Knuth's Algorithm X

From Wikipedia, the free encyclopedia
(Redirected from Algorithm X)

"Algorithm X" is the name Donald Knuth used in his paper "Dancing Links" to refer to "the most obvious trial-and-error approach" for finding all solutions to the exact cover problem.[1] Technically, Algorithm X is a recursive, nondeterministic, depth-first, backtracking algorithm. While Algorithm X is generally useful as a succinct explanation of how the exact cover problem may be solved, Knuth's intent in presenting it was merely to demonstrate the utility of the dancing links technique via an efficient implementation he called DLX.[1]

The exact cover problem is represented in Algorithm X using a matrix A consisting of 0s and 1s. The goal is to select a subset of the rows so that the digit 1 appears in each column exactly once.

Algorithm X functions as follows:

1. If the matrix A has no columns, the current partial solution is a valid solution; terminate successfully.
2. Otherwise choose a column c (deterministically).
3. Choose a row r such that $A_{r,c} = 1$ (nondeterministically).
4. Include row r in the partial solution.
5. For each column j such that $A_{r,j} = 1$,
   for each row i such that $A_{i,j} = 1$,
   delete row i from matrix A;
   delete column j from matrix A.
6. Repeat this algorithm recursively on the reduced matrix A.

The nondeterministic choice of r means that the algorithm essentially clones itself into independent subalgorithms; each subalgorithm inherits the current matrix A, but reduces it with respect to a different row r. If column c is entirely zero, there are no subalgorithms and the process terminates unsuccessfully.

The subalgorithms form a search tree in a natural way, with the original problem at the root and with level k containing each subalgorithm that corresponds to k chosen rows. Backtracking is the process of traversing the tree in preorder, depth first.

Any systematic rule for choosing column c in this procedure will find all solutions, but some rules work much better than others. To reduce the number of iterations, Knuth suggests that the column choosing algorithm select a column with the lowest number of 1s in it.

Example [edit]

For example, consider the exact cover problem specified by the universe $U = \{1, 2, 3, 4, 5, 6, 7\}$ and the collection of sets $\mathcal{S} = \{A, B, C, D, E, F\}$, where:

- $A = \{1, 4, 7\}$;
- $B = \{1, 4\}$;
- $C = \{4, 5, 7\}$;
- $D = \{3, 5, 6\}$;
- $E = \{2, 3, 6, 7\}$; and
- $F = \{2, 7\}$.

This problem is represented by the matrix:

Wikipedia The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Français
Српски / srpski
Edit links

	1	2	3	4	5	6	7
A	1	0	0	1	0	0	1
B	1	0	0	1	0	0	0
C	0	0	0	1	1	0	1
D	0	0	1	0	1	1	0
E	0	1	1	0	0	1	1
F	0	1	0	0	0	0	1

Algorithm X with Knuth's suggested heuristic for selecting columns solves this problem as follows:

Level 0

Step 1—The matrix is not empty, so the algorithm proceeds.

Step 2—The lowest number of 1s in any column is two. Column 1 is the first column with two 1s and thus is selected (deterministically):

	1	2	3	4	5	6	7
A	1	0	0	1	0	0	1
B	1	0	0	1	0	0	0
C	0	0	0	1	1	0	1
D	0	0	1	0	1	1	0
E	0	1	1	0	0	1	1
F	0	1	0	0	0	0	1

Step 3—Rows *A* and *B* each have a 1 in column 1 and thus are selected (nondeterministically).

The algorithm moves to the first branch at level 1…

Level 1: Select Row *A*

Step 4—Row *A* is included in the partial solution.

Step 5—Row *A* has a 1 in columns 1, 4, and 7:

	1	2	3	4	5	6	7
A	1	0	0	1	0	0	1
B	1	0	0	1	0	0	0
C	0	0	0	1	1	0	1
D	0	0	1	0	1	1	0
E	0	1	1	0	0	1	1
F	0	1	0	0	0	0	1

Column 1 has a 1 in rows *A* and *B*; column 4 has a 1 in rows *A*, *B*, and *C*; and column 7 has a 1 in rows *A*, *C*, *E*, and *F*. Thus rows *A*, *B*, *C*, *E*, and *F* are to be removed and columns 1, 4 and 7 are to be removed:

	1	2	3	4	5	6	7
A	1	0	0	1	0	0	1
B	1	0	0	1	0	0	0
C	0	0	0	1	1	0	1
D	0	0	1	0	1	1	0
E	0	1	1	0	0	1	1
F	0	1	0	0	0	0	1

Row *D* remains and columns 2, 3, 5, and 6 remain:

	2	3	5	6
D	0	1	1	1

Step 1—The matrix is not empty, so the algorithm proceeds.

Step 2—The lowest number of 1s in any column is zero and column 2 is the first column with zero 1s:

	2	3	5	6
D	0	1	1	1

Thus this branch of the algorithm terminates unsuccessfully.

The algorithm moves to the next branch at level 1…

Level 1: Select Row *B*

Step 4—Row *B* is included in the partial solution.

Row *B* has a 1 in columns 1 and 4:

	1	2	3	4	5	6	7
A	1	0	0	1	0	0	1
B	1	0	0	1	0	0	0
C	0	0	0	1	1	0	1
D	0	0	1	0	1	1	0
E	0	1	1	0	0	1	1
F	0	1	0	0	0	0	1

Column 1 has a 1 in rows *A* and *B*; and column 4 has a 1 in rows *A*, *B*, and *C*. Thus rows *A*, *B*, and *C* are to be removed and columns 1 and 4 are to be removed:

	1	2	3	4	5	6	7
A	1	0	0	1	0	0	1
B	1	0	0	1	0	0	0
C	0	0	0	1	1	0	1
D	0	0	1	0	1	1	0
E	0	1	1	0	0	1	1
F	0	1	0	0	0	0	1

Rows *D*, *E*, and *F* remain and columns 2, 3, 5, 6, and 7 remain:

	2	3	5	6	7
D	0	1	1	1	0
E	1	1	0	1	1
F	1	0	0	0	1

Step 1—The matrix is not empty, so the algorithm proceeds.

Step 2—The lowest number of 1s in any column is one. Column 5 is the first column with one 1 and thus is selected (deterministically):

	2	3	5	6	7
D	0	1	1	1	0
E	1	1	0	1	1

F	1	0	0	0	1

Step 3—Row *D* has a 1 in column 5 and thus is selected (nondeterministically).

The algorithm moves to the first branch at level 2…

Level 2: Select Row *D*

Step 4—Row *D* is included in the partial solution.

Step 5—Row *D* has a 1 in columns 3, 5, and 6:

	2	**3**	**5**	**6**	**7**
D	0	**1**	**1**	**1**	0
E	1	1	0	1	1
F	1	0	0	0	1

Column 3 has a 1 in rows *D* and *E*; column 5 has a 1 in row *D*; and column 6 has a 1 in rows *D* and *E*. Thus rows *D* and *E* are to be removed and columns 3, 5, and 6 are to be removed:

	2	**3**	**5**	**6**	**7**
D	0	**1**	**1**	**1**	0
E	1	**1**	0	**1**	1
F	1	0	0	0	1

Row *F* remains and columns 2 and 7 remain:

	2	**7**
F	1	1

Step 1—The matrix is not empty, so the algorithm proceeds.

Step 2—The lowest number of 1s in any column is one. Column 2 is the first column with one 1 and thus is selected (deterministically).

Row *F* has a 1 in column 2 and thus is selected (nondeterministically).

The algorithm moves to the first branch at level 3…

Level 3: Select Row *F*

Step 4—Row *F* is included in the partial solution.

Row *F* has a 1 in columns 2 and 7:

	2	**7**
F	**1**	**1**

Column 2 has a 1 in row *F*; and column 7 has a 1 in row *F*. Thus row *F* is to be removed and columns 2 and 7 are to be removed:

	2	**7**
F	**1**	**1**

Step 1—The matrix is empty, thus this branch of the algorithm terminates successfully.

As rows *B*, *D*, and *F* are selected, the final solution is:

	1	**2**	**3**	**4**	**5**	**6**	**7**
B	1	0	0	1	0	0	0
D	0	0	1	0	1	1	0
F	0	1	0	0	0	0	1

In other words, the subcollection {*B*, *D*, *F*} is an exact cover, since every element is contained in exactly one of the sets *B* = {1, 4}, *D* = {3, 5, 6}, or *F* = {2, 7}.

There are no more selected rows at level 3, thus the algorithm moves to the next branch at level 2…

There are no more selected rows at level 2, thus the algorithm moves to the next branch at level 1…

There are no more selected rows at level 1, thus the algorithm moves to the next branch at level 0…

There are no branches at level 0, thus the algorithm terminates.

In summary, the algorithm determines there is only one exact cover: $\mathcal{S}^* = \{B, D, F\}$.

Implementations [edit]

Donald Knuth's main purpose in describing Algorithm X was to demonstrate the utility of Dancing Links. Knuth showed that Algorithm X can be implemented efficiently on a computer using Dancing Links in a process Knuth calls *"DLX"*. DLX uses the matrix representation of the exact cover problem, implemented as doubly linked lists of the 1s of the matrix: each 1 element has a link to the next 1 above, below, to the left, and to the right of itself. (Technically, because the lists are circular, this forms a torus). Because exact cover problems tend to be sparse, this representation is usually much more efficient in both size and processing time required. DLX then uses Dancing Links to quickly select permutations of rows as possible solutions and to efficiently backtrack (undo) mistaken guesses.[1]

See also [edit]

- Exact cover
- Dancing Links

References [edit]

1. ^ [a] [b] [c] Knuth, Donald (2000). "Dancing links". arXiv:cs/0011047.

- Knuth, Donald E. (2000), "Dancing links", in Davies, Jim; Roscoe, Bill; Woodcock, Jim, *Millennial Perspectives in Computer Science: Proceedings of the 1999 Oxford-Microsoft Symposium in Honour of Sir Tony Hoare*, Palgrave, pp. 187–214, arXiv:cs/0011047, ISBN 978-0-333-92230-9.

External links [edit]

- Free Software implementation of Algorithm X in C - uses the Dancing Links optimization. Includes examples for using the library to solve sudoku and logic grid puzzles.
- Polycube solver Program (with Lua source code) to fill boxes with polycubes using Algorithm X.
- Knuth's Paper describing the Dancing Links optimization - Gzip'd postscript file.

v · t · e	**Donald Knuth**	[hide]
Publications	*The Art of Computer Programming* · "The Complexity of Songs" · *Computers and Typesetting* · *Concrete Mathematics* · *Surreal Numbers* · *Things a Computer Scientist Rarely Talks About* · *Selected papers series*
Software	TeX · METAFONT · MIXAL (MIX · MMIX · GNU MDK)
Fonts	AMS Euler · Computer Modern · Concrete Roman
Literate programming	WEB · CWEB
Algorithms	**Knuth's Algorithm X** · Knuth–Bendix completion algorithm · Knuth–Morris–Pratt algorithm · Knuth shuffle · Robinson–Schensted–Knuth correspondence · Trabb Pardo–Knuth algorithm · Generalization of Dijkstra's algorithm · Knuth's Simpath algorithm
Other	Dancing Links · Knuth reward check · Knuth Prize · Man or boy test · Quater-imaginary base · -yllion · Potrzebie system of weights and measures

Categories: Search algorithms | Donald Knuth