Note to readers:
Please ignore these
side notes; they're just
hints to myself for
preparing the index,
and they're often flaky!

KNUTH

THE ART OF
COMPUTER PROGRAMMING

VOLUME 4  PRE-FASCICLE 5C

DANCING
LINKS

DONALD E. KNUTH  Stanford University

ADDISON–WESLEY

July 22, 2015
See also http://www-cs-faculty.stanford.edu/~knuth/sgb.html for information about The Stanford GraphBase, including downloadable software for dealing with the graphs used in many of the examples in Chapter 7.
See also http://www-cs-faculty.stanford.edu/~knuth/mmixware.html for downloadable software to simulate the MMIX computer.
Copyright © 2015 by Addison-Wesley
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher, except that the official electronic file may be used to print single copies for personal (not commercial) use.
 Zeroth printing (revision -98), 21 July 2015
July 22, 2015
PREFACE

With this issue we have terminated the section “Short Notes.”

... It has never been “crystal clear” why a Contribution cannot be short,
just as it has occasionally been verified in these pages
that a Short Note might be long.

— ROBERT A. SHORT, IEEE Transactions on Computers (1973)

This booklet contains draft material that I'm circulating to experts in the field, in hopes that they can help remove its most egregious errors before too many other people see it. I am also, however, posting it on the Internet for courageous and/or random readers who don't mind the risk of reading a few pages that have not yet reached a very mature state. Beware: This material has not yet been proofread as thoroughly as the manuscripts of Volumes 1, 2, 3, and 4A were at the time of their first printings. And those carefully-checked volumes, alas, were subsequently found to contain thousands of mistakes.

Given this caveat, I hope that my errors this time will not be so numerous and/or obtrusive that you will be discouraged from reading the material carefully. I did try to make the text both interesting and authoritative, as far as it goes. But the field is vast; I cannot hope to have surrounded it enough to corral it completely. So I beg you to let me know about any deficiencies that you discover.

To put the material in context, this portion of fascicle 5 previews Section 7.2.2.1 of The Art of Computer Programming, entitled “Dancing links.” It develops an important data structure technique that is suitable for backtrack programming (which is the main focus of Section 7.2.2). Several subsections (7.2.2.2, 7.2.2.3, etc.) will follow.

* * *

The explosion of research in combinatorial algorithms since the 1970s has meant that I cannot hope to be aware of all the important ideas in this field. I've tried my best to get the story right, yet I fear that in many respects I'm woefully ignorant. So I beg expert readers to steer me in appropriate directions.

Please look, for example, at the exercises that I've classed as research problems (rated with difficulty level 46 or higher), namely exercises ...; I've also implicitly mentioned or posed additional unsolved questions in the answers to exercises ... Are those problems still open? Please inform me if you know of a solution to any of these intriguing questions. And of course if no solution

July 22, 2015
is known today but you do make progress on any of them in the future, I hope you’ll let me know.

I urgently need your help also with respect to some exercises that I made up as I was preparing this material. I certainly don’t like to receive credit for things that have already been published by others, and most of these results are quite natural “fruits” that were just waiting to be “plucked.” Therefore please tell me if you know who deserves to be credited, with respect to the ideas found in exercises 40, . . . . Furthermore I’ve credited exercises . . . to unpublished work of . . . . Have any of those results ever appeared in print, to your knowledge?

* * *

Special thanks are due to . . . for their detailed comments on my early attempts at exposition, as well as to numerous other correspondents who have contributed crucial corrections.

* * *

I happily offer a “finder’s fee” of $2.56 for each error in this draft when it is first reported to me, whether that error be typographical, technical, or historical. The same reward holds for items that I forgot to put in the index. And valuable suggestions for improvements to the text are worth 32¢ each. (Furthermore, if you find a better solution to an exercise, I’ll actually do my best to give you immortal glory, by publishing your name in the eventual book: –)

Cross references to yet-unwritten material sometimes appear as ‘00’; this impossible value is a placeholder for the actual numbers to be supplied later.

Happy reading!

Stanford, California

99 Umbuary 2015

D. E. K.

July 22, 2015

Knuth
What a dance
do they do
Lordy, how I'm tellin' you!

Don't lose your confidence if you slip,
Be grateful for a pleasant trip,
And pick yourself up, dust yourself off, start all over again.
— DOROTHY FIELDS, *Pick Yourself Up* (1936)

7.2.2.1. **Dancing links.** Blah blah de blah blah blah.
EXERCISES

39. [20] By setting up an exact cover problem, show that it’s impossible to color the
queen graph $Q_8$ (exercise 7.1.4–21) with eight colors.

40. [21] In how many ways can $Q_8$ be colored in a “balanced” fashion, using eight
queens of color 0 and seven each of colors 1 to 8?

50. [21] If we merely want to count the number of solutions to an exact cover problem,
without actually constructing them, a completely different approach based on bitwise
manipulation instead of list processing is sometimes useful.

The following naive algorithm illustrates the idea: We’re given an $m \times n$ matrix
of 0s and 1s, represented as $n$-bit vectors $r_1, \ldots, r_m$. The algorithm works with a
(potentially huge) database of pairs $(s_j, c_j)$, where $s_j$ is an $n$-bit number representing
a set of columns, and $c_j$ is a positive integer representing the number of ways to cover
that set exactly. Let $p$ be the $n$-bit mask that represents the primary columns.

N1. [Initialize] Set $N \leftarrow 1$, $s_1 \leftarrow 0$, $c_1 \leftarrow 1$, $k \leftarrow 1$.

N2. [Done?] If $k > m$, terminate; the answer is $\sum_{j=1}^{N} c_j$.

N3. [Append $r_k$ where possible.] Set $t \leftarrow r_k$. For $N \geq j \geq 1$, if $s_j \& t = 0$, insert
$(s_j + t, c_j)$ into the database (see below).

N4. [Loop on $k$.] Set $k \leftarrow k + 1$ and return to N2.

To insert $(s, c)$ there are two cases: If $s = s_i$ for some $(s_i, c_i)$ already present, we simply
set $c_i \leftarrow c_i + c$. Otherwise we set $N \leftarrow N + 1$, $s_N \leftarrow s$, $c_N \leftarrow c$.

Show that this algorithm can be significantly improved by using the following trick:
Set $u_k \leftarrow r_k \& \overline{p}$, where $f_k = r_{k+1} \& \cdots \& r_m$ is the bitwise OR of all future rows. If
$u_k \neq 0$, we can remove any item from the database for which $s_j$ does not contain $u_k$.
We can also exploit the nonprimary columns of $u_k$ to compress the database further.

51. [25] Implement the improved algorithm of the previous exercise, and compare its
running time to that of Algorithm D when applied to the $n$ queens problem.

52. [M21] Explain how the method of exercise 50 could be extended to give representa-
tions of all solutions, instead of simply counting them.

999. [M00] this is a temporary exercise (for dummies)
7.2.2.1

SECTION 7.2.2.1

39. Each of the 92 solutions to the eight queens problem (see Fig. 68) occupies eight of the 64 cells, so we must find eight disjoint solutions. Only 1597 updates of Algorithm D are needed to show that such a mission is impossible. [In fact no solutions can be disjoint, because each solution touches at least three of the twenty cells 13, 14, 15, 16, 22, 27, 31, 38, 41, 48, 51, 58, 61, 68, 72, 77, 83, 84, 85, 86. See Thorold Gosset, Messenger of Mathematics 44 (1914), 48.]

Henry E. Dudeney found the illustrated way to occupy all two cells, in Tit Bits 32 (11 September 1897), 439; 33 (2 October 1897), 3.]

40. This is an exact cover problem with 92 + 312 + 306 + \cdots + 312 = 3284 rows (see exercise 7.2.2-5). Algorithm D needs about 2 million updates to find the solution shown, and about 83 billion to find all 11,092 of them.

50. Set \( f_m \leftarrow 0 \) and \( f_{m-1} \leftarrow f_m \) \( r_k \) for \( m \geq k \geq 1 \). The bits of \( u_k \) represent columns that are being changed for the last time.

Let \( u_k = u' + u'' \), where \( u' = u_k \) and \( p \). If \( u_k \neq 0 \) at the beginning of step N4, we compress the database as follows: For \( N \geq j \geq 1 \), if \( s_j \& u' \neq u' \), delete \((s_j, c_j)\); otherwise if \( s_j \& u'' \neq 0 \), delete \((s_j, c_j)\) and insert \((s_j \& u'' \mid u', c_j)\).

To delete \((s_j, c_j)\), set \((s_j, c_j) \leftarrow (s_N, c_N)\) and \( N \leftarrow N - 1 \). When this improved algorithm terminates in step N1, we always have \( N \leq 1 \). Furthermore, if we let \( p_k = r_1 \cdots r_{m-1} \), the size of \( N \) never exceeds \( 2^m \), where \( m = \nu_p r_1 f_k \) is the size of the "frontier" (see exercise 7.1.4-55).

In the special case of \( n \) queens, represented as the exact cover problem in (\ref{eq:8}), this algorithm is due to I. Rivin, R. Zabih, and J. Lamping, Inf. Proc. Letters 41 (1992), 253-256. They proved that the frontier for \( n \) queens never has more than \( 3n \) columns.

51. The author has had reasonably good results using a trilly linked binary search tree for the database, with randomized search keys. (Beware: The swapping algorithm used for deletion was difficult to get right.) This implementation was, however, limited to exact cover problems whose matrix has at most 64 columns; hence it could do \( n \) queens via (\ref{eq:8}) only when \( n \leq 1 \). When \( n = 11 \) its database reached a maximum size of 75,009, and its running time was about 25 megamans. But Algorithm D was a lot better: It needed only about 780K updates to find all \( Q(11) = 2680 \) solutions.

In theory, this method will need only about \( 2^m \) steps as \( n \to \infty \), times a small polynomial function of \( n \). A backtracking algorithm such as Algorithm D, which enumerates each solution explicitly, will probably run asymptotically slower (see exercise 7.2.2-12). But in practice, a breadth-first approach needs too much space.

On the other hand, this method did beat Algorithm D on the \( n \) queen bees problem of exercise 7.2.2-13: When \( n = 11 \) its database grew to 364,864 items; it computed \( H(11) = 506,483 \) in just 30 MB, while Algorithm D needed 27 mega-updates.

52. The set of solutions for \( s_j \) can be represented as a regular expression \( \alpha_j \) instead of by its size, \( c_j \). Instead of inserting \((s_j + t, c_j)\) in step N3, insert \((\alpha_j, \alpha_j)\). If inserting \((s_j, \alpha_j)\), when \((s_j, \alpha_j)\) is already present with \( s_j = s, c_j = \alpha_j \cup \alpha_j \). Alternatively, if only one solution is desired, we could attach a single solution to each \( s_j \) in the database.

999. . .
INDEX AND GLOSSARY

He writes indexes to perfection.
— OLIVER GOLDSMITH, Citizen of the World (1762)

When an index entry refers to a page containing a relevant exercise, see also the answer to that exercise for further information. An answer page is not indexed here unless it refers to a topic not included in the statement of the exercise.

Barris, Harry. 1.
Fields, Dorothy. 1.
MPR: Mathematical Preliminaries Redux, v.
Short, Robert Allen, ill.

Nothing else is indexed yet (sorry).

Preliminary notes for indexing appear in the upper right corner of most pages.

If I’ve mentioned somebody’s name and forgotten to make such an index note, it’s an error (worth $2.50).