

A Generator of Rook Polynomials

Daniel C. Fielder

A list adaptation of an inclusion-exclusion method for calculating the rook polynomials of arbitrary finite chessboards is discussed and presented.

■ General

On a chessboard of any configuration, a rook is *nontaking* if the cell it occupies is not in the same row or column as the cell of any other rook. A *rook polynomial* is an ordinary generating function in which the coefficient of x^k ($k = 0, 1, 2, \dots$) is the number of ways k nontaking rooks can be distributed on a finite board. Rook polynomials have many applications in combinatorics, especially in counting restricted permutations. A chessboard can be described mathematically as an ordered set of its cells' unique (row, column) pairs. A chessboard need not be rectangular. The algorithm for finding rook polynomials discussed in this article is an adaptation of a well-known inclusion-exclusion method, typically described by Riordan [1, 168–170] and Liu [2, 111–118].

This method selects any cell of the board as a *special cell*. As Liu observes, the number of ways k nontaking rooks on the original chessboard C is the number of ways with a rook always *included* in the special cell plus the number of ways with all rooks *excluded* from the special cell. The board C_i is found by crossing out the row *and* column of the special cell. The contribution of the rook polynomial from C_i must be multiplied by x to account for the loss of the cell along with its row and column. The board C_e is found by eliminating the special cell completely. Hence, $C = xC_i + C_e$ forms an interesting “equation” in x , with chessboards as coefficients. Continued reductions of C_i and C_e and their successor boards yield nested sets of x 's from which the rook polynomial of board C emerges. When working “by hand,” recognition of the rook polynomials of intermediate boards often shortens this search—a luxury usually not afforded in a computer reduction.

■ A “By Hand” Example

This small example provides a convenient comparison to the computer algorithm adaptation later. Consider the following chessboard of four cells and (arbitrarily)

make $\boxed{1, 1}$ the special cell at the start. The choice of intermediate special cells is obvious. The following reduction leads to the unique rook polynomial, $1 + 4x + 3x^2$.

$$\begin{array}{|c|c|} \hline 1, 1 & \\ \hline 2, 1 & 2, 2 \\ \hline & 3, 2 \\ \hline \end{array} = x \begin{array}{|c|} \hline 2, 2 \\ \hline 3, 2 \\ \hline \end{array} + \begin{array}{|c|c|} \hline 2, 1 & 2, 2 \\ \hline & 3, 2 \\ \hline \end{array} = x(x \boxed{} + \boxed{3, 2}) + \left(x \boxed{3, 2} + \begin{array}{|c|} \hline 2, 2 \\ \hline 3, 2 \\ \hline \end{array} \right) =$$

$$x^2 + 2x \boxed{3, 2} + (x \boxed{} + \boxed{3, 2}) = x^2 + x + (2x + 1) \boxed{3, 2} =$$

$$x^2 + x + (2x + 1)(x + 1) = x^2 + x + 2x + 2x^2 + 1 + x = 1 + 4x + 3x^2$$

The steps taken use algebraic combination and the knowledge that the rook polynomials of the empty chessboard $\boxed{}$ and a single cell (in this case $\boxed{3, 2}$) are 1 and $1 + x$.

■ A List Example

In this example, the computer algorithm for finding rook polynomials is an orderly list adaptation of the “by hand” example. First, initiate a horizontal “stack” with the list of the chessboard’s (row, column) cells on the initial top of the stack. (In the implementation the top will be the first element of the list `stacklist`.) From data in the top of the stack, generate a list of cells of the first *exclusive* type board and append it to the bottom of the stack. (In the implementation the bottom will be put farthest to the right). Next, compare the (row, column) pattern information of the first cell of the top with the patterns of the remaining top cells to obtain the cell information for the first *inclusive* type list. The x multiplier has the same functional weight as a (row, column) cell in the inclusive list and is added on the right to complete the inclusive list. The inclusive list is appended to the right of the exclusive list and is now at the bottom of the stack.

Here is the result of applying the process to the first top, which is the first element in this list.

$$\{\{1, 1\}, \{2, 1\}, \{2, 2\}, \{3, 2\}\}, \{\{2, 1\}, \{2, 2\}, \{3, 2\}\}, \{\{2, 2\}, \{3, 2\}, \{x\}\}$$

In the following table this will look like a stack.

$$\begin{array}{c} \{\{1, 1\}, \{2, 1\}, \{2, 2\}, \{3, 2\}\} \\ \{\{2, 1\}, \{2, 2\}, \{3, 2\}\} \\ \{\{2, 2\}, \{3, 2\}, \{x\}\} \end{array}$$

The top of the stack has served its purpose and is discarded. The lists can be envisioned as propagating up with each removal of a used top (or to the left in the list implementation). The general philosophy is to repeat the generation and removal of new tops until the stack consists only of x entries and a single 1. The single 1 is derived from the *exclusive* reduction of a single cell (see [2, 113] and the earlier use of $\boxed{3, 2} = 1 + x$).

The following tabulation shows the progress of the algorithm through the 17 successive stack values.

1	$\{(1, 1), (2, 1), (2, 2), (3, 2)\}$	2	$\{(1, 1), (2, 1), (2, 2), (3, 2)\}$ $\{(2, 1), (2, 2), (3, 2)\}$ $\{(2, 2), (3, 2), (x)\}$	3	$\{(2, 1), (2, 2), (3, 2)\}$ $\{(2, 2), (3, 2), (x)\}$
4	$\{(2, 1), (2, 2), (3, 2)\}$ $\{(2, 2), (3, 2), (x)\}$ $\{(2, 2), (3, 2)\}$ $\{(3, 2), (x)\}$	5	$\{(2, 2), (3, 2), (x)\}$ $\{(2, 2), (3, 2)\}$ $\{(3, 2), (x)\}$	6	$\{(2, 2), (3, 2), (x)\}$ $\{(2, 2), (3, 2)\}$ $\{(3, 2), (x)\}$ $\{(3, 2), (x)\}$ $\{(x), (x)\}$
7	$\{(2, 2), (3, 2)\}$ $\{(3, 2), (x)\}$ $\{(3, 2), (x)\}$ $\{(x), (x)\}$	8	$\{(2, 2), (3, 2)\}$ $\{(3, 2), (x)\}$ $\{(3, 2), (x)\}$ $\{(x), (x)\}$ $\{(3, 2)\}$ $\{(x)\}$	9	$\{(3, 2), (x)\}$ $\{(3, 2), (x)\}$ $\{(x), (x)\}$ $\{(3, 2)\}$ $\{(x)\}$
10	$\{(3, 2), (x)\}$ $\{(3, 2), (x)\}$ $\{(x), (x)\}$ $\{(3, 2)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$	11	$\{(3, 2), (x)\}$ $\{(x), (x)\}$ $\{(3, 2)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$	12	$\{(3, 2), (x)\}$ $\{(x), (x)\}$ $\{(3, 2)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x)\}$ $\{(x), (x)\}$
13	$\{(x), (x)\}$ $\{(3, 2)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x)\}$ $\{(x), (x)\}$	14	$\{(3, 2)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x), (x)\}$	15	$\{(3, 2)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x), (x)\}$ $\{(1)\}$ $\{(x)\}$
16	$\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x), (x)\}$ $\{(1)\}$ $\{(x)\}$	17	$\{(1)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x)\}$ $\{(x), (x)\}$ $\{(x), (x)\}$		

The stacks 1, 3, 5, 7, 9, 11, 14, and 17 are stable stack configurations, while the others are stacks in transition. In particular, when top lists consist solely of x 's (as in stacks 13 and 16), those lists are transferred to the bottom of the stack before normal stack operation resumes. The algebraic interpretation of the final stack is to multiply along the rows and add up the column, ignoring the list structure. $\{\{1\}\}$ is interpreted as 1. Under these conditions, the program will convert stack 17 to the rook polynomial, $1 + 4x + 3x^2$.

■ Code and Program Details

Here is the program for converting a list of nonrepeating (row, column) chessboard cells to a rook polynomial and its application to our small example.

```
In[2]:= RookPolynomial[n_List] :=
Module[{stacklist, top, SameColumnValues, SameRowValues,
  EntriesToEliminate, InclusionList, ExclusionList},
  stacklist = {Union[n]};
  While[Union[Flatten[stacklist]] != {1, x}, top = First[stacklist];
    SameColumnValues = Position[top, {_, top[[1, 2]]}];
    SameRowValues = Position[top, {top[[1, 1]], _}];
    EntriesToEliminate = Union[SameColumnValues, SameRowValues];
    InclusionList = Append[Delete[top, EntriesToEliminate], {x}];
    ExclusionList = Delete[top, 1];
    If[Length[ExclusionList] == 0, ExclusionList = {{1}}];
    stacklist = Delete[
      Append[Append[stacklist, ExclusionList], InclusionList], 1];
    While[Union[Flatten[First[stacklist]]] == {x},
      stacklist = RotateLeft[stacklist]];
    Total[Apply[Times, stacklist, 2]]]
```

```
In[3]:= RookPolynomial[{{2, 2}, {3, 2}, {3, 3}, {4, 3}}]
```

```
Out[3]= 1 + 4 x + 3 x2
```

When `stacklist` reaches $\{\{1\}, \{x\}, \dots, \{x\}\}$ under `While` loop control, `stacklist` has reached its correct, final value. This is the only value that yields the rook polynomial directly. Also, `Union[Flatten[stacklist]]` equals $\{1, x\}$ only for the final `stacklist` value.

`SameColumnValues` uses the pattern properties of the `Position` command to obtain a list of positions from the top of the stack that have the same *column* values as the leading (row, column) entry at the top of the stack list. `SameRowValues` does the same for the *row* values. Their `Union` gives the list of (row, column) `EntriesToEliminate` in forming the next *inclusion* chessboard, which is completed by appending x . The *exclusion* chessboard list is completed by deleting the first term from the top of the stack. After that, the inclusion and exclusion chessboard strings are appended to the bottom of the stack before the old top of the stack is discarded.

The lists move up (actually over to the left) and the next list becomes the new top of the stack. If not prevented, the program will try to process any stack tops that have leading x 's. Accordingly, the top of a stack list consisting entirely of x 's is moved to the bottom of the stack in the last `While` loop. When `stacklist` eventually meets the condition for shutting down the main `While` loop, the `Apply` line converts `stacklist` to the rook polynomial.

In this manner, `RookPolynomial[{{row,column},...,{row,column}}]` can be determined for any connected chessboard.

■ References

- [1] J. Riordan, *An Introduction to Combinatorial Analysis*, New York: Wiley, 1958.
- [2] C. Liu, *Introduction to Combinatorial Mathematics*, New York: McGraw-Hill, 1968.

About the Author

Daniel C. Fielder received a Ph.D. in electrical engineering from Georgia Institute of Technology in 1957. Except for a year of teaching at Syracuse University and a summer visiting professorship at Westinghouse Research Labs, he taught continuously and performed research on discrete math applications at Georgia Tech since 1948. He was a Life Fellow of IEEE. Dr. Fielder retired in 1988 as professor emeritus, but continued to teach until the time of his death in October 2003. During WWII, he designed ship degaussing systems at the Bureau of Ships under (then) Lt. Cmdr. Hyman G. Rickover, who later founded the U.S. nuclear navy.

Daniel C. Fielder

*School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250*