Scientific Research Publishing

Advances in Pure Mathematics, 2025, 15(6), 390-411
https://www.scirp.org/journal/apm
ISSN Online: 2160-0384
ISSN Print: 2160-0368

Further Remarks on the Infinity Tree

Philip C. Jackson

TalaMind LLC, Troy, MI, USA
Email: dr.phil.jackson@talamind.com

How to cite this paper: Jackson, P.C. (2025) Further Remarks on the Infinity Tree. *Advances in Pure Mathematics*, **15**, 390-411.
https://doi.org/10.4236/apm.2025.156019

Received: April 30, 2025
Accepted: June 15, 2025
Published: June 18, 2025

Abstract

A 2023 paper discussed how the real numbers between 0 and 1 could be represented by an infinite tree structure called the "infinity tree", which contains only a countably infinite number of nodes and arcs. A 2025 paper discussed how a finite-state Turing machine could, in a countably infinite number of state transitions, write all the infinite paths in the infinity tree to a countably infinite tape, supporting an argument that the real numbers in the interval [0, 1] are countably infinite, in a non-Cantorian theory of infinity. However, it is not claimed that there is an error in Cantor's arguments that [0, 1] is uncountably infinite. Rather, the situation is considered as a paradox, resulting from different choices about how to represent and count the continuum of real numbers. The present, short paper addresses some further questions that mathematicians have asked, and might ask, about this approach.

Keywords

Infinity, Countable, Uncountable, Diagonalization, Real Numbers, Infinity Tree, Continuum Hypothesis, Turing Machine, Infinite Time Turing Machine, Non-Cantorian

1. Introduction

A 2023 paper [1] discussed how the real numbers between 0 and 1 could be represented by an infinite tree structure, called the "*infinity tree*", which contains only a countably infinite number of nodes and arcs. A 2025 paper [2] discussed how a finite-state Turing machine could, in a countably infinite number of state transitions, write all the infinite paths in the infinity tree to a countably infinite tape, supporting an argument that the real numbers in the interval [0, 1] are countably infinite in a non-Cantorian theory of infinity.

The present paper addresses some further questions that mathematicians have asked, and might ask: How do we know that the infinity tree has a countably infinite number of levels? How do we know that the Turing machine only requires

finite time to simulate processing each level of the infinity tree? How can a Turing machine avoid generating infinite 0's at the start of its output, if it simulates processing the entire infinity tree?

Like the preceding papers [1] and [2], this paper does not claim there is an error in Cantor's arguments that [0, 1] is uncountably infinite. Rather, this paper considers the situation as a paradox, resulting from different choices about how to represent and count the continuum of real numbers.

2. The Infinity Tree Has a Countably Infinite Number of Levels

The first paper on the infinity tree (1) defined the tree to have "as many levels as there are natural numbers." This also follows from the definition of the infinity tree given in the second paper (2), which says the infinity tree is a "simple infinite tree, with a branching factor of 10 arcs from each node." A simple infinite tree with a constant branching factor has a countably infinite number of levels.

One mathematician reading the second paper mistakenly thought it was describing the infinity tree as having uncountably many levels. However, a tree with uncountably many levels would not make any sense, as it would imply that a single decimal expression would have uncountably many digits, which would be a very non-standard idea. Cantor only considered decimal expressions which each had countably many digits. The infinity tree papers (1) and (2) follow suit: Each infinite path descending through the infinity tree represents a single decimal expression that has countably many digits.

The infinity tree gives a representation of the real numbers in [0, 1) which is not just a list or set of numbers that can be diagonalized to construct a missing number. It is a tree structure that is defined so that each infinite path in the tree corresponds to a unique infinite decimal expression in [0, 1), and so that each unique infinite decimal expression in [0, 1) corresponds to a unique infinite path in the tree.

A second mathematician reading (2) mistakenly thought it was making the following claims:

> The set of real numbers (say expressed as infinite binary sequences) is the same as the set of nodes in a binary tree.
>
> The set of real numbers is the same as the set of finite binary sequences.

My papers do not make these claims. The infinity tree is an infinite "decimal tree", not a binary tree: Each node in the infinity tree has 10 child nodes. The infinity tree has a countably infinite number of levels. Each infinite path descending through the tree corresponds to a unique infinite decimal expression in [0, 1), and each unique infinite decimal expression in [0, 1) corresponds to a unique infinite path descending through the tree.

The second mathematician wrote:

"I suspect you will find that terminal sequences of the form… 0999… are equivalent to …1000…, so there are countably many double representations of a real

number."

My papers (1) and (2) are not trying to reduce the size of the set of numbers in [0, 1) by considering some numbers in the set to be approximately equal. My purpose in defining the infinity tree is to give a representation of all the real numbers in [0, 1) which is not just a list or set of numbers that Cantor could diagonalize to construct a missing number. It's a tree structure that is defined so that it cannot be diagonalized to construct any number in [0, 1) which it does not contain. Approximate equality is not a factor in this discussion.

3. The Turing Machine Uses Only Finite Time to Process Each Tree Level

This is explained in section 4 of the second paper. (2) Section 4.3 discusses "infinite time Turing machines", for which time is implicitly represented by the computation steps of a Turing machine. The second paper (2) contends that a certain, specific Turing machine would be able to generate and write all the real numbers in [0, 1] within countably infinite computation steps, using a constant, finite number of countably infinite tapes.

For a Turing machine, countably infinite time corresponds to a countably infinite sequence of computation steps. A unit of time (a discrete time step) corresponds to a single computation step, within which: a) a Turing machine reads a symbol on a tape; b) based on the machine's current state, the machine may write a symbol on a tape, and then optionally move one space left or right on a tape; and c) optionally the machine may change its state.

Precisely what happens within each computation step is specified by the finite state transition table for the Turing machine. For example, the table could specify that in a certain state, after reading a certain symbol on a tape, the machine will in parallel write a different symbol on different tapes and then move one space left or right on each tape, all within a single computation step.

The paper describes a Turing machine which simulates a breadth-first descent through the infinity tree. There are a finite number of nodes at each level of the infinity tree. There is a finite path P from the root of the tree to each node N of the tree. For each node N at level L of the tree, the machine requires only finite time to write on its output tape the path from the root of the tree to node N. Thus, the machine takes only finite time to write the paths from the root of the tree to all the nodes at level L onto its output tape. Then, the machine takes only finite time to copy the paths on its output tape back to its input tape. It then uses the input tape to support computing the paths to the next level L + 1 of the tree and overwriting the output tape with paths to level L + 1.

The number of nodes at level L + 1 is ten times the number of nodes at level L. Yet the number of nodes at each finite depth level is always finite. For any finite natural number L, the machine only requires finite time to write the paths to all nodes at level L, and it only requires finite time to write the paths to all nodes at level L + 1, although the amount of finite time increases by a factor of 10 from

level L to level L + 1.

4. Further Discussion of the Turing Machine Argument

The second paper (2) discusses a Turing machine navigating the infinity tree breadth-first. For each level of the tree, all the paths from the root node to that level would be printed out by the Turing machine, going breadth-first across the level. This would generate a series of decimal expressions between 0 and 1, with n digits after the decimal point, for each level n of the tree. These expressions would be written in a sorted order, from 0 to 1.

If the Turing machine were to perform a countably infinite sequence of discrete time steps, then it could produce decimal expressions of infinite length corresponding to all decimal values in the range [0, 1], containing all the expressions on a countably infinite output tape. This list would correspond to all the infinite paths through the infinity tree, containing all the numbers in the interval [0, 1), plus the number 1.

Yet if we were to somehow look at the tape after the machine finishes its processing, then we would see the number 0.00… with a countable infinity of 0's, corresponding to the leftmost path descending through the infinity tree. We would have to somehow skip past this infinite list of 0's, to see the next number in the list, i.e., the next infinite path in the infinity tree.

The design of the Turing machine could be changed so that it only prints "0" and does not print "0." followed by a countable infinity of 0's. Yet there is still a countable infinity of numbers of the form 0.00000…d…, with no limit on the finite number of 0's there are after the decimal point and before the first non-zero decimal d. After that, there are countable infinities of other numbers of the form 0.m…, with the values of m being non-zero digits.

And there is no second decimal number which is the first number < 1 that is numerically greater than 0. For any number $0 < x < 1$, there is another number y such that $0 < y < x$.

So, how can there be a second number in the infinite list?

The answer again is that the Turing machine lists partial expressions for a finite list of numbers at each finite step. "After" an infinite number of processing steps, the machine has listed all the decimal expressions for all numbers in the range 0…1 in the linear space of the tape, so the tape contains these numbers in a sequence. However, we cannot, in finite time, observe a second number in the countably infinite list of all countably infinite expressions, starting from the first location on the tape.

The operation of the Turing machine could also be changed to generate infinity tree paths without unlimited leading zeros in the initial paths. The paths with leading zeros would still be generated eventually, but they would be intermixed with other paths. For example, using multiples of 7 the machine could initially select branch 7 at the first level of the infinity tree, then select branch 4 at the second level, then select branch 1 at the third level, then select branch 8 at the

fourth level, then select branch 5 at the fifth level, etc. This Turing machine would require a different finite state transition table to enable its operation, of course. The Appendix gives a Visual Basic program which simulates this operation if the user chooses not to generate branches in the standard order 0, 1, 2, 3, … In principle this operation could be replicated by a Turing machine simulator, just as a Turing machine simulator was written for generating branches in the standard order. (2)

As noted in (2), the idea of having a countably infinite tape that contains a countably infinite sequence of countably infinite sequences may seem self-contradictory, yet it makes sense when considered as something that the Turing machine creates using countably infinite time, and using a constant, finite set of countably infinite tapes.

And as noted in (2), it is well known that countable infinities can contain multiple countable infinities within them. For example, the countable infinity of natural numbers contains the countable infinities of even numbers, odd numbers, and prime numbers, and also countable infinities of randomly chosen natural numbers.

5. Summary

This short paper has addressed three questions: How do we know that the infinity tree has a countably infinite number of levels? How do we know that the Turing machine only requires finite time to process each level of the infinity tree? How can a Turing machine avoid generating infinite 0's at the start of its output if it processes the entire infinity tree?

Like the preceding papers (1) and (2), this paper does not claim there is an error in Cantor's arguments that [0, 1] is uncountably infinite. Rather, this paper considers the situation as a paradox, resulting from different choices about how to represent and count the continuum of real numbers. The preceding papers (1) and (2) include discussions of references [3]-[12]. These discussions have not been repeated in the present, short paper.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Jackson, P.C. (2023) The Infinity Tree: Representing Infinities of Real Numbers with Countably Infinite Tree Structures. *Advances in Pure Mathematics*, **13**, 198-205.

[2] Jackson, P.C. (2025) Countability of Infinite Paths in the Infinity Tree: Proof of the Continuum Hypothesis in a Non-Cantorian Infinity Theory. *Advances in Pure Mathematics*, **15**, 73-90.

[3] Cantor, G. (1891) On an Elementary Question in the Theory of Manifolds. In: Ewald, W.B., Ed., *From Immanuel Kant to David Hilbert: A Source Book in the Foundations of Mathematics*, Vol. 2, Oxford University Press, 920-922.

[4] Turing, A.M. (1937) On Computable Numbers, with an Application to the Entschei-

dungs problem. *Proceedings of the London Mathematical Society*, **2**, 230-265. https://doi.org/10.1112/plms/s2-42.1.230

[5] Minsky, M.L. (1967) Computation: Finite and Infinite Machines. Prentice Hall.

[6] Stannett, M. (2006) The Case for Hypercomputation. *Applied Mathematics and Computation*, **178**, 8-24.

[7] Hamkins, J.D. and Lewis, A. (2000) Infinite Time Turing Machines. *Journal of Symbolic Logic*, **65**, 567-604. https://doi.org/10.2307/2586556

[8] Cantor, G. (1878) Ein Beitrag zur Mannigfaltigkeitslehre. *Journal für die reine und angewandte Mathematik* (*Crelles Journal*), **84**, 242-258. https://doi.org/10.1515/crll.1878.84.242

[9] Hilbert, D. (1902) Mathematical Problems. *Bulletin of the American Mathematical Society*, **8**, 437-479. https://doi.org/10.1090/s0002-9904-1902-00923-3

[10] Gödel, K. (1940) The Consistency of the Continuum Hypothesis. Princeton University Press.

[11] Cohen, P.J. (1963) The Independence of the Continuum Hypothesis. *Proceedings of the National Academy of Sciences of the United States of America*, **50**, 1143-1148.

[12] Cantor, G. (1874) On a Property of the Set of All Real Algebraic Numbers. In: Ewald, W.B., Ed., *From Immanuel Kant to David Hilbert*: *A Source Book in the Foundations of Mathematics*, Vol. 2, Oxford University Press, 839-843.

Appendix: An Infinity Tree Generation Simulator

```
An Infinity Tree Generation Simulator
Dim MaxPathLevel As Double 'the maximum level of paths generated, in the tree
Dim MaxTapeLength As Double 'the maximum length of paths generated in testing
Dim MaxBranchTapes As Double

Dim PathLength As Double 'the current length of paths being generated

Dim RootLevel As Double 'always 0, the root node of the tree

Dim CurrentLevel As Double 'the current level for paths being generated

Dim minDigitValue As Double
Dim maxDigitValue As Double

Dim newMaxLength As Double
Dim MaxDecimalLength As Double

Dim Inputexprlen As Double
Dim Outputexprlen As Double

Dim FoundOutputEnd As Boolean

Dim InputPosition As Double
Dim OutputPosition As Double

Dim MostRecentRow As Double

Dim cellvalue As String
Dim previousOutputCellValue As String
Dim currentOutputCellValue() As String

Dim InputLevelpointer As Double
Dim OutputLevelpointer As Double

Dim InputLevelTape_CellValue() As String
Dim OutputLevelTape_CellValue() As String

Dim UserSelectedOrder() As Integer
Dim symbolIndex As Integer

Dim branch0Tape_pointer As Double
```

```
Dim branch1Tape_pointer As Double
Dim branch2Tape_pointer As Double
Dim branch3Tape_pointer As Double
Dim branch4Tape_pointer As Double
Dim branch5Tape_pointer As Double
Dim branch6Tape_pointer As Double
Dim branch7Tape_pointer As Double
Dim branch8Tape_pointer As Double
Dim branch9Tape_pointer As Double

Dim branch0Tape_CellValue() As String
Dim branch1Tape_CellValue() As String
Dim branch2Tape_CellValue() As String
Dim branch3Tape_CellValue() As String
Dim branch4Tape_CellValue() As String
Dim branch5Tape_CellValue() As String
Dim branch6Tape_CellValue() As String
Dim branch7Tape_CellValue() As String
Dim branch8Tape_CellValue() As String
Dim branch9Tape_CellValue() As String

Dim initialDigit As Integer
Dim initialIndex As Integer

Dim ChoiceMsg As String
Dim GenerateZeroBranchesFirst As Boolean
Dim Response As Integer

Sub AnInfinityTreePseudocodeSimulation()

    Dim keepworking As Boolean

    Dim Msg, Style, Title, Help, Ctxt, Response, MyString
    Title = "An Infinity Tree Pseudocode Simulation"
    ChoiceMsg = "Do you wish to generate branches in standard order
(0,1,2,3,...)?"
    Msg = "(c) 2025, by Philip C. Jackson, Jr., Ph.D." & vbCrLf & vbCrLf &
ChoiceMsg
    Style = vbYesNo 'Show Yes and No buttons.
    Response = MsgBox(Msg, Style, Title)
    If Response = 6 Then 'Visual Basic's response value for Yes...
        GenerateZeroBranchesFirst = True
    Else 'response = 7 'Visual Basic's response value for No...
```

```
            GenerateZeroBranchesFirst = False
        End If

        CurrentLevel = 1
        If GenerateZeroBranchesFirst Then
            initialDigit = 0 ' the initial digit at this level will be 0
        Else
            initialDigit = 7 ' the initial digit at this level will be 7
        End If
        initialIndex = 1 ' the index of the initial digit for this level is 1
        Call createInitialState 'this writes initial values to the input level tape

        Msg = "Created input tape for level " & CurrentLevel & ". Do you want to
continue?"
        Style = vbYesNo ' Define buttons.
        ' Display message.
        Response = MsgBox(Msg, Style, Title)

        If Response = vbYes Then        ' User chose Yes.
            MyString = "Yes"
            keepworking = True

           'position input level tape head to start of tape
            InputPosition = 1
            'position output level tape head to start of tape
            OutputPosition = 1

            Dim length As Double
            keepworking = True

            'an infinite loop to create expressions for paths from the root
            'to each level of infinity tree
            While keepworking = True
                'set the starting digit for the expressions that will be generated
                'at the next level of the tree
                initialIndex = initialIndex + 1
                If initialIndex > 10 Then intialindex = 1
                initialDigit = UserSelectedOrder(initialIndex)

                'for each expression on input level tape
                    'at each finite step there are a finite number of
                    'finite expressions, separated by spaces
                While locateNextInputDecimalExpression()
```

```
        'process expression and leave input tape pointer positioned
        'at the space after the expression
        keepworking = processNextInputDecimalExpression
        InputPosition = InputPosition + 1
    Wend

    'finished getting Decimals, so append "1" to output tape.
    'write a space and then a "1" to the output tape.
    'output pointer is at position after last symbol previously written.
    OutputLevelTape_CellValue(OutputPosition) = " "
    Worksheets("tapes").Cells(OutputPosition, 2).Value = " "
    FoundOutputEnd = False

    OutputPosition = OutputPosition + 1
    OutputLevelTape_CellValue(OutputPosition) = "1"
    Worksheets("tapes").Cells(OutputPosition, 2).Value = "1"

    Worksheets("tapes").Cells(OutputPosition, 2).Value = "1"

    CurrentLevel = CurrentLevel + 1
    Msg = "Computed output tape for level " & CurrentLevel & _
          ". Do you want to continue?"
    Style = vbYesNo ' Define buttons.
    Response = MsgBox(Msg, Style, Title)
    If Response = vbYes Then
        keepworking = True
    Else
        keepworking = False
    End If

    If keepworking Then
        'position input level tape head to start of input level tape
        InputPosition = 1
        'position output level tape head to start of output level tape
        OutputPosition = 1
        FoundOutputEnd = False
        previousOutputCellValue = ""
        MostRecentRow = 2

        While Not FoundOutputEnd
            cellvalue = OutputLevelTape_CellValue(OutputPosi-
tion)
            InputLevelTape_CellValue(OutputPosition) =
cellvalue
```

```
                    'add header row into location for output in worksheet
"tapes"
                    If previousOutputCellValue = " " And cellvalue = "1"
Then
                        FoundOutputEnd = True
                        InputLevelTape_CellValue(OutputPosition)     =
"1"
                        Worksheets("tapes").Cells(OutputPosition,
1).Value = "1"
                    Else
                        Worksheets("tapes").Cells(OutputPosition   +   1,
1).Value = _
                            cellvalue
                        If previousOutputCellValue = " " And cellvalue =
"0" _
                    Then MostRecentRow = 2
                        Worksheets("tapes").Cells(MostRecentRow,
14).Value = _
                            cellvalue
                        MostRecentRow = MostRecentRow + 1
                    End If
                    previousOutputCellValue = cellvalue
                    OutputPosition = OutputPosition + 1
                Wend

                'newMaxLength includes "0." so subtract 2 to compare
                'with max decimal length
                If (newMaxLength - 2) < MaxDecimalLength Then
                    Msg = "Copied output tape to input tape for level " & _
                            CurrentLevel & ". Do you want to continue?"
                    Response = MsgBox(Msg, Style, Title)
                    If Response = vbYes Then
                        keepworking = True
                    Else
                        keepworking = False
                    End If
                Else
                    keepworking = False
                End If
            End If

            'position input level tape head to start of input level tape
            InputPosition = 1
```

```
                'position output level tape head to start of output level tape
                OutputPosition = 1

            'end loop 'infinite loop creating expressions for paths from
                    'root to each level of the infinity tree
            Wend

            Msg = "The demonstration is over."
            Style = vbDefaultButton2        ' just show OK button.
            Response = MsgBox(Msg, Style, Title)

        Else    ' User chose No.
            keepworking = False

            Msg = "The demonstration is over."
            Style = vbDefaultButton2        ' just show OK button.
            Response = MsgBox(Msg, Style, Title)
        End If
End Sub


'find the next non-blank character that happens after a space in the input tape
'if it's a 0 then return true
'if it's a 1 then return false
'set global var pointing to location of the next non-blank character
Function locateNextInputDecimalExpression() As Boolean

    Dim lookingforleadingzero As Boolean
    Dim lookingforblankspace As Boolean
    Dim lookingforleadingOne As Boolean

    Dim exprlen As Double

    'get character of expression at input position
    cellvalue = InputLevelTape_CellValue(InputPosition)

    'if the first character of the expression is not "1" then
    If cellvalue <> "1" Then
        'verify that cellvalue is "0", as expected
        If cellvalue = "0" Then
            'position branch tape heads to start of branch tapes
            exprlen = 1
            'return true, indicating this routine got a next Decimal expres-
```

```
sion
                locateNextInputDecimalExpression = True
            Else
                'we may already be one step past the "1"
                'return false, indicating this routine did not get a next
                'Decimal expression
                locateNextInputDecimalExpression = False
            End If
        Else
            'return false, indicating this routine did not get a next Decimal ex-
pression
            locateNextInputDecimalExpression = False
        End If

    End Function

    Function processNextInputDecimalExpression() As Boolean

        Dim lookingforleadingzero As Boolean
        Dim lookingforblankspace As Boolean
        Dim lookingforleadingOne As Boolean
        Dim exprlen As Double
        Dim cellvalue As String
        Dim nextDigit As Integer
        Dim randomDigit As Integer

        'get first character of expression
        cellvalue = InputLevelTape_CellValue(InputPosition)

        'if the first character of the expression is not "1" then
        If cellvalue <> "1" Then
            'we are not at the end of the input level tape's expressions.
            'position branch tape heads to start of branch tapes
            exprlen = 1
            'copy the expression to each branch tape
            While cellvalue <> " "
                branch0Tape_CellValue(exprlen) = cellvalue
                branch1Tape_CellValue(exprlen) = cellvalue
                branch2Tape_CellValue(exprlen) = cellvalue
                branch3Tape_CellValue(exprlen) = cellvalue
                branch4Tape_CellValue(exprlen) = cellvalue
                branch5Tape_CellValue(exprlen) = cellvalue
                branch6Tape_CellValue(exprlen) = cellvalue
```

```
        branch7Tape_CellValue(exprlen) = cellvalue
        branch8Tape_CellValue(exprlen) = cellvalue
        branch9Tape_CellValue(exprlen) = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 1).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 2).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 3).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 4).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 5).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 6).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 7).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 8).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 9).Value = cellvalue
        Worksheets("tapes").Cells(exprlen + 1, 2 + 10).Value = cellvalue
        InputPosition = InputPosition + 1
        exprlen = exprlen + 1
        cellvalue = InputLevelTape_CellValue(InputPosition)
    Wend

    If GenerateZeroBranchesFirst Then
        'generate branches in 0...9 order
        branch0Tape_CellValue(exprlen) = "0"
        branch1Tape_CellValue(exprlen) = "1"
        branch2Tape_CellValue(exprlen) = "2"
        branch3Tape_CellValue(exprlen) = "3"
        branch4Tape_CellValue(exprlen) = "4"
        branch5Tape_CellValue(exprlen) = "5"
        branch6Tape_CellValue(exprlen) = "6"
        branch7Tape_CellValue(exprlen) = "7"
        branch8Tape_CellValue(exprlen) = "8"
        branch9Tape_CellValue(exprlen) = "9"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 1).Value = "0"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 2).Value = "1"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 3).Value = "2"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 4).Value = "3"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 5).Value = "4"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 6).Value = "5"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 7).Value = "6"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 8).Value = "7"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 9).Value = "8"
        Worksheets("tapes").Cells(exprlen + 1, 2 + 10).Value = "9"

    Else 'add digits in the sequence 7,4,1,8,5,2,9,6,3,0
        'with the first digit from the sequence based on the level in the
tree
```

```
initialDigit = UserSelectedOrder(initialIndex)

'the first, initial digit, already chosen for this level of the tree,
'was chosen before calling this routine
nextDigit = initialDigit
branch0Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 1).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch1Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 2).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch2Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 3).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch3Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 4).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch4Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 5).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch5Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 6).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch6Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 7).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch7Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 8).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch8Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 9).Value = _
    CStr(nextDigit)
nextDigit = getPermutedNextDigit(nextDigit)
branch9Tape_CellValue(exprlen) = CStr(nextDigit)
Worksheets("tapes").Cells(exprlen + 1, 2 + 10).Value = _
```

```
                    CStr(nextDigit)
                End If

                newMaxLength = exprlen

                'append copy of branch tape expression to output tape
                For branch = 0 To 9
                    exprlen = 1
                    MostRecentRow = 2 'will be used to show most recent value written
                        'to output or input tape
                    For exprlen = 1 To newMaxLength

                        If branch = 0 Then cellvalue = branch0Tape_CellValue(exprlen)
                        If branch = 1 Then cellvalue = branch1Tape_CellValue(exprlen)
                        If branch = 2 Then cellvalue = branch2Tape_CellValue(exprlen)
                        If branch = 3 Then cellvalue = branch3Tape_CellValue(exprlen)
                        If branch = 4 Then cellvalue = branch4Tape_CellValue(exprlen)
                        If branch = 5 Then cellvalue = branch5Tape_CellValue(exprlen)
                        If branch = 6 Then cellvalue = branch6Tape_CellValue(exprlen)
                        If branch = 7 Then cellvalue = branch7Tape_CellValue(exprlen)
                        If branch = 8 Then cellvalue = branch8Tape_CellValue(exprlen)
                        If branch = 9 Then cellvalue = branch9Tape_CellValue(exprlen)
                        While cellvalue <> " "
                            OutputLevelTape_CellValue(OutputPosition) = cellvalue
                            Worksheets("tapes").Cells(OutputPosition + 1, 2).Value = _
                                cellvalue
                            Worksheets("tapes").Cells(MostRecentRow, 14).Value = _
                                cellvalue
                            exprlen = exprlen + 1
```

```
                    OutputPosition = OutputPosition + 1
                    MostRecentRow = MostRecentRow + 1
                    If branch = 0 Then cellvalue = branch0Tape_CellValue(exprlen)
                    If branch = 1 Then cellvalue = branch1Tape_CellValue(exprlen)
                    If branch = 2 Then cellvalue = branch2Tape_CellValue(exprlen)
                    If branch = 3 Then cellvalue = branch3Tape_CellValue(exprlen)
                    If branch = 4 Then cellvalue = branch4Tape_CellValue(exprlen)
                    If branch = 5 Then cellvalue = branch5Tape_CellValue(exprlen)
                    If branch = 6 Then cellvalue = branch6Tape_CellValue(exprlen)
                    If branch = 7 Then cellvalue = branch7Tape_CellValue(exprlen)
                    If branch = 8 Then cellvalue = branch8Tape_CellValue(exprlen)
                    If branch = 9 Then cellvalue = branch9Tape_CellValue(exprlen)
                Wend
                'add an empty space to output tape
                OutputLevelTape_CellValue(OutputPosition) = " "
                Worksheets("tapes").Cells(OutputPosition + 1, 2).Value = "
"

                'increment position for next character on output tape
                OutputPosition = OutputPosition + 1

            Next exprlen

        Next branch

    End If

    'making this routine a function prevents it from being an option
    'in the startup menu. ...
    processNextInputDecimalExpression = True

End Function

Function getPermutedNextDigit(digit As Integer) As Integer
```

```
    Dim nextDigit As Integer
    If digit = 7 Then
        nextDigit = 4
    ElseIf digit = 4 Then
        nextDigit = 1
    ElseIf digit = 1 Then
        nextDigit = 8
    ElseIf digit = 8 Then
        nextDigit = 5
    ElseIf digit = 5 Then
        nextDigit = 2
    ElseIf digit = 2 Then
        nextDigit = 9
    ElseIf digit = 9 Then
        nextDigit = 6
    ElseIf digit = 6 Then
        nextDigit = 3
    ElseIf digit = 3 Then
        nextDigit = 0
    ElseIf digit = 0 Then
        nextDigit = 7
    End If
    getPermutedNextDigit = nextDigit
End Function

'this is defined as a function, so that it doesn't show up in the user menu...
'yet VB allows it to be called as a subroutine
Function createInitialState() As Boolean
    'define and allocate space for each tape
    'in theory, each tape has a countable infinity of cells
    'in reality for any simulation, each tape has a finite length

    Dim integerCellValue As Integer
    Dim integerIndex As Integer

    'this routine creates input expressions of length 3, e.g. "0.0"
    Inputexprlen = 3

    MaxTapeLength = 800100
    'using a column of a spreadsheet to represent a tape, each row
    'represents a cell of a tape.
    'this many rows are needed to support output for 5 decimals,
    'with 1 decimal per row...
```

```
'this is sufficient for a demo, so there's no need to make the
'maximum tape length be an input parameter in a separate worksheet
MaxDecimalLength = 5 '5 is the most that can be supported by this
'code in VBA with tape length = 800,100

'MaxPathLevel = MaxTapeLength
'considering root node as level 0, with tape length 0.
'a variable for maximum path level is not needed by this simulator.
'max tape length is sufficient.

MaxBranchTapes = 9 '0 through 9
RootLevel = 0

minDigitValue = 0
maxDigitValue = 9

ReDim UserSelectedOrder(1 To 10)

If GenerateZeroBranchesFirst Then
    UserSelectedOrder(1) = 0
    UserSelectedOrder(2) = 1
    UserSelectedOrder(3) = 2
    UserSelectedOrder(4) = 3
    UserSelectedOrder(5) = 4
    UserSelectedOrder(6) = 5
    UserSelectedOrder(7) = 6
    UserSelectedOrder(8) = 7
    UserSelectedOrder(9) = 8
    UserSelectedOrder(10) = 9
Else
    UserSelectedOrder(1) = 7
    UserSelectedOrder(2) = 4
    UserSelectedOrder(3) = 1
    UserSelectedOrder(4) = 8
    UserSelectedOrder(5) = 5
    UserSelectedOrder(6) = 2
    UserSelectedOrder(7) = 9
    UserSelectedOrder(8) = 6
    UserSelectedOrder(9) = 3
    UserSelectedOrder(10) = 0
End If

'the first cell of every tape will be counted as cell # 1
```

```
ReDim InputLevelTape_CellValue(1 To MaxTapeLength)
ReDim OutputLevelTape_CellValue(1 To MaxTapeLength)

ReDim branch0Tape_CellValue(1 To MaxTapeLength)
ReDim branch1Tape_CellValue(1 To MaxTapeLength)
ReDim branch2Tape_CellValue(1 To MaxTapeLength)
ReDim branch3Tape_CellValue(1 To MaxTapeLength)
ReDim branch4Tape_CellValue(1 To MaxTapeLength)
ReDim branch5Tape_CellValue(1 To MaxTapeLength)
ReDim branch6Tape_CellValue(1 To MaxTapeLength)
ReDim branch7Tape_CellValue(1 To MaxTapeLength)
ReDim branch8Tape_CellValue(1 To MaxTapeLength)
ReDim branch9Tape_CellValue(1 To MaxTapeLength)

'create initial starting state of tapes:
'initially, all tapes have only blank spaces on every cell
For i = 1 To MaxTapeLength
    InputLevelTape_CellValue(i) = " "
    OutputLevelTape_CellValue(i) = " "
    branch0Tape_CellValue(i) = " "
    branch1Tape_CellValue(i) = " "
    branch2Tape_CellValue(i) = " "
    branch3Tape_CellValue(i) = " "
    branch4Tape_CellValue(i) = " "
    branch5Tape_CellValue(i) = " "
    branch6Tape_CellValue(i) = " "
    branch7Tape_CellValue(i) = " "
    branch8Tape_CellValue(i) = " "
    branch9Tape_CellValue(i) = " "
Next i

Application.Calculation = xlManual
Application.ScreenUpdating = False
Sheets("tapes").UsedRange.Delete
Sheets("tapes").UsedRange.HorizontalAlignment = xlRight
Application.ScreenUpdating = True
Application.Calculation = xlAutomatic

'need to restore the column headers after the cells are cleared.
Worksheets("tapes").Cells(1, 1).Value = "input tape"
Worksheets("tapes").Cells(1, 2).Value = "output tape"
Worksheets("tapes").Cells(1, 3).Value = "branch 0"
Worksheets("tapes").Cells(1, 4).Value = "branch 1"
```

```
Worksheets("tapes").Cells(1, 5).Value = "branch 2"
Worksheets("tapes").Cells(1, 6).Value = "branch 3"
Worksheets("tapes").Cells(1, 7).Value = "branch 4"
Worksheets("tapes").Cells(1, 8).Value = "branch 5"
Worksheets("tapes").Cells(1, 9).Value = "branch 6"
Worksheets("tapes").Cells(1, 10).Value = "branch 7"
Worksheets("tapes").Cells(1, 11).Value = "branch 8"
Worksheets("tapes").Cells(1, 12).Value = "branch 9"
Worksheets("tapes").Cells(1, 14).Value = "most recent #"

'position tape heads to start of all tapes
branch0Tape_pointer = 1
branch1Tape_pointer = 1
branch2Tape_pointer = 1
branch3Tape_pointer = 1
branch4Tape_pointer = 1
branch5Tape_pointer = 1
branch6Tape_pointer = 1
branch7Tape_pointer = 1
branch8Tape_pointer = 1
branch9Tape_pointer = 1

integerIndex = 1

InputPosition = 1

For integerIndex = 1 To 10
    InputLevelTape_CellValue(InputPosition) = "0"
    Worksheets("tapes").Cells(1 + InputPosition, 1).Value = _
        InputLevelTape_CellValue(InputPosition)
    InputPosition = InputPosition + 1
    InputLevelTape_CellValue(InputPosition) = "."
    Worksheets("tapes").Cells(1 + InputPosition, 1).Value = _
        InputLevelTape_CellValue(InputPosition)
    InputPosition = InputPosition + 1
    integerCellValue = UserSelectedOrder(integerIndex)
    InputLevelTape_CellValue(InputPosition) = CStr(integerCellValue)
    Worksheets("tapes").Cells(1 + InputPosition, 1).Value = _
        InputLevelTape_CellValue(InputPosition)
    InputPosition = InputPosition + 1
    InputLevelTape_CellValue(InputPosition) = " "
    Worksheets("tapes").Cells(1 + InputPosition, 1).Value = _
        InputLevelTape_CellValue(InputPosition)
```

```
            InputPosition = InputPosition + 1
        Next integerIndex

        InputLevelTape_CellValue(InputPosition) = "1"
        Worksheets("tapes").Cells(1 + InputPosition, 1).Value = _
            InputLevelTape_CellValue(InputPosition)

        'making this routine a function prevents it from being an
        'option in the startup menu. ...
        createInitialState = True

    End Function
```