FEATURE COLUMN Monthly essays on mathematical topics

(/samplin

Puzzling Over Exact Cover Problems

This article will describe how to represent Kanoodle as an "exact cover problem" and how Donald Knuth implemented an algorithm to find solutions to exact cover pro

Mail to a friend (mailto: Enter_Colleague's_Address?
subject=Feature%20Column%20from%20the%20AMS&body=Dear%20Colleague,%20%3Cbr%20/%3E%3Cbr%20/%3EI%20found%20this%20%3Cem%3E%3Cstrong%3EFeat
column/fcarc-kanoodle)

Introduction

For a birthday gift, one of my children recently received a puzzle called Kanoodle

that provides 12 distinctly shaped pieces:

and asks the player to assemble them into a 5 by 11 rectangular grid.

Without assistance, the puzzle is extremely difficult so a booklet includes many starting configurations from which to begin.

After solving the puzzle a few times, I wondered if there was an algorithmic way for finding solutions and soon discovered that there was indeed a very interest

First, we will describe a generic version of the exact cover problem. Suppose we have a set S and a collection of subsets A_1, A_2, ..., A_n. We ask to find a subcoll

Imagine we have the set $S = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}\}$ and subsets

Notice that the subsets $A_3 = \{\mathbf{2}, \mathbf{3}\}$ and $A_5 = \{\mathbf{1}, \mathbf{4}, \mathbf{5}\}$ form a solution to the exact cover problem: each element of S appears in exactly one of the sets.

In the general situation, if we have a subcollection of the sets A_1, A_2, ..., A_n, it is relatively easy to determine whether it is a solution. However, it is much more elements and a collection of 2222 of its subsets. The number of subcollections of the sets is the impossibly large number

Notice that, in this naive approach, the amount of work required grows exponentially with the number of subsets n. In fact, there is no known algorithm for finc collection of problems of great interest in the theory of computation.

While there is no known algorithm that is guaranteed to find solutions to the general problem in a reasonable amount of time, we will describe an algorithm to f

Finding solutions to the exact cover problem

Let's now look at an algorithm, called "Algorithm X" by Knuth, for finding solutions to the exact cover problem. It's not particularly sophisticated; indeed Knut

Let's consider our example again, with slightly different notation. Imagine our set S is the set $S = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}\}$ and that we have subsets

We will visualize this situation using a matrix, a rectangular array in which each column corresponds to an element of S and each row is one of our subsets. A c

Let's consider one of the elements of *S*, say, the element **1**. It is contained in two subsets, *A* and *E*. Therefore, any solutior

Here is the basic step in the algorithm. Let's begin by looking for solutions containing the subset *A*. Since *A* contains the e

So we travel down the columns corresponding to elements **1** and **5**

and eliminate any row corresponding to a subset containing **1** or **5**. In this case, we eliminate subset *E*.

After removing these rows and columns, we are left with this simpler configuration for which we seek a solution to the ex

If we ever come to a matrix with no columns, then every element of *S* has been placed into a unique subset. In other word

Remember that we began by attempting to place the element **1** in a subset and we initially chose subset *A*. We also need to consider, in the same way, placing **1**
Each edge is labeled by the subset chosen to be part of the solution we seek.

Algorithm X, written more carefully, looks like:

We have some choice in how we implement the third step in which we choose a column *c*. In the example above, we have simply chosen the leftmost column a

While the algorithm, as written above, shows that we need to remove rows and columns from our matrix, we will also need to reinsert them later. For instance, configuration, which requires us to reinsert the rows and columns we initially removed.

All told, we need to be able to perform a rather small set of operations. We need to be able to list all the entries in a row or column, and we need to be able to re

Implementing Algorithm X with Dancing Links

To implement Algorithm X, we will need to be able to remove rows and columns from our matrix, and we will need to be able to put them back in. In addition,

We could store this information, in the usual way, as a matrix containing 0's and 1's to indicate an element's membership in a particular set. However, Knuth de

We'll begin by describing a *doubly-linked list,* which is an ordered list of items together with a link from each item to its predecessor and successor. Notice that

For each element *a* in the list, *L*[*a*] will denote it predecessor, the element to the left of *a*, and *R*[*a*] will denote its successor to the right.

Notice that it is easy to remove an element from the list. For instance, if we want to remove element *a*, we simply reroute the links from *a*'s predecessor and suc

We do this by setting:

Notice that there's no need to change the links *out* of *a*. In fact, this makes it very simple to reinsert *a*. Simply set

Now to implement Algorithm X, we store the columns of the matrix in a doubly-linked list. Remember that individual columns will disappear and reappear as v

Each column will itself form a doubly-linked list consisting of all the entries in the rows of the matrix. Furthermore, each row will be represented in a doubly-li

This is an ideal setup for implementing Algorithm X. Each element of the matrix is contained in two doubly-linked lists: one for the row and one for the colum

As the algorithm progresses, the links are continually being modified in a way that led Knuth to call this implementation "Dancing Links."

Viewing Kanoodle as an exact cover problem

We can apply Algorithm X to find solutions to the Kanoodle puzzle once we understand how to view the puzzle as an exact cover problem. Remember that we and an empty grid of 5 rows and 11 columns.

A solution is found when:

A solution will be determined by a placement of each piece on the board. We will therefore have one row in our matrix for each possible placement of each of t

There will be 67 columns: one for each of the Kanoodle pieces and one for each cell in the 5 by 11 grid. A particular placement of a particular piece has an entr

For instance, this placement gives a row with five entries in the columns labeled by **A**, **[4, 3]**, **[5,3]**, **[5,4]**, and **[5,5]**. (The cell [*r*, *c*] is in row *r* and column *c*.)

The search tree for Kanoodle is huge and there are many solutions. To help the algorithm run faster, I used the symmetry of the board to restrict the possible pla
of the possible placements of **L** separately.

I ran my version of Dancing Links for several hours in a few of these cases and generated many thousands of solutions, a few of which are shown below. Howe

Since I had satisfied my initial urge to find an algorithmic means for generating solutions, and since I was interested in studying the search tree that Dancing Li

Sudoku is also an exact cover problem

Many readers will be familiar with Sudoku puzzles, like the one shown here. There is a nine by nine grid of cells with nine boxes consisting of three by three g

- Every cell contains exactly one of the symbols.
- Every column has exactly one occurrence of each symbol.
- Every row has exactly one occurrence of each symbol.
- Every three by three box has exactly one occurrence of each symbol.

For instance, the solution (there is only one) to the puzzle above is

With the rules explained as above, we will set up an exact cover problem in which a row of our matrix corresponds to a possible way of filling a cell with a sym

- The first 81 are labeled by the 81 different cells and will be used to indicate that a symbol has been placed in a particular cell.
- The next 81 are labeled by the nine rows and nine possible symbols. An entry in the column **R** = 3, **S** = 7 means that the third row contains the symbol 7.
- The next 81 are labeled by the nine columns and nine possible symbols.
- The final 81 are labeled by the nine boxes and nine possible symbols.

For instance, placing the symbol 5 in the first row and first column of the puzzle leads to a row in the matrix with four entries in the columns correspondi

The matrix is initially set up with one row for each cell whose symbol is given and nine rows for each of the other cells corresponding to the nine different sym

The complexity of the search tree that results from applying Algorithm X reflects how difficult the puzzle is. For instance, the puzzle above is labeled as "Easy
with a single entry.

By contrast, the following puzzle is labeled as "Evil."

Algorithm X completes the middle row before coming to a branching point: the cell shaded red may be filled in with either a 4 or 5.

If we put a 5 in that cell, the entries in 19 more cells are determined before we come to a dead end (it is impossible to place the symbol 9 in the seventh row):

However, putting a 4 in the red cell leads to a solution with no more branching:

The search tree for this puzzle looks like:

Evil, of course, wears many faces. The following puzzle has a search tree that is shown below and drawn without intermediate nodes to emphasize the branchin

References

- Donald E. Knuth, *Dancing Links,* available as paper P159 at http://www-cs-faculty.stanford.edu/~knuth/preprints.html (http://www-cs-faculty.stanford.edu/~knut
  Knuth's paper applies Dancing Links to the classic problem of placing pentominoes on a board with 60 cells, a problem very similar to the Kanoodle puzzle. In a
  Knuth also offers his source code for Dancing Links at http://www-cs-faculty.stanford.edu/~knuth/programs.html. (http://www-cs-faculty.stanford.edu/~knuth/pro
- Solomon W. Golomb, **Polyominoes,** Scribner, 1965.
- Wikipedia pages for exact cover problems (http://en.wikipedia.org/wiki/Exact_cover) and NP-complete problems (http://en.wikipedia.org/wiki/NP-complete).
- J. Glenn Brookshear, **Theory of Computation: Formal Languages, Automata, and Complexity,** Benjamin/Cummings Publishing, 1989.
- I obtained the Sudoku puzzles online from Web Sudoku. (http://www.websudoku.com/)

Those who can access JSTOR can find some of the papers mentioned above there. For those with access, the American Mathematical Society's MathSciNet car

Comments: Email Webmaster (/cgi-bin/fmail/fmail.cgi?emailto=52616e646f6d4956d39de4e6d2aad1a38f8cb94dbc4da0693e459c60ec3a73f018ea44432c107e5