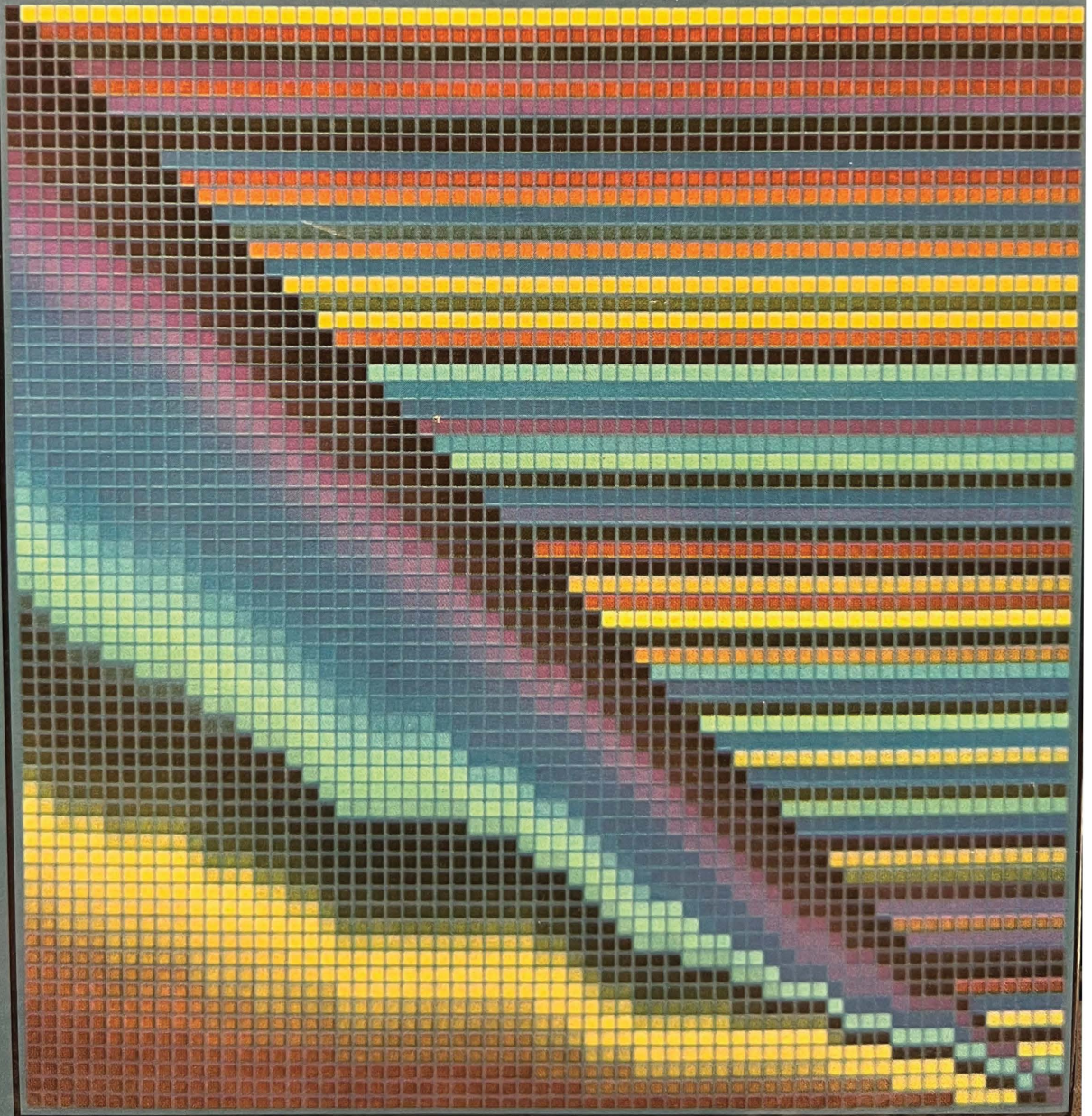


ALGORITHMS

ROBERT SEDGEWICK



EXPERIAN EXHIBIT 1013
IPR2023-01406

ALGORITHMS

ROBERT SEDGEWICK

Princeton University



ADDISON-WESLEY PUBLISHING COMPANY, INC.

Reading, Massachusetts • Menlo Park, California • Don Mills, Ontario
Wokingham, England • Amsterdam • Sydney • Singapore • Tokyo
Madrid • Bogotá • Santiago • San Juan

*To Adam, Brett, Robbie
and especially Linda*

This book is in the
Addison-Wesley Series in Computer Science

Consulting Editor
Michael A. Harrison

Sponsoring Editor
James T. DeWolfe

Library of Congress Cataloging in Publication Data

Sedgewick, Robert, 1946-
Algorithms.

1. Algorithms. I. Title.
QA76.6.S435 1983 519.4 82-11672
ISBN 0-201-06672-6

Reproduced by Addison-Wesley from camera-ready copy supplied by the author.

Reprinted with corrections, August 1984

Copyright © 1983 by Addison-Wesley Publishing Company, Inc.

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 prior written permission of the publisher. Printed in the United States of America.

ISBN 0-201-06672-6
LMNOPQR-HA-8987

Pascal system to perform the computational task implied by a given Pascal program.

Describing Patterns

We'll consider pattern descriptions made up of symbols tied together with the following three fundamental operations.

- (i) *Concatenation*. This is the operation used in the last chapter. If two characters are adjacent in the pattern, then there is a match if and only if the same two characters are adjacent in the text. For example, AB means A followed by B.
- (ii) *Or*. This is the operation that allows us to specify alternatives in the pattern. If we have an "or" between two characters, then there is a match if and only if either of the characters occurs in the text. We'll denote this operation by using the symbol + and use parentheses to allow it to be combined with concatenation in arbitrarily complicated ways. For example, A+B means "either A or B"; C(AC+B)D means "either CACD or CBD"; and (A+C)((B+C)D) means "either ABD or CBD or ACD or CCD."
- (iii) *Closure*. This operation allows parts of the pattern to be repeated arbitrarily. If we have the closure of a symbol, then there is a match if and only if the symbol occurs any number of times (including 0). Closure will be denoted by placing a * after the character or parenthesized group to be repeated. For example, AB* matches strings consisting of an A followed by any number of B's, while (AB)* matches strings consisting of alternating A's and B's.

A string of symbols built up using these three operations is called a *regular expression*. Each regular expression describes many specific text patterns. Our goal is to develop an algorithm which will determine if any of the patterns described by a given regular expression occur in a given text string.

We'll concentrate on concatenation, *or*, and closure in order to show the basic principles in developing a regular-expression pattern matching algorithm. Various additions are commonly made in actual systems for convenience. For example, -A might mean "match any character *except* A." This *not* operation is the same as an *or* involving all the characters except A but is much easier to use. Similarly, "?" might mean "match any letter." Again, this is obviously much more compact than a large *or*. Other examples of additional symbols which might make specification of large patterns easier are symbols which match the beginning or end of a line, any letter or any number, etc.

These operations can be remarkably descriptive. For example, the pattern $?*(ie + ei)?*$ matches all words which have *ie* or *ei* in them (and so