar) 1881 (1882br|3361ar) 1882 (1883br|3893ar) 1883 (1884br|2401ar) 1884 (1885br|2178ar) 1885 (1886br|3593ar) 1886 (1887br|2552ar) 1887 (1888br|3863ar) 1888 (1889br|3148ar) 1889 (1890br|2054ar) 1890 (1891br|2722ar) 1891 (1892br|3652ar) 1892 (1893br|2766ar) 1893 (1894br|3502ar) 1894 (1895br|3316ar) 1895 (1896br|2343ar) 1896 (1897br|3438ar) 1897 (1898br|2014ar) 1898 (1899br|3549ar) 1899 (1900br|3292ar) 1900 (1901br|2218ar) 1901 (1902br|2973ar) 1902 (1903br|3862ar) 1903 (1904br|3862ar) 1904 (1905br|2992ar) 1905 (1906br|3784ar) 1906 (1907br|3712ar) 1907 (1908br|3172ar) 1908 (1909br|3712ar) 1909 (1910br|2680ar) 1910 (1911br|2680ar) 1911 (1912br|3376ar) 1912 (1913br|2891ar) 1913 (1914br|3602ar) 1914 (1915br|2984ar) 1915 (1916br|3496ar) 1916 (1917br|2123ar) 1917 (1918br|2476ar) 1918 (1919br|2883ar) 1919 (1920br|3386ar) 1920 (1921br|1996ar) 1921 (1922br|3953ar) 1922 (1923br|3534ar) 1923 (1924br|3800ar) 1924 (1925br|2868ar) 1925 (1926br|3941ar) 1926 (1927br|2163ar) 1927 (1928br|3399ar) 1928 (1929br|3476ar) 1929 (1930br|3649ar) 1930 (1931br|2500ar) 1931 (1932br|2973ar) 1932 (1933br|3862ar) 1933 (1934br|3862ar) 1934 (1935br|2992ar) 1935 (1936br|3784ar) 1936 (1937br|3712ar) 1937 (1938br|3172ar) 1938 (1939br|3712ar) 1939 (1940br|2680ar) 1940 (1941br|2680ar) 1941 (1942br|3781ar) 1942 (1943br|3781ar) 1943 (1944br|3781ar) 1944 (1945br|3781ar) 1945 (1946br|3781ar) 1946 (1947br|3781ar) 1947 (1948br|3781ar) 1948 (1949br|3781ar) 1949 (1950br|2268ar) 1950 (1951br|2045ar) 1951 (1952br|2055ar) 1952 (1953br|1996ar) 1953 (1954br|3963ar) 1954 (1955br|2126ar) 1955 (1956br|2072ar) 1956 (1957br|2962ar) 1957 (1958br|2533ar) 1958 (1959br|2208ar) 1959 (1960br|3456ar) 1960 (1961br|2970ar) 1961 (1962br|3500ar) 1962 (1963br|2970ar) 1963 (1964br|3500ar) 1964 (1965br|2970ar) 1965 (1966br|3500ar) 1966 (1967br|2970ar) 1967 (1968br|3500ar) 1968 (1969br|2970ar) 1969 (1970br|3500ar) 1970 (1971br|3183ar) 1971 (1972br|3077ar) 1972 (1973br|2513ar) 1973 (1974br|3106ar) 1974 (1975br|2126ar) 1975 (1976br|2268ar) 1976 (1977br|2168ar) 1977 (1978br|2168ar) 1978 (1979br|2168ar) 1979 (1980br|2168ar) 1980 (1981br|2542ar) 1981 (1982br|3390ar) 1982 (1983br|3010ar) 1983 (1984br|2433ar) 1984 (1985br|3652ar) 1985 (1986br|3652ar) 1986 (1987br|2880ar) 1987 (1988br|3733ar) 1988 (1989br|3733ar) 1989 (1990br|3781ar) 1990 (1991br|3020ar) 1991 (1992br|2078ar) 1992 (1993br|2078ar) 1993 (1994br|2078ar) 1994 (1995br|2078ar) 1995 (1996br|2078ar) 1996 (1997br|2078ar) 1997 (1998br|2078ar) 1998 (1999br|2078ar) 1999 (2000br|2078ar) 2000 (2001br|2078ar) 2001 (2002br|2078ar) 2002 (2003br|2078ar) 2003 (2004br|2078ar) 2004 (2005br|2078ar) 2005 (2006br|2078ar) 2006 (2007br|2078ar) 2007 (2008br|2078ar) 2008 (2009br|2078ar) 2009 (2010br|2078ar) 2010 (2011br|2078ar) 2011 (2012br|2078ar) 2012 (2013br|2078ar) 2013 (2014br|2078ar) 2014 (2015br|2078ar) 2015 (2016br|2078ar) 2016 (2017br|2078ar) 2017 (2018br|2078ar) 2018 (2019br|2078ar) 2019 (2020br|2078ar) 2020 (2021br|2078ar) 2021 (2022br|2078ar) 2022 (2023br|2078ar) 2023 (2024br|2078ar) 2024 (2025br|2078ar) 2025 (2026br|2078ar) 2026 (2027br|2078ar) 2027 (2028br|2078ar) 2028 (2029br|2078ar) 2029 (2030br|2078ar) 2030 (2031br|2078ar) 2031 (2032br|2078ar) 2032 (2033br|2078ar) 2033 (2034br|2078ar) 2034 (2035br|2078ar) 2035 (2036br|2078ar) 2036 (2037br|2078ar) 2037 (2038br|2078ar) 2038 (2039br|2078ar) 2039 (2040br|2078ar) 2040 (2041br|2078ar) 2041 (2042br|2078ar) 2042 (2043br|2078ar) 2043 (2044br|2078ar) 2044 (2045br|2078ar) 2045 (2046br|2078ar) 2046 (2047br|2078ar) 2047 (2048br|2078ar) 2048 (2049br|2078ar) 2049 (2050br|2078ar) 2050 (2051br|2078ar) 2051 (2052br|2078ar) 2052 (2053br|2078ar) 2053 (2054br|2078ar) 2054 (2055br|2078ar) 2055 (2056br|2078ar) 2056 (2057br|2078ar) 2057 (2058br|2078ar) 2058 (2059br|2078ar) 2059 (2060br|2078ar) 2060 (2061br|2078ar) 2061 (2062br|2078ar) 2062 (2063br|2078ar) 2063 (2064br|2078ar) 2064 (2065br|2078ar) 2065 (2066br|2078ar) 2066 (2067br|2078ar) 2067 (2068br|2078ar) 2068 (2069br|2078ar) 2069 (2070br|2078ar) 2070 (2071br|2078ar) 2071 (2072br|2078ar) 2072 (2073br|2078ar) 2073 (2074br|2078ar) 2074 (2075br|2078ar) 2075 (2076br|2078ar) 2076 (2077br|2078ar) 2077 (2078br|2078ar) 2078 (2079br|2078ar) 2079 (2080br|2078ar) 2080 (2081br|2078ar) 2081 (2082br|2078ar) 2082 (2083br|2078ar) 2083 (2084br|2078ar) 2084 (2085br|2078ar) 2085 (2086br|2078ar) 2086 (2087br|2078ar) 2087 (2088br|2078ar) 2088 (2089br|2078ar) 2089 (2090br|2078ar) 2090 (2091br|2078ar) 2091 (2092br|2078ar) 2092 (2093br|2078ar) 2093 (2094br|2078ar) 2094 (2095br|2078ar) 2095 (2096br|2078ar) 2096 (2097br|2078ar) 2097 (2098br|2078ar) 2098 (2099br|2078ar) 2099 (2100br|2078ar) 2100 (2101br|2078ar) 2101 (2102br|2078ar) 2102 (2103br|2078ar) 2103 (2104br|2078ar) 2104 (2105br|2078ar) 2105 (2106br|2078ar) 2106 (2107br|2078ar) 2107 (2108br|2078ar) 2108 (2109br|2078ar) 2109 (2110br|2078ar) 2110 (2111br|2078ar) 2111 (2112br|2078ar) 2112 (2113br|2078ar) 2113 (2114br|2078ar) 2114 (2115br|2078ar) 2115 (2116br|2078ar) 2116 (2117br|2078ar) 2117 (2118br|2078ar) 2118 (2119br|2078ar) 2119 (2120br|2078ar) 2120 (2121br|2078ar) 2121 (2122br|2078ar) 2122 (2123br|2078ar) 2123 (2124br|2078ar) 2124 (2125br|2078ar) 2125 (2126br|2078ar) 2126 (2127br|2078ar) 2127 (2128br|2078ar) 2128 (2129br|2078ar) 2129 (2130br|2078ar) 2130 (2131br|2078ar) 2131 (2132br|2078ar) 2132 (2133br|2078ar) 2133 (2134br|2078ar) 2134 (2135br|2078ar) 2135 (2136br|2078ar) 2136 (2137br|2078ar) 2137 (2138br|2078ar) 2138 (2139br|2078ar) 2139 (2140br|2078ar) 2140 (2141br|2078ar) 2141 (2142br|2078ar) 2142 (2143br|2078ar) 2143 (2144br|2078ar) 2144 (2145br|2078ar) 2145 (2146br|2078ar) 2146 (2147br|2078ar) 2147 (2148br|2078ar) 2148 (2149br|2078ar) 2149 (2150br|2078ar) 2150 (2151br|2078ar) 2151 (2152br|2078ar) 2152 (2153br|2078ar) 2153 (2154br|2078ar) 2154 (2155br|2078ar) 2155 (2156br|2078ar) 2156 (2157br|2078ar) 2157 (2158br|2078ar) 2158 (2159br|2078ar) 2159 (2160br|2078ar) 2160 (2161br|2078ar) 2161 (2162br|2078ar) 2162 (2163br|2078ar) 2163 (2164br|2078ar) 2164 (2165br|2078ar) 2165 (2166br|2078ar) 2166 (2167br|2078ar) 2167 (2168br|2078ar) 2168 (2169br|2078ar) 2169 (2170br|2078ar) 2170 (2171br|2078ar) 2171 (2172br|2078ar) 2172 (2173br|2078ar) 2173 (2174br|2078ar) 2174 (2175br|2078ar) 2175 (2176br|2078ar) 2176 (2177

3020 (3021br | 2040ar) 3021 (3022br | 2453ar) 3022 (3023br | 3984ar) 3023 (3024br | 12245ar) 3024 (3025br | 3715ar) 3025 (3026br | 2009ar) 3026 (3027br | 2186ar) 3027 (3028br | 3783ar) 3028 (3029br | 3296ar) 3029 (3030br | 2265ar) 3030 (3031br | 3020ar) 3031 (3032br | 3463ar) 3032 (3033br | 2690ar) 3033 (3034br | 3590ar) 3034 (3035br | 2874ar) 3035 (3036br | 2472ar) 3036 (3037br | 2546ar) 3037 (3038br | 2376ar) 3038 (3039br | 3874ar) 3039 (3040br | 2433ar) 3040 (3041br | 3599ar) 3041 (3042br | 3199ar) 3042 (3043br | 2796ar) 3043 (3044br | 3038ar) 3044 (3045br | 3555ar) 3045 (3046br | 3694ar) 3046 (3047br | 3639ar) 3047 (3048br | 2124ar) 3048 (3049br | 3504ar) 3049 (3050br | 2970ar) 3050 (3051br | 3694ar) 3051 (3052br | 3936ar) 3052 (3053br | 2124ar) 3053 (3054br | 3340ar) 3054 (3055br | 2873ar) 3055 (3056br | 2600ar) 3056 (3057br | 2019ar) 3057 (3058br | 2360ar) 3058 (3059br | 3874ar) 3059 (3060br | 3009ar) 3060 (3061br | 3244ar) 3061 (3062br | 2369ar) 3062 (3063br | 3052ar) 3063 (3064br | 3130ar) 3064 (3065br | 2124ar) 3065 (3066br | 2081ar) 3066 (3067br | 3872ar) 3067 (3068br | 2433ar) 3068 (3069br | 3706ar) 3069 (3070br | 3515ar) 3070 (3071br | 3699ar) 3071 (3072br | 3778ar) 3072 (3073br | 3592ar) 3073 (3074br | 3733ar) 3074 (3075br | 3592ar) 3075 (3076br | 3733ar) 3076 (3077br | 3592ar) 3077 (3078br | 3733ar) 3078 (3079br | 3592ar) 3079 (3080br | 3733ar) 3080 (3081br | 2126ar) 3081 (3082br | 3390ar) 3082 (3083br | 3136ar) 3083 (3084br | 3367ar) 3084 (3085br | 2764ar) 3085 (3086br | 3462ar) 3086 (3087br | 2302ar) 3087 (3088br | 2430ar) 3088 (3089br | 3706ar) 3089 (3090br | 2369ar) 3090 (3091br | 3197ar) 3091 (3092br | 2054ar) 3092 (3093br | 2168ar) 3093 (3094br | 3936ar) 3094 (3095br | 2155ar) 3095 (3096br | 2264ar) 3096 (3097br | 2178ar) 3097 (3098br | 3953ar) 3098 (3099br | 2208ar) 3099 (3100br | 3521ar) 3100 (3101br | 3699ar) 3101 (3102br | 3778ar) 3102 (3103br | 3592ar) 3103 (3104br | 3733ar) 3104 (3105br | 3592ar) 3105 (3106br | 3733ar) 3106 (3107br | 3592ar) 3107 (3108br | 3733ar) 3108 (3109br | 3592ar) 3109 (3110br | 3733ar) 3110 (3111br | 2755ar) 3111 (3112br | 3671ar) 3112 (3113br | 2168ar) 3113 (3114br | 3936ar) 3114 (3115br | 3126ar) 3115 (3116br | 2432ar) 3116 (3117br | 2764ar) 3117 (3118br | 3478ar) 3118 (3119br | 2219ar) 3119 (3120br | 3497ar) 3120 (3121br | 2539ar) 3121 (3122br | 3399ar) 3122 (3123br | 2219ar) 3123 (3124br | 2008ar) 3124 (3125br | 2473ar) 3125 (3126br | 3532ar) 3126 (3127br | 1994ar) 3127 (3128br | 3621ar) 3128 (3129br | 3574ar) 3129 (3130br | 3937ar) 3130 (3131br | 2531ar) 3131 (3132br | 2798ar) 3132 (3133br | 1999ar) 3133 (3134br | 2455ar) 3134 (3135br | 3476ar) 3135 (3136br | 3464ar) 3136 (3137br | 2546ar) 3137 (3138br | 3591ar) 3138 (3139br | 3722ar) 3139 (3140br | 3590ar) 3140 (3141br | 2740ar) 3141 (3142br | 2265ar) 3142 (3143br | 2735ar) 3143 (3144br | 2656ar) 3144 (3145br | 3428ar) 3145 (3146br | 2970ar) 3146 (3147br | 2764ar) 3147 (3148br | 3833ar) 3148 (3149br | 2888ar) 3149 (3150br | 3863ar) 3150 (3151br | 3940ar) 3151 (3152br | 2265ar) 3152 (3153br | 2787ar) 3153 (3154br | 3652ar) 3154 (3155br | 2787ar) 3155 (3156br | 3130ar) 3156 (3157br | 2124ar) 3157 (3158br | 3938ar) 3158 (3159br | 3833ar) 3159 (3160br | 3782ar) 3160 (3161br | 2671ar) 3161 (3162br | 2477ar) 3162 (3163br | 2045ar) 3163 (3164br | 2432ar) 3164 (3165br | 2923ar) 3165 (3166br | 3940ar) 3166 (3167br | 2147ar) 3167 (3168br | 2391ar) 3168 (3169br | 2219ar) 3169 (3170br | 2009ar) 3170 (3171br | 2186ar) 3171 (3172br | 3787ar) 3172 (3173br | 3272ar) 3173 (3174br | 3591ar) 3174 (3175br | 2740ar) 3175 (3176br | 2265ar) 3176 (3177br | 2688ar) 3177 (3178br | 3936ar) 3178 (3179br | 2635ar) 3179 (3180br | 3033ar) 3180 (3181br | 2186ar) 3181 (3182br | 3956ar) 3182 (3183br | 2208ar) 3183 (3184br | 3601ar) 3184 (3185br | 3602ar) 3185 (3186br | 3652ar) 3186 (3187br | 2508ar) 3187 (3188br | 2457ar) 3188 (3189br | 2892ar) 3189 (3190br | 2340ar) 3190 (3191br | 2795ar) 3191 (3192br | 3505ar) 3192 (3193br | 2740ar) 3193 (3194br | 2265ar) 3194 (3195br | 2178ar) 3195 (3196br | 3953ar) 3196 (3197br | 2208ar) 3197 (3198br | 3601ar) 3198 (3199br | 3605ar) 3199 (3200br | 3648ar) 3200 (3201br | 2544ar) 3201 (3202br | 3409ar) 3202 (3203br | 3901ar) 3203 (3204br | 1989ar) 3204 (3205br | 2892ar) 3205 (3206br | 3109ar) 3206 (3207br | 3716ar) 3207 (3208br | 3208ar) 3208 (3209br | 3716ar) 3209 (3210br | 2013ar) 3210 (3211br | 2688ar) 3211 (3212br | 3936ar) 3212 (3213br | 2156ar) 3213 (3214br | 3409ar) 3214 (3215br | 3182ar) 3215 (3216br | 2126ar) 3216 (3217br | 2126ar) 3217 (3218br | 2103ar) 3218 (3219br | 2683ar) 3219 (3220br | 3670ar) 3220 (3221br | 2162ar) 3221 (3222br | 2162ar) 3222 (3223br | 319ar) 3223 (3224br | 3244ar) 3224 (3225br | 3244ar) 3225 (3226br | 3797ar) 3226 (3227br | 3799ar) 3227 (3228br | 3487ar) 3228 (3229br | 2539ar) 3229 (3230br | 2375ar) 3230 (3231br | 2219ar) 3231 (3232br | 2008ar) 3232 (3233br | 2743ar) 3233 (3234br | 2650ar) 3234 (3235br | 3186ar) 3235 (3236br | 3781ar) 3236 (3237br | 3851ar) 3237 (3238br | 2103ar) 3238 (3239br | 2173ar) 3239 (3240br | 2055ar) 3240 (3241br | 2030ar) 3241 (3242br | 2107ar) 3242 (3243br | 2507ar) 3243 (3244br | 3032ar) 3244 (3245br | 2659ar) 3245 (3246br | 3396ar) 3246 (3247br | 3722ar) 3247 (3248br | 3781ar) 3248 (3249br | 3055ar) 3249 (3250br | 2513ar) 3250 (3251br | 3555ar) 3251 (3252br | 2270ar) 3252 (3253br | 3136ar) 3253 (3254br | 3797ar) 3254 (3255br | 3797ar) 3255 (3256br | 3799ar) 3256 (3257br | 3799ar) 3257 (3258br | 3799ar) 3258 (3259br | 2159ar) 3259 (3260br | 2496ar) 3260 (3261br | 2210ar) 3261 (3262br | 3782ar) 3262 (3263br | 3126ar) 3263 (3264br | 3032ar) 3264 (3265br | 2552ar) 3265 (3266br | 3389ar) 3266 (3267br | 2126ar) 3267 (3268br | 2072ar) 3268 (3269br | 3055ar) 3269 (3270br | 2513ar) 3270 (3271br | 2124ar) 3271 (3272br | 2106ar) 3272 (3273br | 2542ar) 3273 (3274br | 2323ar) 3274 (3275br | 2168ar) 3275 (3276br | 3311ar) 3276 (3277br | 2539ar) 3277 (3278br | 3028ar) 3278 (3279br | 3514ar) 3279 (3280br | 3514ar) 3280 (3281br | 2720ar) 3281 (3282br | 2433ar) 3282 (3283br | 2975ar) 3283 (3284br | 2564ar) 3284 (3285br | 2034ar) 3285 (3286br | 3393ar) 3286 (3287br | 2762ar) 3287 (3288br | 3781ar) 3288 (3289br | 3842ar) 3289 (3290br | 2433ar) 3290 (3291br | 2891ar) 3291 (3292br | 3728ar) 3292 (3293br | 2720ar) 3293 (3294br | 3524ar) 3294 (3295br | 2923ar) 3295 (3296br | 3191ar) 3296 (3297br | 2540ar) 3297 (3298br | 3606ar) 3298 (3299br | 2158ar) 3299 (3300br | 3514ar) 3300 (3301br | 2722ar) 3301 (3302br | 2437ar) 3302 (3303br | 2990ar) 3303 (3304br | 3120ar) 3304 (3305br | 2740ar) 3305 (3306br | 3966ar) 3306 (3307br | 2168ar) 3307 (3308br | 2432ar) 3308 (3309br | 2932ar) 3309 (3310br | 3936ar) 3310 (3311br | 2156ar) 3311 (3312br | 2076ar) 3312 (3313br | 2168ar) 3313 (3314br | 2123ar) 3314 (3315br | 2123ar) 3315 (3316br | 2123ar) 3316 (3317br | 2123ar) 3317 (3318br | 2123ar) 3318 (3319br | 2123ar) 3319 (3320br | 2123ar) 3320 (3321br | 2123ar) 3321 (3322br | 2123ar) 3322 (3323br | 2123ar) 3323 (3324br | 2123ar) 3324 (3325br | 2123ar) 3325 (3326br | 2123ar) 3326 (3327br | 2123ar) 3327 (3328br | 2123ar) 3328 (3329br | 2123ar) 3329 (3330br | 2123ar) 3330 (3331br | 2123ar) 3331 (3332br | 2123ar) 3332 (3333br | 2123ar) 3333 (3334br | 2123ar) 3334 (3335br | 2123ar) 3335 (3336br | 2123ar) 3336 (3337br | 2123ar) 3337 (3338br | 2123ar) 3338 (3339br | 2123ar) 3339 (3340br | 2123ar) 3340 (3341br | 2123ar) 3341 (3342br | 2123ar) 3342 (3343br | 2123ar) 3343 (3344br | 2123ar) 3344 (3345br | 2123ar) 3345 (3346br | 2123ar) 3346 (3347br | 2123ar) 3347 (3348br | 2123ar) 3348 (3349br | 2123ar) 3349 (3350br | 2123ar) 3350 (3351br | 2123ar) 3351 (3352br | 2123ar) 3352 (3353br | 2123ar) 3353 (3354br | 2123ar) 3354 (3355br | 2123ar) 3355 (3356br | 2123ar) 3356 (3357br | 2123ar) 3357 (3358br | 2123ar) 3358 (3359br | 2123ar) 3359 (3360br | 2123ar) 3360 (3361br | 2123ar) 3361 (3362br | 2123ar) 3362 (3363br | 2123ar) 3363 (3364br | 2123ar) 3364 (3365br | 2123ar) 3365 (3366br | 2123ar) 3366 (3367br | 2123ar) 3367 (3368br | 2123ar) 3368 (3369br | 2123ar) 3369 (3370br | 2123ar) 3370 (3371br | 2123ar) 3371 (3372br | 2123ar) 3372 (3373br | 2123ar) 3373 (3374br | 2123ar) 3374 (3375br | 2123ar) 3375 (3376br | 2123ar) 3376 (3377br | 2123ar) 3377 (3378br | 2123ar) 3378 (3379br | 2123ar) 3379 (3380br | 2123ar) 3380 (3381br | 2123ar) 3381 (3382br | 2123ar) 3382 (3383br | 2123ar) 3383 (3384br | 2123ar) 3384 (3385br | 2123ar) 3385 (3386br | 2123ar) 3386 (3387br | 2123ar) 3387 (3388br | 2123ar) 3388 (3389br | 2123ar) 3389 (3390br | 2123ar) 3390 (3391br | 2123ar) 3391 (3392br | 2123ar) 3392 (3393br | 2123ar) 3393 (3394br | 2123ar) 3394 (3395br | 2123ar) 3395 (3396br | 2123ar) 3396 (3397br | 2123ar) 3397 (3398br | 2123ar) 3398 (3399br | 2123ar) 3399 (3400br | 2123ar) 3400 (3401br | 2123ar) 3401 (3402br | 2123ar) 3402 (3403br | 2123ar) 3403 (3404br | 2123ar) 3404 (3405br | 2123ar) 3405 (3406br | 2123ar) 3406 (3407br | 2123ar) 3407 (3408br | 2123ar) 3408 (3409br | 2123ar) 3409 (3410br | 2123ar) 3410 (3411br | 2123ar) 3411 (3412br | 2123ar) 3412 (3413br | 2123ar) 3413 (3414br | 2123ar) 3414 (3415br | 2123ar) 3415 (3416br | 2123ar) 3416 (3417br | 2123ar) 3417 (3418br | 2123ar) 3418 (3419br | 2123ar) 3419 (3420br | 2123ar) 3420 (3421br | 2123ar) 3421 (3422br | 2123ar) 3422 (3423br | 2123ar) 3423 (3424br | 2123ar) 3424 (3425br | 2123ar) 3425 (3426br | 2123ar) 3426 (3427br | 2123ar) 3427 (3428br | 2123ar) 3428 (3429br | 2123ar) 3429 (3430br | 2123ar) 3430 (3431br | 2123ar) 3431 (3432br | 2123ar) 3432 (3433br | 2123ar) 3433 (3434br | 2123ar) 3434 (3435br | 2123ar) 3435 (3436br | 2123ar) 3436 (3437br | 2123ar) 3437 (3438br | 2123ar) 3438 (3439br | 2123ar) 3439 (3440br | 2123ar) 3440 (3441br | 2123ar) 3441 (3442br | 2123ar) 3442 (3443br | 2123ar) 3443 (3444br | 2123ar) 3444 (3445br | 2123ar) 3445 (3446br | 2123ar) 3446 (3447br | 2123ar) 3447 (3448br | 2123ar) 3448 (3449br | 2123ar) 3449 (3450br | 2123ar) 3450 (3451br | 2123ar) 3451 (3452br | 2123ar) 3452 (3453br | 2123ar) 3453 (3454br | 2123ar) 3454 (3455br | 2123ar) 3455 (3456br | 2123ar) 3456 (3457br | 2123ar) 3457 (3458br | 2123ar) 3458 (3459br | 2123ar) 3459 (3460br | 2123ar) 3460 (3461br | 2123ar) 3461 (3462br | 2123ar) 3462 (3463br | 2123ar) 3463 (3464br | 2123ar) 3464 (3465br | 2123ar) 3465 (3466br | 2123ar) 3466 (3467br | 2123ar) 3467 (3468br | 2123ar) 3468 (3469br | 2123ar) 3469 (3470br | 2123ar) 3470 (3471br | 2123ar) 3471 (3472br | 2123ar) 3472 (3473br | 2123ar) 3473 (3474br | 2123ar) 3474 (3475br | 2123ar) 3475 (3476br | 2123ar) 3476 (3477br | 2123ar) 3477 (3478br | 2123ar) 3478 (3479br | 2123ar) 3479 (3480br | 2123ar) 3480 (3481br | 2123ar) 3481 (3482br | 2123ar) 3482 (3483br | 2123ar) 3483 (3484br | 2123ar) 3484 (3485br | 2123ar) 3485 (3486br | 2123ar) 3486 (3487br | 2123ar) 3487 (3488br | 2123ar) 3488 (3489br | 2123ar) 3489 (3490br | 2123ar) 3490 (3491br | 2123ar) 3491 (3492br | 2123ar) 3492 (3493br | 2123ar) 3493 (3494br | 2123ar) 3494 (3495br | 2123ar) 3495 (3496br | 2123ar) 3496 (3497br | 2123ar) 3497 (3498br | 2123ar) 3498 (3499br | 2123ar) 3499 (3500br | 2123ar) 3500 (3501br | 2123ar) 3501 (3502br | 2123ar) 3502 (3503br | 2123ar) 3503 (3504br | 2123ar) 3504 (3505br | 2123ar) 3505 (3506br | 2123ar) 3506 (3507br | 2123ar) 3507 (3508br | 2123ar) 3508 (3509br | 2123ar) 3509 (3510br | 2123ar) 3510 (3511br | 2123ar) 3511 (3512br | 2123ar) 3512 (3513br | 2123ar) 3513 (3514br | 2123ar) 3514 (3515br | 2123ar) 3515 (3516br | 2123ar) 3516 (3517br | 2123ar) 3517 (3518br | 2123ar) 3518 (3519br | 2123ar) 3519 (3520br | 2123ar) 3520 (3521br | 2123ar) 3521 (3522br | 2123ar) 3522 (3523br | 2123ar) 3523 (3524br | 2123ar) 3524 (3525br | 2123ar) 3525 (3526br | 2123ar) 3526 (3527br | 2123ar) 3527 (3528br | 2123ar) 3528 (3529br | 2123ar) 3529 (3530br | 2123ar) 3530 (3531br | 2123ar) 3531 (3532br | 2123ar) 3532 (3533br | 2123ar) 3533 (3534br | 2123ar) 3534 (3535br | 2123ar) 3535 (3536br | 2123ar) 3536 (3537br | 2123ar) 3537 (3538br | 2123ar) 3538 (3539br | 2123ar) 3539 (3540br | 2123ar) 3540 (3541br | 2123ar) 3541 (3542br | 2123ar) 3542 (3543br | 2123ar) 3543 (3544br | 2123ar) 3544 (3545br | 2123ar) 3545 (3546br | 2123ar) 3546 (3547br | 2123ar) 3547 (3548br | 2123ar) 3548 (3549br | 2123ar) 3549 (3550br | 2123ar) 3550 (3551br | 2123ar) 3551 (3552br | 2123ar) 3552 (3553br | 2123ar) 3553 (3554br | 2123ar) 3554 (3555br | 2123ar) 3555 (3556br | 2123ar) 3556 (3557br | 2123ar) 3557 (3558br | 2123ar) 3558 (3559br | 2123ar) 3559 (3560br | 2123ar) 3560 (3561br | 2123ar) 3561 (3562br | 2123ar) 3562 (3563br | 2123ar) 3563 (3564br | 2123ar) 3564 (3565br | 2123ar) 3565 (3566br | 2123ar) 3566 (3567br | 2123ar) 3567 (3568br | 2123ar) 3568 (3569br | 2123ar) 3569 (3570br | 2123ar) 3570 (3571br | 2123ar) 3571 (3572br | 2123ar) 3572 (3573br | 2123ar) 3573 (3574br | 2123ar) 3574 (3575br | 2123ar) 3575 (3576br | 2123ar) 3576 (3577br | 2123ar) 3577 (3578br | 2123ar) 3578 (3579br | 2123ar) 3579 (3580br | 2123ar) 3580 (3581br | 2123ar) 3581 (3582br | 2123ar) 3582 (3583br | 2123ar) 3583 (3584br | 2123ar) 3584 (3585br | 2123ar) 3585 (3586br | 2123ar) 3586 (3587br | 2123ar) 3587 (3588br | 2123ar) 3588 (3589br | 2123ar) 3589 (3590br | 2123ar) 3590 (3591br | 2123ar) 3591 (3592br | 2123ar) 3592 (3593br | 2123ar) 3593 (3594br | 2123ar) 3594 (3595br | 2123ar) 3595 (3596br | 2123ar) 3596 (3597br | 2123ar) 3597 (3598br | 2123ar) 3598 (3599br | 2123ar) 3599 (3600br | 2123ar) 3600 (3601br | 2123ar) 3601 (3602br | 2123ar) 3602 (3603br | 2123ar) 3603 (3604br | 2123ar) 3604 (3605br | 2123ar) 3605 (3606br | 2123ar) 3606 (3607br | 2123ar) 3607 (3608br | 2123ar) 3608 (3609br | 2123ar) 3609 (3610br | 2123ar) 3610 (3611br | 2123ar) 3611 (3612br | 2123ar) 3612 (3613br | 2123ar) 3613 (3614br | 2123ar) 3614 (3615br | 2123ar) 3615 (3616br | 2123ar) 3616 (3617br | 2123ar) 3617 (3618br | 2123ar) 3618 (3619br | 2123ar) 3619 (3620br | 2123ar) 3620 (3621br | 2123ar) 3621 (3622br | 2123ar) 3622 (3623br | 2123ar) 3623 (3624br | 2123ar) 3624 (3625br | 2123ar) 3625 (3626br | 2123ar) 3626 (3627br | 2123ar) 3627 (3628br | 2123ar) 3628 (3629br | 2123ar) 3629 (3630br | 2123ar) 3630 (3631br | 2123ar) 3631 (3632br | 2123ar) 3632 (3633br | 2123ar) 3633 (3634br | 2123ar) 3634 (3635br | 2123ar) 3635 (3636br | 2123ar) 3636 (3637br | 2123ar) 3637 (3638br | 2123ar) 3638 (3639br | 2123ar) 3639 (3640br | 2123ar) 3640 (3641br | 2123ar) 3641 (3642br | 2123ar) 3642 (3643br | 2123ar) 3643 (3644br | 2123ar) 3644 (3645br | 2123ar) 3645 (3646br | 2123ar) 3646 (3647br | 2123ar) 3647 (3648br | 2123ar) 3648 (3649br | 2123ar) 3649 (3650br | 2123ar) 3650 (3651br | 2123ar) 3651 (3652br | 2123ar) 3652 (3653br | 2123ar) 3653 (3654br | 2123ar) 3654 (3655br | 2123ar) 3655 (3656br | 2123ar) 3656 (3657br | 2123ar) 3657 (3658br | 2123ar) 3658 (3659br | 2123ar) 3659 (3660br | 2123ar) 3660 (3661br | 2123ar) 3661 (3662br | 2123ar) 3662 (3663br | 2123ar) 3663 (3664br | 2123ar) 3664 (3665br | 2123ar) 3665 (3666br | 2123ar) 3666 (3667br | 2123ar) 3667 (3668br | 2123ar) 3668 (3669br | 2123ar) 3669 (3670br | 2123ar) 3670 (36

20

7610(7609aR|7611bL) 7611(7612aR|7612bR) 7612(7616aL|7616bL) 7613(ERROR-|7614bL) 7614(7615aR|7615bR) 7615(7616aL|7616bL) 7616(7617aR|7618bR) 7617(7619aR|7616bR) 7618(7616aR|7616bR) 7619(7620bR|ERROR-) 7620(7621aR|7621bR) 7621(7622aR|7623bR) 7622(7621aR|7621bR) 7623(7624aR|7621bR) 7624(7625aR|7626bR) 7625(7626aL|7626bL) 7626(7627aL|7624bR) 7627(7628aR|7628aR) 7628(7629aR|7629bR) 7629(7630aR|7635bR) 7630(7631aL|7633bL) 7631(7632aR|7632bR) 7632(7638aL|7638bL) 7633(7634aR|7634bR) 7634(7149aL|7149bL) 7635(ERROR-|7636bL) 7636(7637aR|7637bR) 7637(7149aL|7149bL) 7638(7639aR|7644aR) 7639(7640aL|7642bL) 7640(7641aL|7641bL) 7641(7638aL|7638bL) 7642(7643aL|7643bL) 7643(7638aL|7638bL) 7644(7645aL|7647bL) 7645(7646aL|7646bL) 7646(7649aL|7649bL) 7647(7648aL|7648bL) 7648(7638aL|7638bL) 7649(7650aR|7655bR) 7650(7651aL|7653bL) 7651(7652aL|7652bL) 7652(7649aL|7649bL) 7653(7654aL|7654bL) 7654(7649aL|7649bL) 7655(7656aL|7658bL) 7656(7657aR|7657aR) 7657(7660aL|7660bL) 7658(7659aL|7659bL) 7659(7649aL|7649bL) 7660(7661aR|7666bR) 7661(7662aL|7664bL) 7662(7663aL|7663bL) 7663(7660aL|7660bL) 7664(7665aR|7665bR) 7665(7669aL|7669bL) 7666(ERROR-|7667bL) 7667(7668aR|7668bR) 7668(7669aL|7669bL) 7669(7670aR|7675bR) 7670(7671aL|7673bL) 7671(7672bR|7672bR) 7672(7680aL|7680bL) 7673(7674aL|7674bL) 7674(7669aL|7669bL) 7675(7676aL|7678bL) 7676(7677aL|7677bL) 7677(7669aL|7669bL) 7678(7679aL|7679bL) 7679(7669aL|7669bL) 7680(7681aR|7686bR) 7681(7682aL|7684aL) 7682(7683aL|7683bL) 7683(7721aL|7721bL) 7684(7685bL|7685bL) 7685(7689aL|7689bL) 7686(7697aR|7687aL) 7687(7688bL|7688bL) 7688(7691aL|7691bL) 7689(ERROR-|7690aR) 7690(7693bR|ERROR-) 7691(ERROR-|7692bR) 7692(7695bR|ERROR-) 7693(ERROR-|7694aR) 7694(7702aR|7702bR) 7695(ERROR-|7696bR) 7696(7707aR|7707bR) 7697(7698aR|7699aR) 7698(7697aR|7697bR) 7699(7700aL|7697bR) 7700(7701aR|7701aR) 7701(7746aR|7746bR) 7702(7703aR|7704bR) 7703(7702aR|7702bR) 7704(7705bL|7702bR) 7705(7706aR|7706aR) 7706(7746aR|7746bR) 7707(7708aR|7709bR) 7708(7707aR|7707bR) 7709(7710bL|7707bR) 7710(7711bR|7711bR) 7711(7746aR|7746bR) 7712(7703aR|7718bR) 7713(7714aL|7716bL) 7714(7715aL|7715bL) 7715(7712aL|7712bL) 7716(7717aL|7717bL) 7717(7712aL|7712bL) 7718(7680aR|7719bL) 7719(7720aL|7720bL) 7720(7712aL|7712bL) 7721(ERROR-|7722bR) 7722(7723bL|ERROR-) 7723(7724aL|7724bL) 7724(7725aL|7725bL) 7725(7726aR|7731bR) 7726(7727aL|7729aL) 7727(7728aR|7728bR) 7728(7734aL|7734bL) 7729(7730aL|7730bL) 7730(7725aL|7725bL) 7731(ERROR-|7732aL) 7732(7733aL|7733aL) 7733(7725aL|7725bL) 7734(7735aR|7736bR) 7735(7734aR|7734bR) 7736(7734aR|7990bR) 7737(7738aR|7741bR) 7738(7739bL|7737bR) 7739(7740aR|7740aR) 7740(7764aR|7764bR) 7741(7742bL|7744bL) 7742(7743aR|7743aR) 7743(7746aR|7746bR) 7744(7745aR|7745aR) 7745(7755aR|7755bR) 7746(7747aR|7752bR) 7747(7748aL|7750aL) 7748(7749bR|7749bR) 7749(7764aR|7764bR) 7750(7751bR|7751bR) 7751(7737aR|7737bR) 7752(7737aR|7753aL) 7753(7754bR|7754bR) 7754(7755aR|7755bR) 7755(7756aR|7761bR) 7756(7757bL|7759bL) 7757(7758bR|7758bR) 7758(7764aR|7764aR) 7759(7760bR|7760bR) 7760(7737aR|7737bR) 7761(7762bL|7755bR) 7762(7763bR|7763bR) 7763(7746aR|7746bR) 7764(7765aR|7768bR) 7765(7764aR|7764aR) 7766(7767aR|7767aR) 7767(7737aR|7737bR) 7768(7769aR|7769bR) 7769(7773aR|7737bR) 7770(7746aR|7746bR) 7771(7772aR|7772aR) 7772(7756aR|7755bR) 7773(7774aR|7779bR) 7774(7775aL|7777aL) 7775(7776bR|7776bR) 7776(7764aR|7764bR) 7777(7778bR|7778bR) 7778(7737aR|7737bR) 7779(7782aR|7780aL) 7780(7781bR|7781bR) 7781(7756aR|7755bR) 7782(7783bR|ERROR-) 7783(7784aL|7784aL) 7784(7785aR|7788bR) 7785(7786aL|ERROR-) 7786(7787aR|7787bR) 7787(7791aL|7791bL) 7788(7789aL|ERROR-) 7789(7790aL|7790bL) 7790(7784aL|7784bL) 7791(7792aR|7797bR) 7792(7793aL|7795bL) 7793(7794aL|7794bL) 7794(7795aL|7795bL) 7795(7796aR|7796bR) 7796(7797aL|7797bL) 7797(7798aL|7800bL) 7798(7799aR|7800bL) 7799(7712aL|7712bL) 7800(7801aL|7801bL) 7801(7791aL|7791bL) 7802(7803aR|7808bR) 7803(7804aL|7806bL) 7804(7805aL|7805bL) 7805(7802aL|7802bL) 7806(7807aL|7807bL) 7807(7802aL|7802bL) 7808(7809aL|7811bL) 7809(7810aL|7810aL) 7810(7813aL|7813bL) 7811(7812aL|7812bL) 7812(7802aL|7802bL) 7813(7814aR|7819bR) 7814(7815aL|7817bL) 7815(7816aL|7816bL) 7816(7813aL|7813bL) 7817(7818aR|7818aR) 7818(7822aL|7822bL) 7819(ERROR-|7820bL) 7820(7821aR|7821bR) 7821(7822aL|7822bL) 7822(7823aR|7828bR) 7823(7824aL|7826bL) 7824(7825bL|7825bL) 7825(7833aL|7833bL) 7826(7827aL|7827bL) 7827(7822aL|7822bL) 7828(7829aL|7831bL) 7829(7830aL|7830bL) 7830(7822aL|7822bL) 7831(7832aL|7832bL) 7832(7822aL|7822bL) 7833(7834aR|7839bR) 7834(7835aL|7837bL) 7835(7836aL|7836bL) 7836(7833aL|7833bL) 7837(7838aL|7838bL) 7838(7833aL|7833bL) 7839(7840aL|7842bL) 7840(7841aL|7841aL) 7841(7844aL|7844bL) 7842(7843aL|7843bL) 7843(7833aL|7833bL) 7844(7845bL|ERROR-) 7845(7846aL|7846bL) 7846(7847aR|7852bR) 7847(7848aL|7850aL) 7848(7849aR|7849bR) 7849(7855aL|7855bL) 7850(7851aL|7851aL) 7851(7846aL|7846bL) 7852(ERROR-|7853aL) 7853(7854aL|7854aL) 7854(7846aL|7846bL) 7855(7856aR|7857bR) 7856(7855aR|7855bR) 7857(7858aR|7858bR) 7858(7859aR|7860bR) 7859(7858aR|7858bR) 7860(7861aL|7858bR) 7861(7862aR|7862aR) 7862(7863aR|7863bR) 7863(7864aR|7869bR) 7864(7865aL|7867aL) 7865(7866aR|7866bR) 7866(7946aL|7946bL) 7867(7869bL|7869bL) 7868(7872aL|7872bL) 7869(ERROR-|7870aL) 7870(7871bL|7871bL) 7871(7885aL|7885bL) 7872(7873aR|7878bR) 7873(7874aL|7876bL) 7874(7875aL|7875bL) 7875(7872aL|7872bL) 7876(7877aL|7877bL) 7877(7872aL|7872bL) 7878(7879aL|7881bL) 7879(7880aL|7880bL) 7880(7894aL|7894bL) 7881(7882aL|7882bL) 7882(7872aL|7872bL) 7883(7884aR|7889bR) 7884(7885aL|7887bL) 7885(7886aL|7886bL) 7886(7883aL|7883bL) 7887(7888aL|7888bL) 7888(7889aL|7889bL) 7889(7894aL|7894bL) 7890(ERROR-|7901bL) 7891(7921aL|7921bL) 7892(7893aL|7893bL) 7893(7883aL|7883bL) 7894(7895aR|7900bR) 7895(7896bL|7896bL) 7896(7897aR|7897aR) 7897(7930aL|7930bL) 7898(7899aL|7899bL) 7899(7894aL|7894bL) 7900(ERROR-|7901bL) 7901(7902aL|7902aL) 7902(7903aL|7903bL) 7903(7904aR|7909bR) 7904(7905bL|7907bL) 7905(7906bR|7906bR) 7906(7930aL|7930bL) 7907(7908bL|7908bL) 7908(7909aL|7909bL) 7909(ERROR-|7910bL) 7910(7911aL|7911bL) 7911(7903aL|7903bL) 7912(7913aR|7918bR) 7913(7914bL|7916bL) 7914(7915aR|7915aR) 7915(7933aL|7933bL) 7916(7917aL|7917bL) 7917(7912aL|7912bL) 7918(ERROR-|7919bL) 7919(7920aL|7920aL) 7920(7921aL|7921bL) 7921(7922aR|7927bR) 7922(7923bL|7925bL) 7923(7924bR|7924bR) 7924(7933aL|7933bL) 7925(7926bL|7926bL) 7926(7912aL|7912bL) 7927(ERROR-|7928bL) 7928(7929aL|7929bL) 7929(7921aL|7921bL) 7930(7931aR|7932bR) 7931(7930aR|7930bR) 7932(7936aR|7936bR) 7933(7934aR|7935bR) 7934(7933aR|7933bR) 7935(7941aR|7933bR) 7936(7937aR|7938bR) 7937(7936aR|7936bR) 7938(7939aL|7940aR) 7939(7940aR|7940aR) 7940(7863aR|7863bR) 7941(7942aR|7943bR) 7942(7941aR|7941bR) 7943(7944bL|7941bR) 7944(7945bR|7945bR) 7945(7863aR|7863bR) 7946(7947aR|7952bR) 7947(7948aL|7950aL) 7948(7949aL|7949bL) 7949(7955aL|7955bL) 7950(7951aL|7951aL) 7951(7946aL|7946bL) 7952(ERROR-|7953aL) 7953(7954aL|7954aL) 7954(7946aL|7946bL) 7955(7956aR|7961bR) 7956(7957aL|7959aL) 7957(7958aL|7958bL) 7958(7964aL|7964bL) 7959(7960aL|7960aL) 7960(7946aL|7946bL) 7961(ERROR-|7962aL) 7962(7963aL|7963aL) 7963(7946aL|7946bL) 7964(7965aR|7966bR) 7965(BAL-|7967bR) 7966(ERROR-|7967bR) 7967(7968aR|7973bR) 7968(7969aL|7971bL) 7969(7970aL|7970bL) 7970(7967aL|7967bL) 7971(7972aL|7972bL) 7972(7967aL|7967bL) 7973(7974aL|7976bL) 7974(7975aL|7975bL) 7975(7976aR|7976bR) 7976(7977aL|7977bL) 7977(7967aL|7967bL) 7978(7979aR|7984bR) 7979(7980bL|7982bL) 7980(7981bR|7981bR) 7981(7987aL|7987bL) 7982(7983bL|7983bL) 7983(7978aL|7978bL) 7984(ERROR-|7985bL) 7985(7986aR|7986aR) 7986(7987aL|7987bL) 7987(7988aR|7989bR) 7988(7990aR|7987bR) 7989(7987aR|7987bR) 7990(7991aR|7992bR) 7991(7990aR|ERROR-) 7992(ERROR-|7993bL) 7993(7994aR|7994aR) 7994(7995aL|7995bL) 7995(7996aR|7997bR) 7996(7998aR|7998aR) 7997(7999aR|7999bR) 7998(7999aR|8000bR) 7999(8001aR|7995bR) 8000(7995aR|7995bR) 8001(8002aR|8002bR) 8002(8003aL|7995bR) 8003(8004aR|8004bR) 8004(8006aL|8006bL) 8005(7995aR|7995bR) 8006(8007aR|8008bR) 8007(8006aR|ERROR-) 8008(ERROR-|8009bL) 8009(8010aL|8010bL) 8010(8011aL|8011bL) 8011(8012bL|ERROR-) 8012(4484aL|4484bL)