DECREMENTAL TAG SYSTEMS AND RANDOM TREES

PATRICK BAHLS, JOSH KNOX, AND MARK MCCLURE

ABSTRACT. We introduce a variation of Post's tag systems that leads to a finite state machine. Our system is simpler than those considered by Post, in that there are only finitely many states. It is more complicated, in that any given state can evolve in multiple directions. Most importantly, we are able to analyze the system fairly completely and use it to investigate the properties of certain types of randomly generated trees.

1. INTRODUCTION

We fix a positive natural number m and consider sequences of the form $\boldsymbol{x}^n = (x_1, \ldots, x_n)$, where each $x_i \in \{0, 1, \ldots, m\}$ and $x_i \neq 0$ for some i. Denote the set of all such sequences by X_n. Given $\boldsymbol{x} \in X_n$ and a fixed $i \in \{1, 2, \ldots, n\}$ satisfying $x_i \neq 0$, define $\sigma_i(\boldsymbol{x})$ by

$$\sigma_i\left((x_1, \ldots, x_n)\right) = (x_2, \ldots, x_i - 1, \ldots, x_n, m)\,. \tag{1}$$

Note that the domain of each σ_i is a proper subset of X_n that contains *only* those $\boldsymbol{x}$ such that $x_i \neq 0$. Also, when σ_1 is applicable, it essentially simply shifts and appends since the decremented element disappears off the end.

This decrement and shift operation provides a simple modification of the concept of "tag system" introduced by Emil Post [1] and studied much later by Wolfram [2]. Post's tag systems allow blocks of arbitrary length to be removed from the beginning and tagged onto the end according to various rules. Since the lengths of the removed block and the appended block need not be equal, the lengths of the terms in the resulting sequence under iteration can vary resulting in infinitely many states and very complicated behavior. Our system is simpler, since there are only finitely many states; in fact, $(m+1)^n - 1$ states. However, most states can spawn multiple children resulting in many possible futures. In this sense, it is analogous to the multi-state systems studied by Wolfram [2]. This process is naturally modeled by a directed graph with labeled edges or a finite state machine. The vertices are indexed by the states. We place a directed edge from vertex $\boldsymbol{x}$ to vertex $\boldsymbol{y}$ and label it with an i, if $\sigma_i(\boldsymbol{x}) = \boldsymbol{y}$. The labeled directed graph model for $m = n = 2$ is shown in figure 1.

We are particularly interested in identifying the recurrent states, those admitting a closed path from the state to itself. The recurrent states in figure 1 are simply $(1, 2)$ and $(2, 2)$. It is a fairly simple matter to model the decremental tag process on a computer, construct the graph model, and determine the recurrent states using standard graph algorithms. The recurrent states can be determined much more efficiently, however, using the particular structure of this system. It turns out that these *decremental tag systems* can be used to model an interesting random tree construction we call the *use it or lose it construction.* In fact, our original

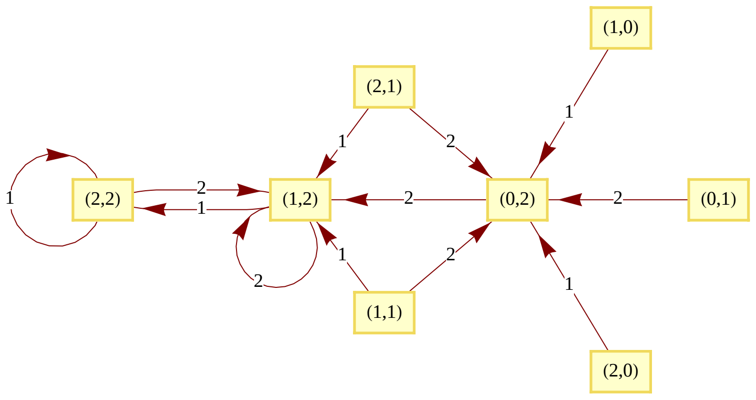

FIGURE 1. The digraph model for $m = n = 2$

motivation was to compute certain parameters describing the expected structure of such a randomly generated tree.

In section two, we describe our recursive procedure to efficiently find the recurrent states of the decremental tag process. The arguments are fairly delicate, but we find the resulting procedure to be simple and elegant. In section three, we apply the results of section two to random trees.

2. THE RECURRENT STATES

Recall that a state is recurrent if there is a closed path from that vertex to itself. We will denote the set of recurrent states in X_n by $R(n)$. The objective in this section is to characterize the recurrent states and provide a recursive construction of $R(n)$. Following standard terminology for directed graphs, a *walk* is a sequence of vertices $(u_1, u_2, \ldots, u_n)$ so that there is a directed edge from u_i to u_{i+1} for each i. A *path* is a walk consisting of distinct vertices and a *closed path* is a walk consisting of distinct vertices, except for the beginning and ending vertices which must be the same.

In our situation, a walk can also be described by the labels of the edges, since distinct labels emanating from a vertex lead to distinct terminal vertices. Thus a walk of length k is determined by an initial vertex and a sequence of labels $(i_1, \ldots, i_k)$, provided that σ_i is never applied to a state with $x_i = 0$. We will call a walk described in this fashion a *legal walk* and will use similar terminology for *legal path* and *closed legal path*. The terminal vertex of a legal walk will be called the *destination* of the walk. As it turns out, the destination of a walk is independent of the starting point, provided the walk is long enough.

Lemma 1. *Suppose that $\boldsymbol{i} = (i_1, \ldots, i_k)$ defines a legal walk starting from $\boldsymbol{x} = (x_1, \ldots, x_n)$ with destination $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$ and that $\boldsymbol{i}$ also defines a legal walk from $\boldsymbol{y} = (y_1, \ldots, y_n)$ with destination $\boldsymbol{z'} = (z'_1, \ldots, z'_n)$. Then, $z_j = z'_j$ for $j > \max(n-k, 0)$.*

Proof. The lemma is trivial if $k = 1$, since the last element is an m after one step. Now suppose that the statement is true for a fixed natural number k and consider a legal walk of length $k+1$. By induction, the last $\min(k,n)$ terms are the same after the first k steps. Thus, for $k<n$, the situation looks like

$$(x_1,\ldots,x_n) \xrightarrow{(i_1,\ldots,i_k)} (z_1,\ldots z_{n-k}, z_{n-k+1},\ldots,z_n)$$

$$(y_1,\ldots,y_n) \xrightarrow{(i_1,\ldots,i_k)} \left(z'_1,\ldots z'_{n-k}, z_{n-k+1},\ldots,z_n\right).$$

Note that the last k terms are the same. After one more step, we have

$$(x_1,\ldots,x_n) \xrightarrow{(i_1,\ldots,i_k,i_{k+1})} \left(\bar{z}_2,\ldots \bar{z}'_{n-k}, \bar{z}_{n-k+1},\ldots,\bar{z}_n, m\right)$$

$$(y_1,\ldots,y_n) \xrightarrow{(i_1,\ldots,i_k,i_{k+1})} \left(\bar{z}'_2,\ldots \bar{z}'_{n-k}, \bar{z}_{n-k+1},\ldots,\bar{z}_n, m\right),$$

where

$$\bar{z}_i = \begin{cases} z_i & \text{if } i \neq i_{k+1} \\ z_i - 1 & \text{if } i = i_{k+1}. \end{cases}$$

A similar definition holds for $\bar{z}'$. Now, the last $\min(k+1,n)$ terms are the same, yielding the result. □

An immediate consequence is the following.

Corollary 1. *Suppose that $(i_1,\ldots,i_k)$ defines legal walks starting from both the vertices $\boldsymbol{x}=(x_1,\ldots,x_n)$ and $\boldsymbol{y}=(y_1,\ldots,y_n)$. Then, the destinations of those walks are the same, provided $k \geq n$.*

We can now state the following characterization of the recurrent states.

Lemma 2. *For $m>2$, let $\boldsymbol{x}_0 \in R(n)$ denote the special (clearly recurrent) state $(m,\ldots,m)$. Then $R(n)$ consists precisely of those states that are accessible from $\boldsymbol{x}_0$.*

Proof. It's very easy to see that $\boldsymbol{x}_0$ is accessible from any other state; simply iteratively choose the smallest possible i such that $x_i \neq 0$ to generate a path from the given state to $\boldsymbol{x}_0$. It follows that any state which is accessible from $\boldsymbol{x}_0$ is recurrent. Conversely, suppose that $\boldsymbol{x}$ is a recurrent state. Thus, there is a closed path from $\boldsymbol{x}$ to itself. By repeating this path if necessary, we generate a legal walk of length $k \geq n$ from $\boldsymbol{x}$ to itself. If the walk is legal from $\boldsymbol{x}$, then it is certainly legal from $\boldsymbol{x}_0$, since the state values can only be larger. Since the length of the walk exceeds n, the destination is the same by lemma 1. Thus $\boldsymbol{x}$ is accessible from $\boldsymbol{x}_0$. □

Lemma 2 and its proof yield the following corollary.

Corollary 2. *The directed graph that models the recurrent states is strongly connected.*

Our recursive construction of the recurrent states will depend on the nature of shortest paths from $\boldsymbol{x_0}$. The following lemma essentially states that such a path does not shift decremented elements off the end.

Lemma 3. *Suppose there are one or more paths from $\boldsymbol{x_0}$ to $\boldsymbol{x}$. Clearly, there is a path $(i_1,\ldots,i_k)$ of shortest length. This shortest path satisfies $i_j > k-(j-1)$ for each j.*

Proof. Suppose to the contrary that $(i_1, \ldots, i_k)$ is a path of shortest length from $\boldsymbol{x_0}$ to $\boldsymbol{x}$ and that the inequality is not satisfied for some j. Thus,

$$(m, \ldots, m) \xrightarrow{(i_1,\ldots,i_j)} \big(y_1, \ldots, y_{i_j}, y_{i_j+1}, \ldots, y_n\big) \xrightarrow{(i_{j+1},\ldots,i_k)} (x_1, \ldots, x_n)$$

Focusing on the first $j-1$ steps, we see

$$(m, \ldots, m) \xrightarrow{(i_1,\ldots,i_{j-1})} \big(z, y_1, \ldots, y_{i_j}, y_{i_j+1}, \ldots, y_{n-1}\big)$$

The value of z is unimportant, as it will be shifted off at the next step. Since the initial state $\boldsymbol{x_0}$ has all elements the same, we could have applied $(i_1 - 1, \ldots, i_{j-1} - 1)$ and found

$$(m, \ldots, m) \xrightarrow{(i_1-1,\ldots,i_{j-1}-1)} \big(y_1, \ldots, y_{i_j}, y_{i_j+1}, \ldots, y_{n-1}, m\big)$$

Of course, $y_n = m$. Thus, the result after application of $(i_1 - 1, \ldots, i_{j-1} - 1)$ is the same as the result after application of $(i_1, \ldots, i_j)$. Therefore, $(i_1 - 1, \ldots, i_{j-1} - 1, i_{j+1}, \ldots, i_k)$ is a shorter path from $\boldsymbol{x_0}$ to $\boldsymbol{x}$. This is contrary to our initial assumption so the inequality must be satisfied for shortest paths. □

There is no a priori reason to suppose that k is restricted by n. However, taking $j = 1$ in the previous lemma and using the fact that $i_1 \leq n$ yields the following corollary.

Corollary 3. *Given any $\boldsymbol{x} \in R(n)$, there is a path from $\boldsymbol{x}_0$ to $\boldsymbol{x}$ with length at most $n-1$.*

For the remainder of this section, we will be interested in the relationship between $R(n)$ and $R(n+1)$. For clarity, we will typically denote an element of $R(n)$ with a superscript. Thus $\boldsymbol{x}^n$ denotes the typical element in $R(n)$ and $\boldsymbol{x}_0^n \in R(n)$ denotes $(m, \ldots, m)$. A simple example of this type is the following lemma.

Lemma 4. *Suppose that $\boldsymbol{x}^n = (x_1, \ldots, x_n) \in R(n)$. Then $\boldsymbol{x}^{n-1} = (x_2, \ldots, x_n) \in R(n-1)$ and $\boldsymbol{x}^{n+1} = (m, x_1, \ldots, x_n) \in R(n+1)$.*

Proof. If $(i_1, \ldots, i_k)$ is a shortest legal path from $\boldsymbol{x}_0^n$ to $\boldsymbol{x}^n$, then $(i_1 + 1, \ldots, i_k + 1)$ is a legal path from $\boldsymbol{x}_0^{n+1}$ to $\boldsymbol{x}^{n+1}$ and $(i_1 - 1, \ldots, i_k - 1)$ is a legal path from $\boldsymbol{x}_0^{n-1}$ to $\boldsymbol{x}^{n-1}$, deleting zeros if necessary. □

Using these same ideas, we can now present our recursive construction of $R(n)$.

Theorem 1. *Fix $m \geq 2$ and let $R(n)$ denote the set of recurrent states for the decremental tag system on sequences of length n chosen from the symbols $\{0, 1, \ldots, m\}$. Define the set $S(n)$ recursively as follows:*

- *$S(1) = \{(m)\}$*
- *Given $(x_1, \ldots, x_n) \in S(n)$, and a non-negative integer $r \leq m$, $(m - r, x_1, \ldots, x_n) \in S(n+1)$ iff $x_1 + \cdots + x_n \geq (m-1)n + r$.*

Then $S(n) = R(n)$.

Proof. We prove the theorem by induction. For $n = 1$, the theorem states that (m) is the only recurrent state, which is clearly true. Now, fix n and assume for induction that $S(n') = R(n')$ for all $n' \leq n$.

We first show that $R(n+1) \subset S(n+1)$. Let $\boldsymbol{x}^n = (x_1, \ldots, x_n) \in R(n)$ and suppose that $x_1 + \cdots + x_n < (m-1)n + r$. We must show that $\boldsymbol{x}^{n+1} = (m - r, x_1, \ldots, x_n) \notin R(n+1)$. Now, since each application of a σ_i decrements a value, the number of steps required to get from $\boldsymbol{x_0}^{n+1}$ to $\boldsymbol{x}^{n+1}$ is $m(n+1)$ minus the sum of the terms in $\boldsymbol{x}^{n+1}$. Thus, a shortest path from $\boldsymbol{x}_0^{n+1}$ to $\boldsymbol{x}^{n+1}$ requires $mn-(x_1+\cdots+x_n)+r>n-1$ steps and the result now follows from corollary 3.

To show that $S(n+1)\subset R(n+1)$, suppose that $\boldsymbol{x}^{n+1}\in S(n+1)$. Thus, $\boldsymbol{x}^{n+1}=(m-r,x_1,\ldots,x_n)$ where $x_1+\cdots+x_n\geq(m-1)n+r$ and $\boldsymbol{x}^n=(x_1,\ldots,x_n)\in S(n)=R(n)$. Now let $\boldsymbol{x}_{r,k}^{n+1}=(m,\ldots,m,m-r,m,\ldots,m)$, where the $m-r$ term appears in position $k+1$. We will show that $\boldsymbol{x}_{r,k}^{n+1}\in R(n+1)$ and that there is a legal walk from $\boldsymbol{x}_{r,k}^{n+1}$ to $\boldsymbol{x}^{n+1}$. This implies that $\boldsymbol{x}^{n+1}\in R(n+1)$.

To show that $\boldsymbol{x}_{r,k}^{n+1}\in R(n+1)$, we first note that $(m-r,m,\ldots,m)\in S(n-k+1)$. This is because we may prepend $m-r$ to $\boldsymbol{x}_0^{n-k}\in S(n-k)$ since $(n-k)m\geq(m-1)(n-k)-r$. Thus, $(m-r,m,\ldots,m)\in R(n-k+1)=S(n-k+1)$ by the induction hypotheses. Then $\boldsymbol{x}_{r,k}^{n+1}\in R(n+1)$, since we can simply prepend k ms to $(m-r,m,\ldots,m)$ to generate $\boldsymbol{x}_{r,k}^{n+1}$.

Now, let $(i_1,\ldots,i_k)$ be a shortest path from $\boldsymbol{x}_0^n$ to $\boldsymbol{x}^n$. We claim that $(i_1+1,\ldots,i_k+1)$ is a legal walk from $\boldsymbol{x}_{r,k}^{n+1}$ to $\boldsymbol{x}_0^{n+1}$. This is almost immediate. Note that the inequalities $i_j>k-(j-1)$ for $j=1,\ldots,k$ imply that the m in position k of $\boldsymbol{x}_0^n$ is never decremented by the action of $(i_1,\ldots,i_k)$. Thus, the $m-r$ in position $k+1$ of $\boldsymbol{x}_{r,k}^{n+1}$ is never decremented under the action of $(i_1+1,\ldots,i_k+1)$. □

This recursive construction is easy to implement on a computer and we have done so in Mathematica. Furthermore, the construction of these recurrent states is a key step in the application described in the next section.

3. Application to use it or lose it trees

The study of trees generated by a random process has a long history. We introduce here a variation of the classical construction of Erdös and Rényi [3] that we call the *use it or lose it* construction. It turns out that decremental tag systems offer a tool to analyze the expected structure of use it or lose it trees. Start with any tree whose vertices are labeled with integers chosen from $\{0,1,2,\ldots,n\}$. The label of a vertex indicates its life expectancy and the labels need not be distinct. A vertex labeled 0 is dead. We then construct a new tree by randomly choosing one of the live vertices of the given tree and attaching $m-1$ new leaves to the chosen vertex. All the new leaves have life expectancy n and the life expectancy of the chosen vertex is reset to n (hence the name of the process). Figure 2 shows a few steps in this process corresponding to $m=n=2$ and figure 3 shows a graph after 150 steps in the process. Note that since $m=2$, we attach $m-1$ nodes to the chosen leaf resulting in two nodes of maximum life expectancy in the new tree.

It is now natural to ask what properties a large tree generated from this process might have. What, for example, is the probability that a vertex chosen randomly from the tree is a leaf? We call this the *leaf probability* question. It's fairly simple to answer the question when each vertex has lifespan one and $m-1$ leaves are added at each step. A vertex born into such a tree has one chance out of m to be selected at the next level and not die as a leaf. Thus with probability $(m-1)/m$, any given vertex will die as a leaf. The situation is more complicated when the lifespan is larger, but our computations indicate the leaf probability is independent of the lifespan. We have no general proof of this fact, rather we have an algorithm (using decremental tag systems) that may be used to compute the leaf probability

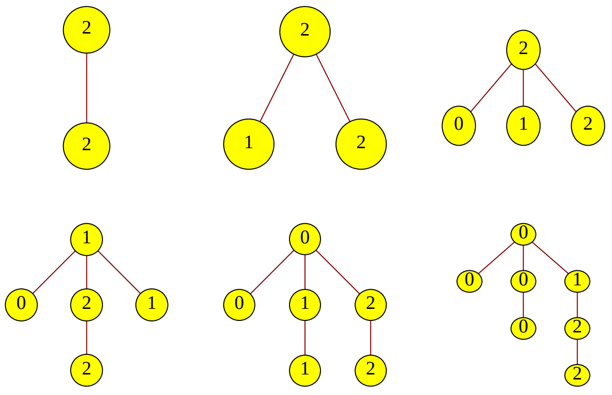

FIGURE 2. A few use it or lose it steps

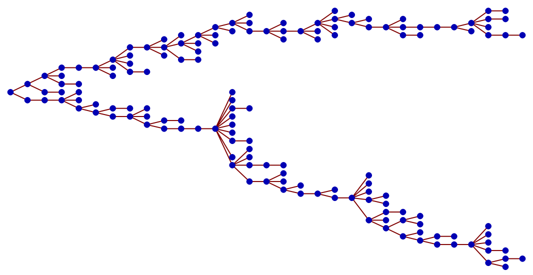

FIGURE 3. 150 use it or lose it steps

for particular m and n. Based on those computations, we conjecture that the leaf probability is indeed independent of lifespan.

In order to model the use it or lose it process with a decremental tag system, we introduce the state vector of a tree, $\boldsymbol{x} = (x_1, \ldots, x_n)$, where x_i indicates the number of vertices in the tree with life expectancy i. If we choose a vertex with life expectancy i from a tree in state $\boldsymbol{x}$, we generate a tree in state $\sigma_i(\boldsymbol{x})$, where σ is exactly the decremental tag operation. The parameter $m-1$ in the use it or lose it description is chosen precisely to obtain this direct correspondence.

We now consider the probability that a leaf born into a particular state eventually dies as a leaf; these may be computed recursively. Clearly a leaf with remaining life expectancy one in a tree in state $\boldsymbol{x}$ has probability $(x_1 + \cdots + x_n - 1) \,/\, (x_1 + \cdots + x_n)$ of dying and remaining a leaf. The probability that a leaf with higher life expectancy i remains a leaf may be computed using conditional probabilities. More precisely, let $P(\boldsymbol{x}, i)$ denote the probability that a leaf in a state $\boldsymbol{x}$ tree with remaining life expectancy i eventually dies as a leaf. Then,

$$P(\boldsymbol{x}, i) = \sum_{j=1}^{n} \frac{x_j - \delta_{i,j}}{x_1 + \cdots + x_n} P\left(\sigma_j(\boldsymbol{x}), i-1\right). \tag{2}$$

The $\delta_{i,j}$ in formula 2 is the Kronecker delta. Thus,

$$(x_j - \delta_{i,j}) \,/\, (x_1 + \cdots + x_n) \tag{3}$$

represents the probability of picking a vertex with lifespan j, where we subtract off a 1 in the particular case that $j = i$. After picking that vertex, with lifespan say j, we generate a new tree in state $\sigma_j(\boldsymbol{x})$ and the leaf under consideration has remaining life expectancy $i - 1$. Thus formula 2 arises by computing conditional probabilities and summing over all j. Since we know each $P(\boldsymbol{x}, 1)$, this can be used to compute each $P(\boldsymbol{x}, i)$ recursively. The asymptotic probability that a randomly chosen vertex from a large use it or lose it tree is a leaf is then a weighted average over all possible states. The probability that a randomly generated use it or lose it tree is in a particular state is not uniformly distributed over all the recurrent states, but these can be computed using a Markov chain corresponding to the directed graph model of the recurrent states. We illustrate this with an example.

Consider the simplest non-trivial example of a use it or lose it process - each vertex has lifespan 2 and gives birth to one new vertex. The possible recurrent states of such a tree are $(2, 2)$ and $(1, 2)$ indicating that at any stage there are two vertices with lifespan 2 and either 1 or 2 vertices with lifespan 1. We would like to follow a particular vertex born in some state through it's life's process; we'll indicate the (state, current lifespan) pair as $(\boldsymbol{x}, i)$. Thus the new leaf is born as either $((2, 2), 2)$ or $((1, 2), 2)$. Assuming that this new vertex eventually dies as a leaf, we can follow its path through the appropriate tree shown in figure 4.

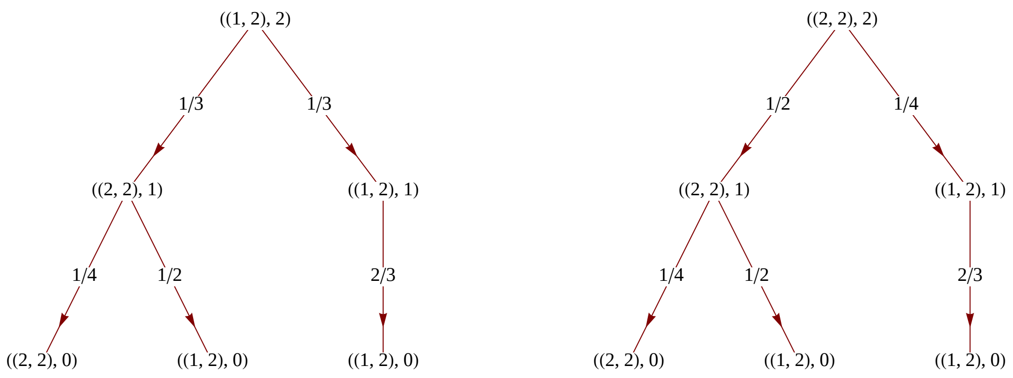

FIGURE 4. Life cycle of a leaf

Now, simply computing conditional probabilities and summing, we see that a vertex born in state $((1, 2), 2)$ has probability $\frac{1}{3} \cdot \frac{1}{4} + \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{2}{3} = \frac{17}{36}$ of dying as a leaf, while a vertex born into state $((2, 2), 2)$ has probability $\frac{1}{2} \cdot \frac{1}{4} + \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{2}{3} = \frac{13}{24}$ of dying as a leaf.

Finally, in order to compute the overall probability that a vertex born into this system dies as a leaf, regardless of the original state, we need to take a weighted average (according to state probability) of these two answers. This weighted average may be computed using a Markov chain corresponding to the digraph model, or

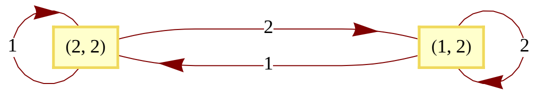

FIGURE 5. The transition diagram

transition diagram, for the two recurrent states shown in figure 5. This is simply the recurrent portion of the directed graph shown in figure 1.

Now, the transition matrix associated with this directed graph has rows and columns indexed by the states. The entry in row u and column v indicates the probability of moving from state u to state v. Basic Markov chain theory tells us that the largest eigenvalue of this matrix is one, that it is a simple eigenvalue, and that the long term probability distribution of the two states as this process evolves is given by the corresponding eigenvector normalized so that its sum is one. In this particular example, the transition matrix and dominant eigenvector are

$$\begin{pmatrix} 1/2 & 1/2 \\ 1/3 & 2/3 \end{pmatrix} \text{ and } \begin{pmatrix} 2/5 \\ 3/5 \end{pmatrix}. \tag{4}$$

This leads to a leaf probability of $\frac{3}{5} \cdot \frac{17}{36} + \frac{2}{5} \cdot \frac{13}{24} = \frac{1}{2}$. We note that the lifespan hasn't changed the leaf probability from the case where $m = 2$ and $n = 1$, in spite of the complexity of the new computation.

While we have no general proof that leaf probability is independent of lifespan in this construction, the recursive construction of $R(n)$ and subsequent computation of leaf probability can easily be programmed. We have done so and verified our leaf probability conjecture for $m \leq 20$ and $n \leq 10$.

4. Directions for future work

This paper raises a number of obvious questions for both random tree construction and decremental tag systems. Foremost is

Conjecture 1. *The probability that a given vertex in a use it or lose it tree dies as a leaf is* $1/2$*, independent of the parameter pair* (m, n)*.*

This implies that that these trees share some structural properties with the classical Erdös-Rényi process, where the leaf probability is also $1/2$. We might also ask other structural questions about use it or lose it trees.

- What is the expected value of the independence number of such a tree?
- What is the expected value of the diameter of such a tree?
- What is the expected degree sequence of such a tree?

We might also consider other random tree constructions.

The decremental tag systems in this paper were developed specifically to model use it or lose it trees, but they seem interesting in their own right. Our main conjecture concerns the number of recurrent states.

Conjecture 2. *Given* n*, let* $m = n$ *and let* D_n *denote the number of recurrent states of in the corresponding decremental tag system. Then* D_n *is the* n^{th} *Catalan number.*

We generated this conjecture by simply plugging the first few D_ns into The On-Line Encyclopedia of Integer Sequences [4].

Computational supplement. All of the algorithms outlined in the this paper have been implemented by the authors using Mathematica. These programs are freely available here:
`http://facstaff.unca.edu/mcmcclur/DecrementalTagTrees/`.

References

[1] E. Post, "Formal Reductions of the General Combinatorial Decision Problem," *American Journal of Mathematics*, **65** (1943) 197–215.

[2] S. Wolfram, *A New Kind of Science* (Wolfram Media, Champaign, IL, 2002).

[3] P. Erdös and A. Rényi, "On the evolution of random graphs," *Magyar Tud. Akad. Mat. Kutato Int. Kozl.*, **5** (1960) 17–61.

[4] Neil Sloan, *The On-Line Encyclopedia of Integer Sequences*, `http://www.research.att.com/~njas/sequences/`.