`rabbits` adapts our backtrack paradigm to this problem. Given the number *n* of rabbits and the depth of search, it exhaustively looks for a solution. The coding that we follow is the same as the one adopted to describe the empty space in the previous section as depicted in Figure 2

```
In[61]:= rabbits[n_Integer, depth_] :=
  Module[{goal, back, move, getCandidates, ans = {}, lim = depth},

   back[x_] := 1 /; Length[x] > lim;
   back[x : {___, {g_, _}}] :=
    (AppendTo[ans, x]; lim = Min[lim, Length[x]]) /; g == goal;
   back[feasible : {{{_String ..}, _Integer} ..}] :=
    Module[{candidates},
     candidates = Select[getCandidates[Last[feasible]],
       Not[MemberQ[First /@ feasible, First[#]]] &];
     Map[back[Join[feasible, {#}]] &, candidates]];

   move := {{x___, a_, b_, "0", y___} → {{x, "0", b, a, y}, 1},
     {x___, "0", a_, b_, y___} → {{x, b, a, "0", y}, 2},
     {x___, a_, "0", y___} → {{x, "0", a, y}, 3},
     {x___, "0", a_, y___} → {{x, a, "0", y}, 4}};

   getCandidates[state : {{_String ..}, _Integer}] :=
    ReplaceList[First[state], move];

   goal = Join[{"0"}, ToString /@ Range[n, 1, -1]];
   back[{{ToString /@ Range[0, n], 0}}];
   If[ans == {}, {}, Last[ans]]]
```

For example, if there are two rabbits we obtain:

```
In[62]:= u = rabbits[2, 10];
         Map[{StringJoin[First[#]], Last[#]} &, u]

Out[63]= {{012, 0}, {102, 4}, {120, 4}, {021, 1}}
```

With three rabbits, the solution sequence grows as:

```
In[64]:= u = rabbits[3, 10];
         Map[{StringJoin[First[#]], Last[#]} &, u]

Out[65]= {{0123, 0}, {2103, 2}, {2130, 4}, {2031, 1}, {2301, 4}, {0321, 1}}
```

Compiling a table of solutions for the cases of up to five rabbits, we obtain:

```
In[66]:= Table[u = rabbits[n, 18];
           {n, FromDigits[Last /@ u]}, {n, 2, 5}] // MatrixForm
```

Out[66]//MatrixForm=

$$\begin{pmatrix} 2 & 441 \\ 3 & 24\,141 \\ 4 & 4\,241\,142\,411 \\ 5 & 2\,441\,322\,311\,422\,311 \end{pmatrix}$$

To get some insight into the problem, consider the following results.

Definition 3. *Let X_n be the word defined as: $X_1 = 4$, $X_2 = 2$ and*

$$X_n = \begin{cases} 2^{n/2}\,3\,(14)^{n-2}\,2^{(n/2)-1}\,4\,(14)^{n-2}\,X_{n-2} & n \text{ even} \\ 2^{(n-1)/2}\,4\,(14)^{n-1}\,2^{(n-3)/2}\,4\,(14)^{n-3}\,X_{n-2} & n \text{ odd} \end{cases}.$$

Theorem 8. *$0123\ldots n \xrightarrow{X_n\,3^n} 0\,n(n-1)\,(n-2)\ldots 21.$*

Proof. For n even:

$$\begin{aligned} 012\ldots n &\xrightarrow{2^{n/2}} 214\,365\ldots n\,(n-1)\,0 \\ &\xrightarrow{3} 214\,365\ldots n\,0\,(n-1) \\ &\xrightarrow{(14)^{n-2}} n\,0\,214\,365\ldots (n-2)\,(n-3)\,(n-1) \\ &\xrightarrow{2^{(n/2)-1}} n\,123\ldots (n-2)\,0\,(n-1) \\ &\xrightarrow{4} n\,123\ldots (n-2)\,(n-1)\,0 \\ &\xrightarrow{(14)^{n-2}} n\,(n-1)\,0123\ldots (n-2) \end{aligned}$$

the case n odd is handled similarly. □

Although useful, this result gives a large upper bound for the solution. For instance, instead of the 21 movements required to get from 0123456 to 0654321, we need 43 by using Theorem 8. Just to appreciate the subtleties involved, let us consider the steps used in applying one of the shortest solutions 2223113 ∵ 22231132223113:

$$\begin{aligned} &0\,123\,456 \xrightarrow{222} 2\,143\,650 \xrightarrow{311} 2\,041\,635 \xrightarrow{322} 4\,261\,035 \xrightarrow{231} \\ &4\,260\,513 \xrightarrow{132} 6\,402\,513 \xrightarrow{223} 6\,452\,301 \xrightarrow{113} 0\,654\,321. \end{aligned}$$

If instead of describing the movement of the 0 marker, we explicitly mention the label of the rabbit that has to move/jump, we have the following result.

Theorem 9. *Let e be the increasing sequence of even labels 0246 … and o the decreasing sequence of odd labels … 531 of the n rabbits involved in the rabbits problem.*

Then the following holds:

$$0123 \ldots n \xrightarrow{(e\,o)^{n+1}} 0123 \ldots n$$
$$0123 \ldots n \xrightarrow{(e\,o)^{n/2}\,e} 0\,n\,(n-1) \ldots 321 \quad n \text{ even}$$
$$0123 \ldots n \xrightarrow{(e\,o)^{(n+1)/2}\,e} n(n-1) \ldots 210 \quad n \text{ odd.}$$

This provides sequence $(246\,531)^3$ 246 as a 21-step solution for the case $n = 6$. In general, it provides a sequence significantly smaller than the one given by Theorem 8.

Once the training of our rabbits is over, they can be put in a circular board connecting the last square to the first one to study their reactions to a more difficult setup. This new arrangement is reflected in the following list of transformations that gives rise to the following function `Crabbits` (circular rabbits).

In[67]:=
```
moves := {{"0", x___, a_, b_} → {{"0", b, a, x}, 1},
   {"0", a_, b_, x___} → {{"0", x, b, a}, 2},
   {"0", x___, a_} → {{"0", a, x}, 3},
   {"0", a_, x___} → {{"0", x, a}, 4}};
```

In[68]:=
```
Crabbits[n_Integer, depth_] :=
 Module[{goal, back, moves, getCandidates, ans = {}, lim = depth},

  back[x_] := 1 /; Length[x] > lim;
  back[x : {___, {g_, _}}] :=
   (AppendTo[ans, x]; lim = Min[lim, Length[x]]) /; g == goal;
  back[feasible : {{{_String ..}, _Integer} ..}] :=
   Module[{candidates},
    candidates = Select[getCandidates[Last[feasible]],
      Not[MemberQ[First /@ feasible, First[#]]] &];
    Map[back[Join[feasible, {#}]] &, candidates]];

  moves := {{"0", x___, a_, b_} → {{"0", b, a, x}, 1},
    {"0", a_, b_, x___} → {{"0", x, b, a}, 2},
    {"0", x___, a_} → {{"0", a, x}, 3},
    {"0", a_, x___} → {{"0", x, a}, 4}};

  getCandidates[state : {{_String ..}, _Integer}] :=
   ReplaceList[First[state], moves];

  goal = Join[{"0"}, ToString /@ Range[n, 1, -1]];
  back[{{ToString /@ Range[0, n], 0}}];
  If[ans == {}, {}, Last[ans]]]
```

The results for up to six rabbits are:

In[69]:=
```
Table[u = Crabbits[n, 15];
  {n, FromDigits[Last /@ u]}, {n, 2, 6}] // MatrixForm
```

Out[69]//MatrixForm=

2	4
3	2
4	4422
5	422 232
6	44 422 322 322

Let us note that the symbol "0" in the previous transformation `move` is not really necessary. In looking at the resulting 0-less transformations for the first time, who would realize that these strange transformations correspond to movements performed by rabbits in a circular one-dimensional board!

■ Acknowledgments

This work was completed while the author was a visiting scholar at Wolfram Research, Inc., whose assistance and enthusiastic support is gratefully acknowledged. I would also like to thank the University of Queretaro in Mexico for their continued support.

■ References

[1] D. E. Knuth, "Estimating the Efficiency of Backtrack Programs" in *Selected Papers on the Analysis of Algorithms*, Stanford, CA: CSLI Publications, 2000, pp. 55-75.

[2] S. S. Skiena, *The Algorithm Design Manual*, New York: Springer-Verlag, 1997.

[3] S. S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory in Mathematica*, Reading, MA: Addison-Wesley, 1990.

[4] H. S. M. Coxeter and S. L. Greitzer, *Geometry Revisited*, Washington, DC: The Mathematical Association of America, 1967.

[5] T. H. O'Beirne, "Jug and Bottle Department," in *Puzzles and Paradoxes*, New York: Oxford University Press, 1965 pp. 49-75.

[6] P. Abbott, ed., "Tricks of the Trade," *The Mathematica Journal*, **6**(4), Fall 1996, pp. 17-26.

[7] E. R. Berlekamp, J. H. Conway, and R. Guy, *Winning Ways: For Your Mathematical Plays*, Vol. 2, London: Academic Press, 1982.

About the Author

Jaime Rangel-Mondragón received M.Sc. and Ph.D. degrees in applied mathematics and computation from the University College of North Wales in Bangor, UK. He has been a visiting scholar at Wolfram Research, Inc. and held research positions in the Faculty of Informatics at UCNW, the College of Mexico, the Center of Research and Advanced Studies, the Monterrey Institute of Technology, the Queretaro Institute of Technology, and the University of Queretaro in Mexico, where he is presently a member of the Faculty of Informatics. A prolific contributor to *MathSource*, he currently writes the column *Geometric Themes* for the *Mathematica in Education and Research* journal. His research interests include recreational mathematics, combinatorics, the theory of computing, computational geometry, and functional languages.

Jaime Rangel-Mondragón
Universidad Autónoma de Querétaro
Mexico
jrangelmondragon@gmail.com