The SPIN-OUT Puzzle

Kirk Pruhs
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260
kirk@cs.pitt.edu

1 Introduction

Puzzles have proven to be good tools for teaching principles of computer science. In particular, the towers of Hanoi puzzle (towers for short), and it many variants (e.g. [2, 3]), are good tools for teaching recursion. Our purpose is to bring to the reader's attention another puzzle, SPIN-OUT, that the author has found to be a good tool for teaching recursion as a problem solving tool. At first encounter students find that developing a recursive solution for SPIN-OUT seems significantly harder than solving the towers puzzle, as it requires strengthening the inductive hypothesis, strong induction, and solving a more difficult recurrence relation. However, in the end they find that SPIN-OUT is merely the towers puzzle incognito. The SPIN-OUT puzzle is essentially isomorphic to the Chinese rings puzzle. See, for example, [1] for a discussion of the Chinese rings puzzle. However, the SPIN-OUT puzzle has several advantages as a teaching tool. It is easier to manipulate by hand, it is easier to see how to model the puzzle mathematically, and the author's experience indicates that, unlike the inexpensive models of the Chinese ring puzzles, it will not break from overly exuberant student handling. We develop the recursive solution for SPIN-OUT here, even though it could be deduced from the solution for the Chinese rings puzzle by appropriate substitution of terms (eg. spinner for ring), because the Chinese rings puzzle seems to be not all that well known, and it is nice to have a solution using terminology appropriate for the puzzle.

Before describing SPIN-OUT some words of credit are in order. SPIN-OUT is copyrighted and patented by Binary Arts Corporation. They credit its invention to William Kiester, circa 1930. More information can be found in a small pamphlet that comes with the puzzle. Our informal survey indicates that SPIN-OUT can be found in most game stores. Otherwise, one can write to Binary Arts directly at: 703 Timber Branch Drive; Alexandria, VA 22302.

2 Puzzle Description

The SPIN-OUT puzzle consists of 7 spinners, the darkened shapes in figure 1, mounted on a sliding bar encased in plastic housing. The housing is surrounded by the darker lines in the figure. We number the spinners from right to left. So, 1 is rightmost spinner and 7 is the leftmost. We think of spinners as pointing a particular direction. For example, in the figure spinner 1 is pointing left, and spinner 2 is pointing up. The goal of the puzzle is to slide the bar out from the rest of the puzzle. Because of the shape of the housing all the spinners must be in horizontal position, as are spinners 1 and 4 in the picture, to allow this goal to be accomplished.

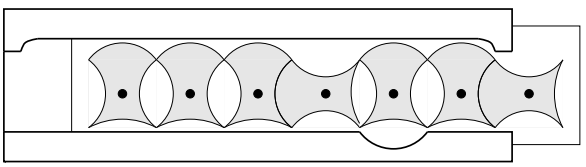

Figure 1: SPIN-OUT puzzle

The shape of the housing and the spinners also impose the following constraints on how the spinners are rotated. Only the spinner above the divot in the bottom housing may be rotated. This is where spinner 3 is in the picture. A horizontal spinner locks a neighboring spinner. For example in the figure spinner 1 prevents spinner 2 from rotating. We shall see that in the optimal solution there is no need to ever point a spinner down or to the right. There are two general ways to rotate a spinner:

Rule 1: If not locked by spinner 2, spinner 1 may be moved over the divot and rotated by sliding the bar all the way to the left.

Rule 2: The spinner immediately to the left of the rightmost vertical spinner may be moved over the

divot and rotated, provided it is not locked by the spinner to its left. In the figure spinner 3 can be rotated under this rule.

We adopt the following notation to denote a configuration of the spinners. We represent a horizontal spinner by 0 and a vertical spinner by 1. We represent a sequence of i horizontal (vertical) spinners by 0^i (1^i). Hence the configuration in the figure can be represented by $1^3 0 1^2 0$.

When SPIN-OUT comes out of the box its in the configuration 1^7. We thus generalize this problem to n spinners and consider the problem reaching configuration 0^n from configuration 1^n.

3 Recursive Solution

Use $A(n)$ to denote the minimum number of spinner rotations required to reach configurations 0^n from 1^n. Instead of giving the final strengthened inductive hypothesis we trace the development of a recursive solution. Note that if we have a solution for $k < n$ this can be used to make the rightmost k spinners horizontal. Consider the time before spinner n is rotated. Note that if $n > 1$ spinner n must be rotated using rule 2. In order for spinner n to be rotated using rule 2 the configuration must be 110^{n-2}. This involves recursively solving our original problem for the rightmost $n-2$ spinners. From there we can rotate spinner n to point left and are left with the problem of reaching 0^n from 010^{n-2}.

Let us now detour from our original line of inquiry to consider the problem of moving from 10^{k-1} to 0^k. Let $C(k)$ be the minimum number of rotations required to solve this problem. Consider the time just before spinner k is first rotated. The configuration at that time must be 010^{k-2}. Thus, as a subproblem we need to consider moving from 0^{k-1} to 10^{k-2}. Note that we can assume by induction that we know how to accomplish the reverse, that is move from 10^{k-2} to 0^{k-1}. Well, let us just strengthen our inductive hypothesis to assume that we also know how to move from 0^{k-1} to 10^{k-2}. A moments reflection shows that this is no great generalization since any sequence α of moves that when applied to configuration A yield configuration B can be reversed to give a sequence for reaching A from B. Hence, we know that $C(k-1)$ rotations are sufficient to reach configuration 10^{k-2}. Rotating spinner k to point left brings us to the configuration 010^{k-2}. By induction one can then reach configuration 0^k using $C(k-1)$ spinner rotations. This gives a recurrence relation for $C(k)$ of $C(k) = 2C(k-1)+1$. Since $C(1) = 1$, the solution for this recurrence relation is $C(k) = 2^k - 1$. Hence, the final inductive hypothesis for this problem is:

Induction Hypothesis: For all k, $C(k) = 2^k - 1$ rotations are necessary and sufficient to reach 10^{k-1}, from 0^k, and $C(k)$ rotations are necessary and sufficient to reach 0^k from 10^{k-1}.

At this point the towers puzzle's cover has been blown. Starting from configuration 10^{k-1}, the ith move of the recursive algorithm for SPIN-OUT rotates spinner s if and only if the standard recursive algorithm for towers moves the sth smallest ring on move i.

Now let us return to our original problem of moving from 1^n to 0^n. When we left the configuration of the puzzle was 010^{n-2}. We now know that it takes $2^{n-1}-1$ rotations to reach 0^n from this configuration. Hence the recurrence relation for $A(n)$ is $A(n) = A(n-2)+2^{n-1}$. By inspection one can determine that $A(1) = 1$ and $A(2) = 2$. One can either solve this using standard techniques, or verify by induction that the solution is $(2^{n+2}-3)/6 + (-1)^n$. So we get as our final inductive hypothesis:

Induction Hypothesis: For all n, $A(n) = (2^{n+2} - 3)/6 + (-1)^n$ moves are necessary and sufficient to reach 0^n from 1^n.

We then arrive at the following recursive code to solve the SPIN-OUT puzzle. Here "rotate(i)" means to rotate spinner number i.

```
Procedure A(n:Integer)
Comment: takes puzzle from 1^n to 010^{n-2}
if n = 1 then "rotate(1)"
else if n = 2 then "rotate(2) rotate(1)"
else begin
    A(n − 2)
    "rotate(n)"
    C(n − 1)
    end
end
```

```
Procedure C(k:Integer)
Comment: takes puzzle from 10^{k-1} to 0^k
if k = 1 then "rotate(1)"
else begin
    C_Reverse(k − 1)
    "rotate(k)"
    C(k − 1)
    end
end
```

```
Procedure C_Reverse(k:Integer)
Comment: takes puzzle from 0^k to 10^{k-1}
if k = 1 then "rotate(1)"
else begin
    C_Reverse(k − 1)
    "rotate(k)"
    C(k − 1)
    end
end
```

4 Odds and Ends

Once again as in the Towers of Hanoi case there is an iterative algorithm, that produces the same sequence of rotations as the recursive algorithm, but is easier for humans to execute.

Iterative Algorithm: If n is odd (even) then on the odd (even) numbered move rotate spinner 1, and on even (odd) numbered moves make the only other legal move. One reason the iterative algorithm is easy for humans to execute is because of the ease in which the spinners can be put in correct position. If the right part of the housing is raised the bar slides all the way to the left putting spinner 1 over the divot, and if the left part of the housing is raised the bar slides to the right until the first vertical spinner catches in the housing, which is the desired position to use rule 2. Hence, one need only remember, "left, right, left, right, ..." Anyone with a military background should have no problem with this.

While it is perhaps disappointing that there is an iterative bypass, it allows one to perform the demonstration. The natural inclination of a student, after hearing the recursive solution for the first time, is to think that they could now solve the puzzle. However, if asked to actually physically solve SPIN-OUT, almost all students will be unable to emulate the recursive algorithm due to insufficient short term memory to keep track of the recursive call stack. In contrast, it is easy for a student to emulate the iterative algorithm after a quick explanation. Field experiments suggest that the optimal time to solve the puzzle, for $n = 7$, is approximately 60 seconds.

It seems to be good practice to always leave some problems unsolved for the reader. The first is to generalize the recursive algorithm to find an algorithm for reaching one arbitrary configuration from another using the fewest number of rotations. As a second problem, consider the following variant of SPIN-OUT. The divot is moved k spinners widths to the left of where it is currently located. Thus, spinners 1 to $k + 1$ may be moved over the divot at any time and, if they are not locked by an adjacent horizontal spinner, may be then rotated. The k spinners to the left of the rightmost horizontal spinner may be moved over the divot and rotated, provided they are not locked by adjacent horizontal spinners. We assume that the bar can be slid arbitrarily far to the left, but may not be removed on the left. The problem is to find the optimal recursive algorithm for going from configuration 1^n to configuration 0^n. We end with a question that the author does not know the answer to, "Is there a nice optimal iterative algorithm for this generalized problem?"

References

[1] A. Dewdney, "Ying and yang: recursion and iteration, the Tower of Hanoi and the Chinese rings", *Scientific American*, bf 251, 19 – 28, November 1984.

[2] S. Minsker, The towers of Antwerpen problem, *Information Processing Letters*, **38**, 107–111, 1991.

[3] J. Wu, and R. Chen, "The towers of Hanoi problem with parallel moves", *Information Processing Letters*, **44**, 241–243, 1992.