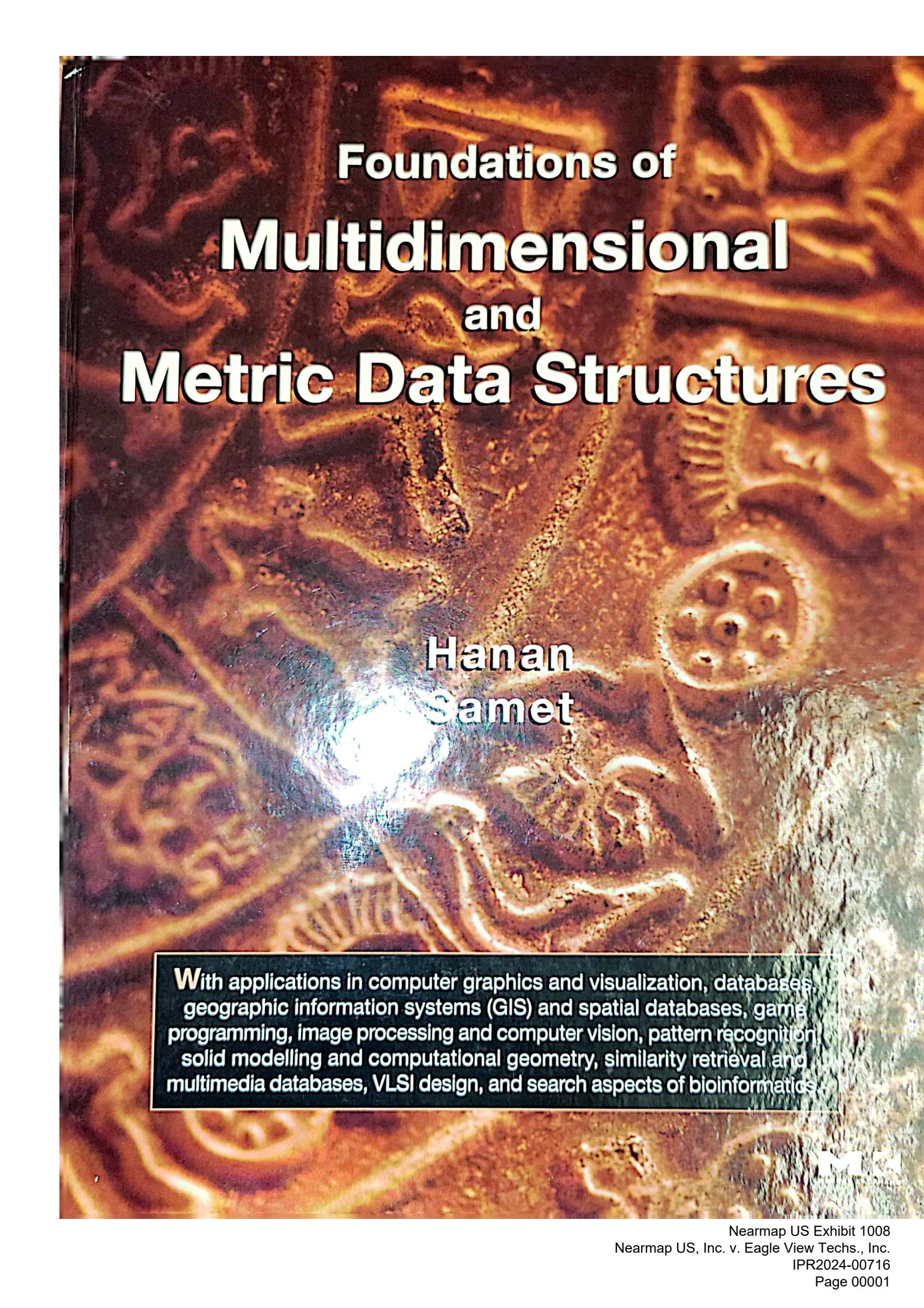




Foundations of Multidimensional and Metric Data Structures

**Hanan
Samet**

With applications in computer graphics and visualization, databases, geographic information systems (GIS) and spatial databases, game programming, image processing and computer vision, pattern recognition, solid modelling and computational geometry, similarity retrieval and multimedia databases, VLSI design, and search aspects of bioinformatics



To Cyrus- thanks a lot for all of your help
in making ACM GIS such a success. It
has been a real pleasure working with you
and I wish you all the best and continued
Success

**Foundations of
Multidimensional and
Metric Data Structures**

Harv!
Seattle, Washington
Nov. 9, 2007

Foundations of Multidimensional and Metric Data Structures

Hanan Samet

University of Maryland, College Park



AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann Publishers is an imprint of Elsevier



MORGAN KAUFMANN PUBLISHERS

Publisher Diane D. Cerra
Publishing Services Manager George Morrison
Assistant Editor Asma Palmeiro
Editorial Intern Marisa Crawford
Cover Design © Picturequest Images
Cover Image Dick Hannus
Project Management Multiscience Press
Composition Multiscience Press & Windfall Software, using ZzT_EX
Technical Illustration Dartmouth Publishing, Inc.
Copyeditor Ken DellaPenta
Proofreader Multiscience Press
Indexer Hanan Samet
Interior printer Maple-Vail Book Manufacturing Group
Cover printer Phoenix Color

Morgan Kaufmann Publishers is an imprint of Elsevier.
500 Sansome Street, Suite 400, San Francisco, CA 94111

This book is printed on acid-free paper.

© 2006 by Hanan Samet. Published by Morgan Kaufmann, an Imprint of Elsevier. All rights reserved.

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

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, scanning, or otherwise—without prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, e-mail: permissions@elsevier.co.uk. You may also complete your request on-line via the Elsevier homepage (<http://elsevier.com>) by selecting "Customer Support" and then "Obtaining Permissions."

Library of Congress Cataloging-in-Publication Data

Application submitted

ISBN 13: 978-0-12-369446-1

ISBN 10: 0-12-369-446-9

For information on all Morgan Kaufmann publications, visit our Web site at www.mkp.com or www.books.elsevier.com

Printed in the United States of America

06 07 08 09 10 5 4 3 2 1

In describing queries involving these relations, we must be careful to distinguish between a point and an object. A *point* is an element in the d -dimensional space from which the objects are drawn. It is not an element of the space into which the objects may be mapped by a particular representation. For example, in the case of a collection of rectangles in two dimensions, a point is an element of the Euclidean plane and not a rectangle, even though we may choose to represent each rectangle by a point in some multidimensional space.

In this chapter, we focus on three types of proximity queries. The first, and most common, is the *point query*, which finds all the objects that contain a given point. It is important to distinguish this query from the point query discussed in Chapter 1 that seeks to determine if a given point p is actually in the dataset (more accurately described as an exact match query). The second type is a *point set query*, which, given a relation $\oplus$ and a set of points Q (typically a region), finds the set of objects S such that $S \oplus Q$ holds. An example is a query, more commonly known as a *window operation*, that finds all the rectangles that intersect a given region. In this example, the relation $\oplus$ is defined by $S \oplus Q$ if $S \cap Q \neq \emptyset$, and Q is the query window. The third type of query is a *geometric join query* (also known as a *spatial join query* [1415]), which, for a relation $\oplus$ and two classes of objects O_1 and O_2 with corresponding subsets S_1 and S_2 , finds all pairs (P_1, P_2) with $P_1 \in S_1$, $P_2 \in S_2$, and $P_1 \oplus P_2$. An example is a query that determines all pairs of overlapping rectangles. In such a case, both O_1 and O_2 correspond to the set of rectangles, and $\oplus$ is the intersection relation. In our examples, S_1 and S_2 are usually the same subsets, although the more general problem can also be handled.

Initially, we present representations that are designed for use with the plane-sweep solution paradigm [119, 1511, 1740]. For many tasks, use of this paradigm yields worst-case optimal solutions in time and space. We examine its use in solving two problems:

1. Reporting all intersections between rectangles (the rectangle intersection problem—Section 3.1)
2. Computing the area of a collection of d -dimensional rectangles (the measure problem—Section 3.2)

We focus on the segment, interval, and priority search trees. They represent a rectangle by the intervals that form its boundaries. However, these representations are usually for formulations of the tasks in a static environment. This means that the identities of all the rectangles must be known if the worst-case time and space bounds are to hold. Furthermore, for some tasks, the addition of a single object to the database may force the reexecution of the algorithm on the entire database.

The remaining representations are for a dynamic environment. They are differentiated by the way in which each rectangle is represented. The first type of representation reduces each rectangle to a point in a usually higher-dimensional space and then treats the problem as if it involves a collection of points (Section 3.3). The second type is region-based in the sense that the subdivision of the space from which the rectangles are drawn depends on the physical extent of the rectangle—it does not just treat a rectangle as one point (Section 3.4). Many of these representations are variants of the quadtree. We show that these quadtree variants are very similar to the segment and interval trees that are used with the plane-sweep paradigm. Moreover, we observe that the quadtree serves as a multidimensional sort and that the process of traversing it is analogous to a plane sweep in multiple dimensions.

3.1 Plane-Sweep Methods and the Rectangle Intersection Problem

The term *plane sweep* is used to characterize a paradigm employed to solve geometric problems by sweeping a line (plane in three dimensions) across the plane (space in three dimensions) and halting at points where the line (plane) makes its first or last intersection

1241, 1304, 1309, 1317, 1471–1472,
 1478, 1481, 1603, 1698, 1992, 2012]
 SparseMap [850, 855, 886]
 Spatial Approximation tree (sa-tree) [853,
 1347–1348, 1704]
 spatial data mining [72, 74, 572, 1173,
 1367–1368, 1839, 1953–1954]
 spatial database [5, 13, 149, 161, 258, 260,
 775, 783–785, 789, 844, 929, 1080, 1238,
 1473, 1546, 1554, 1562, 1645, 1683,
 1746–1747, 1931, 1937–1938, 1963,
 1966, 2023]
 spatial join [7, 47, 81, 83, 88–89, 148, 150,
 160, 256, 259, 261–262, 424, 444, 498,
 560, 592, 772, 779, 845, 860–862, 866,
 892–894, 934–935, 1020, 1072, 1195–
 1198, 1207, 1222–1224, 1415, 1417,
 1458–1459, 1467, 1470, 1474, 1815,
 1867, 1923, 2071, 2073–2074, 2076]
 spatial join index [1210, 1582, 1750]
 spatial k-d tree [1404, 1406–1407]
 spatial networks [377, 491, 605, 612, 715,
 781, 949, 958, 1053–1054, 1181, 1464,
 1493, 1672, 1736, 1748–1749, 1945,
 2055]
 spatial query optimization [75, 78–79, 444,
 566, 1405]
 spatial reasoning [64, 446]
 spatiotemporal database [20–21, 127, 156,
 790, 950, 1203, 1209, 1305, 1614, 1693,
 1895, 2037]
 spatiotemporal pattern retrieval [794]
 spatiotemporal range query [1544]
 sphere tree [895–896, 1410, 1412, 1527,
 1991]