Subject Index

When the page number of an index entry refers to a definition, then the page number is in italics. When the index entry appears in a footnote, then the letter "n" is appended to the page number. A page range for a topic indicates that either the topic is discussed in the given page range or that each page in the range mentions the topic.

0
0-balanced quadtree, *405*, 407, 814

1
1-2 brother tree, 280n
1-balanced quadtree, *405*, 407
1-irregular quadtree, *405*
1DSEARCH, 16
1nf. *See* first normal form (1nf)

2
2-3 tree, 65, 442, *718*, 722
2-3-4 tree, 442, *722*, 722
2-d Fibonacci condition, *223*, 223, 225, 782–783
2-d range tree. *See also* two-dimensional range tree
2DSEARCH, *17*
2nf. *See* second normal form (2nf)

3
(3,0)-AVD, 367, 582, 584–585, *584*, 853
3nf. *See* third normal form (3nf)

4
4-adjacent
  blocks, *217n*
  locations (pixels), *200n*
4-shape, *230*

7
7-shape, *231*

8
8-adjacent
  blocks, *217n*
  locations (pixels), *200n*

9
9-shape, *230*

A
A-series paper, 783
A-tree, 283n, 573
A*-algorithm, 347, 493, 535
AABB. *See* axis-aligned bounding box (AABB)
ABOVE field
  edge record in layered dag, *252*
access structure, xv, 165, *195*
active border, *469*
active rectangle
  multidimensional measure problem, *449*
  plane-sweep method, *429*
AdaBoost machine learning technique, 687
Adaptive Hierarchical Coding (AHC), *71*, *221*, 222, 782
adaptive hierarchical triangulation (AHT), 401–402, *401*, 407
adaptive k-d tree, 58–59, 61, 69–71, 73–74, 129–130, 186, 189–190, 460, 620, 763, 775–776
  regions, 224
adaptive quadtree, 475
adaptive uniform grid, *10n*
adaptively sampled distance field (ADF), 362–365, *363*, 373, 803
  approximation error, 363–365
    absolute, 364
    adaptive, *364*
    relative, 364
  skeleton of object, 364–365
address of a cell in space, *192*, 195
ADF. *See* adaptively sampled distance field (ADF)
ADJ, *218*
adjacency graph, 424
adjacency matrix, 317
adjacency number of a tiling, *198*
ADJQUAD, *754*
admissible order, *200*
AESA, 599, 601, 643–650, 650n, 691, 862
  incremental nearest neighbor query, 645–646
  k nearest neighbor query, 645–646
  range query, 644–645
  search hierarchy, 645, 650
aggregation
  explicit. *See* explicit representation
  implicit. *See* implicit representation
aging, 46, *46*
AHC. *See* Adaptive Hierarchical Coding (AHC)
ALGOL, xx, 743
ALGOL W, xx, 743
alignment problem, *403*, 403, 405–408
all closest pairs query, *485*
all nearest neighbor join (ANN join), *486n*, 490
all nearest neighbors problem, *76*, *485*
α-balanced region, *64*
amortization cost analysis, 143, 731
angle property, *348*, 348–349
animation, 423
ANNBFTRAV, *558*
ANNBFTRAVFARTHEST, *565*, *850*
ANNFNDFTRAV, *564*, *849*
ANNFNDFTRAVFARTHEST, *565*, *851*
ANNINCFARTHEST, *565*, *851*
ANNINCNEAREST, *559*
ANNOPTBFTRAV, *560*
ANNOPTBFTRAVFARTHEST, *565*, *851*
ANNOPTDFTRAV, *564*, *849*
ANNOPTDFTRAVFARTHEST, *565*, *851*
approximate farthest neighbor query, 563
  absolute error bound, 563
  relative error bound, 563
approximate *k* nearest neighbor finding
  best-first algorithm

approximate k nearest neighbor finding *(continued)*
best-first algorithm *(continued)*
general, 557–559
MAXNEARESTDIST, 560
equivalence relation, 490, 557, 651–652
fuzzy logic. *See* fuzzy logic
locality sensitive hashing (LSH). *See* locality sensitive hashing (LSH)
SASH (Spatial Approximation Sample Hierarchy). *See* SASH (Spatial Approximation Sample Hierarchy)
approximate k-center, *623*
approximate k-nearest neighbor finding
equivalence relation, 656
approximate nearest neighbor query, 61, 557–565, 848–852. *See also* ε-nearest neighbor
incremental algorithm, 559–560
k nearest. *See* approximate k nearest neighbor finding
minimum-ambiguity k-d tree. *See* minimum-ambiguity k-d tree
probably approximately correct (PAC). *See* probably approximately correct (PAC) nearest neighbor query
vp-tree. *See* vp-tree
approximate splitting, *46*
Approximate SVD transform matrix, *674*, 674–675
approximate Voronoi diagram (AVD), xxii, 367, 486, 580–585, *580*, 853
(3,0). *See* (3,0)-AVD
(t,ε). *See* (t,ε)-AVD
Approximating and Eliminating Search Algorithm (AESA). *See* AESA
arc tree, 383, *383*, 386, 812
area-area intersection, 386
curve-curve intersection, 386
architecture, 425
area-based rectangle representations, 466–483, 827–833
arrangement, *824*
array pyramid, 266–270, *267*, 790
irregular grid. *See* irregular grid array pyramid
aspect ratio of region, *64*, 70, 580
atomic tile, 197
ATree, 220, *220*, 222, 233, 782
attribute, *1n*
augmented Gabriel graph (AGG), 642–643, *642*, 860–862
nearest neighbor query, 643
augmented gh-tree, 621
Authalic coordinates, 232
AVL tree, 7, 65–66, 82, 164, 718
axes, *467*
axial array, *11*, *136*
axial directory, *11*
axis, *1n*
axis capacity, *476*
axis lines, *467*
axis-aligned bounding box (AABB), *195*
axis-parallel polygon, *195*, *409*, 409n

B

B-partition, *111*
regular. *See* regular B-partition
B-rectangle, *111*
B-region, *111*
B-tree, xviii, 8, 62, 93, 97–98, 98n, 106, 114–116, 136–137, 140, 163, 183, 187, 189, 255–256, 277, 279, 282–283, 285–286, 289, 294, 298–299, 305–308, 310–311, 442, 479, 663, 670, 717–727, *718*, 769, 772, 791, 815, 872–873
balanced. *See* balanced B-tree
biased. *See* biased B-tree
deletion, 720–722, 726–727, 873
rotation, 720
insertion, 719–720, 722, 726
bottom-up, 726
rotation, 719, 726. *See also* deferred splitting
top-down, 727
order, *718*
search, 719, 722, 727, 873
space requirements, 722
storage utilization, 724–725
B^+-tree, 97–98, 98n, 100, 103, 166–167, 171, 183, 216, 255, 279, 366, 411–415, 510, 587–588, 725–726, *725*, 815
deletion, 727, 873
order, *279*
prefix. *See* prefix B^+-tree
search, 727, 873
back-to-front algorithm (painter's algorithm), 236, 787
balanced aspect ratio tree (BAR tree), 64–65, *64*, 70, 764
α-balanced. *See* α-balanced region
aspect ratio. *See* aspect ratio of region
canonical cut. *See* canonical cut
canonical region. *See* canonical region
fat region. *See* fat region
k-cut. *See* k-cut
k-cuttable. *See* k-cuttable
one-cut. *See* one-cut
reduction factor. *See* reduction factor
skinny region. *See* skinny region
balanced B-tree, 164
balanced binary search tree, 14, 18–19, 430, 433, 440–441, 445, 718, 821–822
balanced box-decomposition tree (BBD-tree), 64, 74n, 82–87, *83*, 581, 584, 767
centroid shrink operation. *See* centroid shrink operation in balanced box-decomposition tree (BBD-tree)
inner box. *See* inner box in balanced box-decomposition tree (BBD-tree)
outer box. *See* outer box in balanced box-decomposition tree (BBD-tree)
partitioning box. *See* partitioning box in balanced box-decomposition tree (BBD-tree)
shrink operation. *See* shrink operation in balanced box-decomposition tree (BBD-tree)
split operation. *See* split operation in balanced box-decomposition tree (BBD-tree)
trial midpoint split operation. *See* trial midpoint split operation in balanced box-decomposition tree (BBD-tree)
balanced four-dimensional k-d tree, 461, 482
four-dimensional representative point rectangle representation, 460–462, 826–827
point query, 462
rectangle intersection problem, 460–462
space requirements, 477–478
window query, 462, 477–478
balanced quadtree, *405*
balancing
weight balancing. *See* weight balancing
ball partitioning, 598, *598*, 604–613, 855–856
balltree, 567, 622
band, 385, *385*
BANG file, 82, 107n, 114–118, 125, 129, 167, 171, 256, 479, 581
exact match query, 114–115, 479
two-dimensional representative point rectangle representation, 464
BAR tree. *See* balanced aspect ratio tree (BAR tree)
base prototypes, *623*
batched dynamic data structures, *429*
BBD-tree. *See* balanced box-decomposition tree (BBD-tree)
BD-tree, 74n, 79–83, 87, 107n, 115–116, 115n, 169, 171, 182, 190, 581, 767, 776
exact match query, 82, 87
partial range query, 82, 87
range query, 82
BELOW field
edge record in layered dag, *252*
best-first incremental nearest neighbor query. *See* incremental nearest neighbor finding
best-first k nearest neighbor query. *See* k nearest neighbor finding
Bézier methods, 620
BFTRAV, *549*
BFTRAVFARTHEST, *553*, *847*
biased B-tree, *8*
bicubic Bernstein-Bézier surface, 402
bidistance function, *640*
big O, *xx*
BIGMIN, *92*

BIN field
cnode record in MX-CIF quadtree, *470*
bin method, 434, *443*
rectangle intersection problem, 443
binary heap, *20n*
Binary Line Generalization Tree (BLG-tree), 384, *384*, 386
binary quadtree, *211*
binary search tree, 164, 187
analogy in deletion in point k-d tree. *See* point k-d tree
analogy in deletion in point quadtree. *See* point quadtree
deletion, 31, 52
Binary Searchable Polygonal Representation (BSPR), 384, *384*
compound section. *See* compound section
simple section. *See* simple section
Binary Space Partitioning tree (BSP tree), xxiii, 66–68, *66*, 70, 88, *224*, 224, 233–237, 260, 269, 450, 464n, 573–576, 620, 787
autopartitions, *234*
multi-object. *See* multi-object BSP tree
visibility number, *236*
visibility problem, 233–234
binary tree
extended. *See* extended binary tree
BINCOMPARE, 470–471, *828*
binomial heap, *20n*
bintree, 49, 160, 162, 221–223, *221*, 226–227, 233, 256, 260, 305–306, 372, 409, 411n, 782
generalized. *See* generalized bintree
MX. *See* MX bintree
PR. *See* PR bintree
restricted. *See* restricted bintree
bintree bounding-box cell-tree pyramid, *306*
bioinformatics database, xvi, 164, 609, 692
biquadratic patches, 373
bisector tree, 599, 617–622, *617*, 856–857
eccentric child, *618*
gh-tree
range query, 618
bit concatenation, *90*, 94, 142, 159, 716, 871
range query, 91
reversed. *See* reversed bit concatenation
bit interleaving, *90*, 93–95, 97, 142, 159, 172, 199, 215, 410, 716, 731, 737, 871
Gray code, 93–95
partial range query, 93–95, 768
range query, 91, 93
B-tree, 92, 95, 768
binary search tree, 91–92, 95
reversed. *See* reversed bit interleaving
bit manipulation, 245
bitmap representation, 5–6, 9, 13
compressed. *See* compressed bitmap representation
bk-tree, *610*, 610–612
BLACK value
NODETYPE field
node record in MX quadtree for points, *41*
node record in PR quadtree, *46*
BLG-tree. *See* Binary Line Generalization Tree (BLG-tree)
BLISS, 744
block, *193*
black, *206*
white, *206*
block-nested loop
skyline query. *See* skyline query
bnode record
MX-CIF quadtree, *470*
Boolean query, *3*
BoostMap, 686–687, *686*, 700
Borsuk-Ulam Theorem, *88n*
bottom-up region quadtree construction. *See* region quadtree
boundary face, *336*
boundary model, 313–346, 374, 794–799
Boolean set operations, 372
unambiguous. *See* unambiguous boundary model
Boundary Representation (BRep). *See* boundary model
boundary-based object representation, *192*, 312–408, 794–803, 805–814
image-based, 355–382, 803, 805–811
object-based, 382–399, 812–813
surface-based, 399–408, 814
bounded embedding space, *3*
bounded quadtree (BQT), *476*, 482
space requirements, 477–478
window query, 477–478
bounded-index extendible hashing, *730*
bounding box, *195*
ellipse. *See* ellipse bounding box
bounding-box cell-tree pyramid, *305*, 305, 310–311
bintree. *See* bintree bounding-box cell-tree pyramid
generalized k-d tree. *See* generalized k-d tree bounding-box cell-tree pyramid
point quadtree. *See* point quadtree bounding-box cell-tree pyramid
bounding-box pyramid, 310–311
irregular grid. *See* irregular grid bounding-box pyramid
bounds array, *57*, 461
BOXTREE, 398
Boyce-Codd normal form, 329
branch and bound neighbor finding strategy, 489, 517, 554
BRep. *See* boundary model
bs-tree. *See* bisector tree
BSP tree. *See* Binary Space Partitioning tree (BSP tree)
BSPR. *See* Binary Searchable Polygonal Representation (BSPR)
bucket, *3*, *12*
data. *See* data bucket
overflow. *See* overflow bucket
point. *See* point bucket
primary. *See* primary bucket
region. *See* region bucket
bucket adaptive k-d tree, *61*, 62
bucket address, *729n*
bucket capacity, *45*, 75, 375
bucket generalized k-d trie, *74*, 97, 110–111
bucket generalized pseudo k-d tree, *61*, 61–64, 68, 98, 102–103, 116
bucket merging
EXCELL. *See* EXCELL
grid file. *See* grid file
bucket methods, 95–164, *95*, 170–171, 183–184, 768–775
grid directory. *See* grid directory methods
storage utilization, 163–164
tree directory. *See* tree directory methods
bucket PM quadtree, xxiii, 374, *374*
bucket PM_1 quadtree, *374*, 375–376
bucket PM_2 quadtree, *374*, 375–376, 809
bucket PM_3 quadtree, 374
bucket PMR k-d tree, 82
bucket PMR quadtree, 97
bucket polygon quadtree, *260*, 261–262
vertex. *See* vertex bucket polygon quadtree
bucket PR k-d tree, xxiii, *74*, 75, 82, 97, 110, 116, 160, 164, 171
bucket PR octree
triangulation, 241
bucket PR quadtree, xxiii, 45–46, *45*, 61, 97, 164, 169–170, 182, 187, 581
bucket PR-CIF k-d tree, 476–478, *476*, 482–483
construction cost, 482
deletion, 480
insertion, 480
one-dimensional. *See* one-dimensional bucket PR-CIF k-d tree
space requirements, 477–478, 482
two-dimensional. *See* two-dimensional bucket PR-CIF k-d tree
window query, 477–478, 480
bucket pseudo quadtree, 62–63
bucket rectangle PM quadtree, xxiii, 477, *481*, 481–483
space requirements, 481
window query, 481–482
bucket rectangle quadtree, 481. *See also* bucket rectangle PM quadtree
buddy page, 113, *113*
buddy system, 731
buddy-tree, 97, 110, 112–113, 116
range query, 116
buffer, *402*

buffer tree, 299–301, 794
bulk insertion
  grid file. *See* grid file
  object-tree pyramid. *See* object-tree pyramid
  R-tree. *See* object-tree pyramid
bulk loading, 165
  grid file. *See* grid file
  object-tree pyramid. *See* object-tree pyramid
  R-tree. *See* object-tree pyramid
BV-tree, 12, 82, 97, 107n, 116–129, 141n, 167, 187, 770–773
  construction, 120–122, 770–771
  deletion, 122, 771
  directory node, *117*
  exact match query, 122–125, 771–772
  point page, *117*
  space requirements, 125–127, 772

C

C field
  rectangle record in MX-CIF quadtree, *470*
C programming language, xx, 743, 745
C++, xx, 743
CAD. *See* computer-aided design (CAD)
Caltech Intermediate Form (CIF) quadtree, *466*
canonical cut, *64*, 70
canonical region, *64*
Cantor-diagonal order, 199–202, *199*, 776–778
Cartesian tree, 19n, *27*, 27, 438, 751
CCF, *319*
CCFFCCW, *321*
CCFFCCW field
  edge-edge record in oriented winged-edge data structure, *326*
CCFFCW, *321*
CCFFCW field
  edge-edge record in oriented winged-edge data structure, *326*
CCQUAD, *35*
CCV, *319*
CCVVEND, *321*
CCVVEND field
  edge-edge record in oriented winged-edge data structure, *326*
CCVVSTART, *321*
CCVVSTART field
  edge-edge record in oriented winged-edge data structure, *326*
CCW, *319*
CDT. *See* convex difference tree (CDT)
cell, *449*
  address. *See* address of a cell in space
  black, *202*
  location. *See* location of a cell in space
  white, *202*
cell pyramid, *266*, 266–267, 272, 304
cell tree, *312*
cell-based decomposition. *See* image-based decomposition
cell-based query. *See* location-based query
cell-tree pyramid, 267–269, *267*, 305
  bintree bounding-box. *See* bintree bounding-box cell-tree pyramid
  bounding-box. *See* bounding-box cell-tree pyramid
  generalized k-d tree bounding-box. *See* generalized k-d tree bounding-box cell-tree pyramid
  point quadtree bounding-box. *See* point quadtree bounding-box cell-tree pyramid
CENTER field
  square record in PM quadtree, *367*
centroid shrink operation in balanced box-decomposition tree (BBD-tree), *84*
CF, *319*
CFFCCW, *321*
CFFCCW field
  edge-edge record in oriented winged-edge data structure, *326*
CFFCW, *321*
CFFCW field
  edge-edge record in oriented winged-edge data structure, *326*
chain, *243*
chain code, xxii, 313, 416–417, 816–817
chain tree, *243*
CHAINID field
  TYPE field value gap
    ldnode record in layered dag, *252*
Chamfer metric, 210, *210*, 779
change-based runlength encoding, *409*, 414, 416
Chessboard metric, 779
CHILD field
  bnode record in MX-CIF quadtree, *470*
  cnode record in MX-CIF quadtree, *470*
  node record in MX quadtree for points, *41*
  node record in PM quadtree, *367*
  node record in point k-d tree, *52*
  node record in point quadtree, *30*
  node record in PR quadtree, *46*
CHILDTYPE function
  point k-d tree, *52*
  point quadtree, *31*
CHOOSECHILD, *180*
CIF. *See* Caltech Intermediate Form (CIF) quadtree
CIFCOMPARE, *827*
CIFDELETE, *830*
CIFINSERT, *829*
CIFSEARCH, *828*
circle property, *347*, 347–348, 351
City Block distance, 32. *See also* Manhattan distance
City Block metric, 779
clevel, *118*
CLIPLINES, *804*
clipping, 366
clique, *622*
close-reinsert, *290*
closest pairs. *See* all closest pairs query
cluster, *598*
cluster center, *598*
cluster detection, 62, 269
cluster preservation ratio (CPR), *696*
clustering-based distance-based indexing methods, 598–599, 625
CMAT. *See* corner medial axis transformation (CMAT)
cnode record
  MX-CIF quadtree, *470*
code, *90*
collapsing
  MX quadtree. *See* MX quadtree
  MX-CIF quadtree. *See* MX-CIF quadtree
  PR quadtree. *See* PR quadtree
collision detection, xix, 392–393, 398, 567
collision set
  locality sensitive hashing (LSH). *See* locality sensitive hashing (LSH)
color table, *411*, 415, 814
column order, *200*, 278
combined index, 5–6, *5*
  partial range query, 6, 13, 747
common partition elimination, *228*
compact, *315n*
compaction methods
  difference-based. *See* difference-based compaction methods
companion half-edge, *328*
companion transition, *331*
complete containment rule, *261*, 261n
complete quadtree, 402, 424
complete sector tree, *380*
complete t-ary tree, *30n*
component
  eight-connected. *See* eight-connected component
  four-connected. *See* four-connected component
COMPRESS, *784*
compressed bitmap representation, *6*, 9
Compressed DFT. *See* Compressed Discrete Fourier Transform (CDFT)
Compressed Discrete Fourier Transform (CDFT), 680–681, *680*, 683–684
compressed quadtree, *410*, *581*
COMPRESSTREE, *784*
computational geometry, xv, xix, xxii, 427
computer graphics, xv–xvi, xviii–xix, xxii
computer vision, xv–xvii, xx, xxii, 425
computer-aided design (CAD), xv, xvii, 260
COND, 744
condensing point, *514*
cone tree, 380–382, *380*, 811
  Boolean set operations, 381
  contact point. *See* contact point
  rotation, 381

scaling, 381
translation, 381
conflation problem, *486*
CONFLICT, *237*
conforming Delaunay triangulation, 351, *351*
conforming mesh, *401*, 403, 405–406
conjugation tree, 88–89, *88*, 767–768
connected component labeling, *200n*
connected objects, *191n*
connectivity table, *355*, 355, 802
constrained Delaunay triangulation, 351, *351*, 394
constraint database, xix, 445
constructive solid geometry (CSG), 314, *314*, 372, 374, 393, 409, 417–418, 453, 809
contact point
cone tree, *381*
sector tree, *377*
containment hierarchy, 117, *119*
level. *See* level
containment query, *427*, *444*
one-dimensional. *See* one-dimensional containment query
representative point rectangle representation. *See* representative point rectangle representation
two-dimensional interval tree. *See* two-dimensional interval tree
two-dimensional segment tree. *See* two-dimensional segment tree
containment set, *444*
interval tree. *See* interval tree
priority search tree. *See* priority search tree
segment tree. *See* segment tree
continuous multiresolution model, 399
contractive embedding, *664*, *See also* contractive mapping; pruning property
contractive mapping, *664*, *See also* contractive embedding; pruning property
convex difference tree (CDT), 262–264, 790
convex hull, 195
convex polygon, *503*
COORD field
node record in point k-d tree, *52*
coordinate, *1n*
corner lath data structure, 333, 335–346, 799
corner medial axis transformation (CMAT), *208*, 210
corner stitching, 459–460, *459*, 826
point query, 460–461
solid tile. *See* solid tile
space tile. *See* space tile
window query, 460–461
corner table, *354*, 355
corner transformation, *454*, 670
corner* lath data structure, 342
correctness criterion, 498, 520
corridor, *402*
cost-based unbalanced R-tree (CUR-tree), *302*
COUNTCHILDS, *176*
cover fieldtree, 257–259, *257*, 475, 480. *See also* loose quadtree
coverage, *274*
coverage-based splitting, 255–260, *255*, 789
covers, *430*
CQUAD, *35*
cracks, *402*
create, *743*
CREATEPNODE, *756*
criterion 1 (one), *32*
criterion 2 (two), *32*
cross-hairs, *63*
CROSSAXIS, *828*
CSG. *See* constructive solid geometry (CSG)
CSG tree, 372
well behaved, 372
cube, *232*
cubetree, *297*, 298
CUR-tree. *See* cost-based unbalanced R-tree (CUR-tree)
curse of dimensionality, xvi–xvii, 486–489, *486*, 561, 563, 592n, 663, 669
CV, *319*
CVVEND, *321*
CVVEND field
edge-edge record in oriented winged-edge data structure, *326*
CVVSTART, *321*
CVVSTART field
edge-edge record in oriented winged-edge data structure, *326*
CW, *319*

D

D-CEL data structure, 323–324, *323*, 329, 343
D-CEL-face. *See* D-CEL-face data structure
D-CEL-face-vertex. *See* D-CEL-face-vertex data structure
D-CEL-vertex. *See* D-CEL-vertex data structure
D-CEL-face data structure, *324*, 324, 343
D-CEL-face-vertex data structure, *324*, 329, 796
D-CEL-vertex data structure, *324*, 324, 343
D-Euclidean
metric. *See* D-Euclidean metric
D-Euclidean metric, 210
d-heap, *20n*
D-tree, 68–69, *68*, 224
data bucket, *103*
DATA field
edgelist record in PM quadtree, *368*
data mining, 164
data organization methods, *3*, 28, 49, 77, 95, 184–185, 188, 190, 455
data-dependent strategy
LSD tree. *See* LSD tree
database, xxii
bioinformatics. *See* bioinformatics database
constraint. *See* constraint database
dynamic. *See* dynamic database
moving object. *See* moving object database
multimedia. *See* multimedia database
spatial. *See* spatial database
spatiotemporal. *See* spatiotemporal database
static. *See* static database
database management systems (DBMS), xv, xxii
DBMS. *See* database management systems (DBMS)
d_C, *210*, *See also* Chamfer metric
dead area, *274*
decimation
surface. *See* surface simplification
DECODE2DRE, *815*
DECODE2DRE2, *816*
decomposition
image-based. *See* image-based decomposition
object-based. *See* object-based decomposition
quaternary hierarchical triangular. *See* quaternary hierarchical triangular decomposition
regular. *See* regular decomposition
ternary hierarchical triangular. *See* ternary hierarchical triangular decomposition
decomposition hierarchy, 117
decoupling, 165–167, *165*
deferred splitting, *98*, 98n, *283*, 285, 289, 308, 719
Delaunay graph, 346–348, 352–353, 598, *629*, 629n, 630–633, 637–639, 641, 858
nearest neighbor query, 630, 650
Las Vegas algorithm, 712
Delaunay tetrahedralization, xxii, 352–353, *352*
Delaunay triangulation (DT), xxii, *241*, 322, 347–354, *347*, 367, 377, 400, 402, 584, 629n, 631–632, 638–641, 640n, 643, 788, 799–802, 861–862
conforming. *See* conforming Delaunay triangulation
constrained. *See* constrained Delaunay triangulation
nondegenerate. *See* nondegenerate Delaunay triangulation
DEM. *See* Digital Elevation Model (DEM)

density-based splitting, *255*, 260–264, 790. *See also* bucket methods
depth-first *k* nearest neighbors query. *See k* nearest neighbor finding
DEQUEUE, *518*
DF-expression, *410*, 411, 414, 814
DFT. *See* Discrete Fourier Transform (DFT)
DFTRAV, *519*
DFTRAVFARTHEST, 521, *836*
DIAGONALNEIGHBOR, *781*
diameter of a point set, 699, *699*, 867
DICTIONARY field
  node record in PM quadtree, *367*
difference neighborhood graph (DNG), *642*, 642, 860
difference-based compaction methods, 408–423, 814–819
Digital Elevation Model (DEM), 262, 400
digital geometry, xx
digital searching, *2*
Digital Terrain Model (DTM), 262, 400
digital tree, *2*
Dijkstra's shortest-path algorithm, 509, 513–515
dimension, *1n*
dimension reduction methods, 662–685, 697, 709, 864–867
  Discrete Fourier Transform (DFT). *See* Discrete Fourier Transform (DFT)
  representative point, 670–671
  Singular Value Decomposition (SVD). *See* Singular Value Decomposition (SVD)
  using only one dimension, 668–670, 865
directed Hausdorff distance, *392n*, 392n
directory hierarchy, *117*
  level. *See* level
directory merging
  EXCELL. *See* EXCELL
  grid file. *See* grid file
directory node
  BV-tree. *See* BV-tree
  PK-tree. *See* PK-tree
directory page, *103*
  external. *See* external directory page
  internal. *See* internal directory page
Dirichlet domain. *See* Voronoi diagram
DISC field
  node record in point k-d tree, *52*
disc property, *347*, 347, 351, 640, 800
Discrete Cosine Transform (DCT), 676
Discrete Fourier Transform (DFT), 676–685, 866–867
  adjoint of a unitary matrix, *680*
  coefficients, *676*
  compressed. *See* Compressed Discrete Fourier Transform (CDFT)
  ith harmonic, *683*
  Parseval's theorem. *See* Parseval's theorem
  sorted. *See* Sorted Discrete Fourier Transform (SDFT)
  subtrail, *681*
  trail, *681*
  unitary matrix, *680*
  value, *683*
  variance. *See* Variance Discrete Fourier Transform (VDFT)
discrete multiresolution model, 399
discrete Voronoi diagram, 490
discriminator zone expression (DZE), *80*
disjoint object pyramid, 310–312, *310*
disk-resident data, 2
distance
  bidistance. *See* bidistance function
  edit. *See* edit distance
  Hamming. *See* Hamming edit distance
  Hausdorff. *See* Hausdorff distance
  Levenshtein. *See* Levenshtein edit distance
  multi. *See* multidistance function
  network, 487
  pseudo-Euclidean. *See* pseudo-Euclidean distance
distance browsing, *491*
distance field, *363*
distance join, *489*
distance matrix methods, 598, 643–650, 862
distance metric, *488*
distance scan, *491*
distance semijoin, 486, *490*, *658*
distance-based indexing methods, 598–662, 855–864
  clustering-based. *See* clustering-based distance-based indexing methods
  pivot-based. *See* pivot-based distance-based indexing methods
distance-preserving embedding, 687–689
distortion, 687, *688*
distribution-dependent strategy
  LSD tree. *See* LSD tree
divide-and-conquer methods, xvii, xxii
divided k-d tree, *65*
divided-leaf octal tree, *370*
D_k, *518*
dlevel, *118*
dodecahedron, *232*
dominance merge, *461*
dominance relation, *461*, 503
double balance, *753*
double Gray order, *94*, 94, 199–202, *199*, 216, 776–779
  partial range query, 94–95, 768
  range query, 95
doubly chained tree (DCT), 6–8, *7*
  exact match query, 7, 13, 747
doubly linked list, 432
Douglas-Peucker line generalization algorithm, 383–384, 390, 393
DOWN field
  TYPE field value edge
    ldnode record in layered dag, *252*
  TYPE field value gap
    ldnode record in layered dag, *252*
DQP. *See* Dynamically Quantized Pyramid (DQP)
DQS. *See* Dynamically Quantized Space (DQS)
DR-tree, *303*
DT. *See* Delaunay triangulation (DT)
DTM. *See* Digital Terrain Model (DTM)
dual contraction, *396*
dual graph, *322*
dual mesh, 322
dynamic database, 2
dynamic hashing, *730*, 730
dynamic interval tree, *438*, 439, 821
  rectangle intersection problem, 438
  spatial join query, 439, 821
dynamic multipaging, *136*, 147n, 154, 159
dynamic p hashing, *151*
  range query, 152
dynamic programming, 136, 783
dynamic pseudo k-d tree
  regions, 224
dynamic storage allocation, 731
dynamic z hashing, 140, 150–152, *151*, 774
  range query, 151–152
Dynamically Quantized Pyramid (DQP), 62–64, *62*, 186, 269
  cross-hairs. *See* cross-hairs
  space requirements, 64
  warping process. *See* warping process
Dynamically Quantized Space (DQS), 62–64, *62*
  space requirements, 64

E

Earth, 231
Earth's curvature, 231
ECDF tree, 7, *13*
  range query, 14, 748
  ranking, 14, 748
  space requirements, 14, 748
edge
  oriented. *See* oriented edge
EDGE field
  edge_orientation_pair record in ORIENTEDFACEEDGETABLE and ORIENTEDVERTEXEDGETABLE, *326*
  TYPE field value edge
    ldnode record in layered dag, *252*
edge quadtree, 359–362, *360*, 364–365, 803
edge record
  layered dag, *252*
edge-based object representations using winged-edges, 317–329, 345, 795–796
edge-EXCELL, 365, 370
edge-face relation, *316*, 317
  multiple shells and multiply connected faces, 317

edge-lath table, *332*
edge-loop relation, 317
edge-ordered dynamic tree, 253
edge-ordering property, *243*
edge-vertex relation, 317
edgelist record
  PM quadtree, *368*
edge_orientation_pair record
  oriented winged-edge data structure, *326*
edit distance, 488
  Hamming. *See* Hamming edit distance
  Levenshtein. *See* Levenshtein edit distance
efficient pixel interleaving compression technique (EPICT), 215, *215*, 219, 781
eight-connected, *200n*
eight-connected component, *200n*
eight-connected region. *See* eight-connected component
Elias algorithm, *593*
ellipse bounding box, 383
else, *744*
elseif, *744*
embedding, *687*
  Euclidean. *See* Euclidean embedding
  isometric. *See* isometric embedding
  Lipschitz. *See* Lipschitz embedding
embedding methods, *489*, 515, 598, 601, 612, 623, 647, 685–716, 867–871
  distance-preserving. *See* distance-preserving embedding
  distortion. *See* distortion
  Euclidean. *See also* Euclidean embedding
  isometric. *See* isometric embedding
  Lipschitz. *See* Lipschitz embedding
  proximity-preserving. *See* proximity-preserving embedding
  weakly proximity preserving. *See* weakly proximity preserving embedding
embedding space, *2*
  bounded. *See* bounded embedding space
embedding space organization methods, *3*, 10, 28, 49, 77, 95, 184–185, 188, 190, 455, 489n
empirical cumulative distribution function tree (ECDF tree). *See* ECDF tree
EN. *See* extended neighborhood (EN)
ENCCW(*e*) edge around a vertex, *319n*
ENCCW(*e*) edge in a face, *319n*
enclosure query, *427*, *444*
  interval tree. *See* interval tree
  priority search tree. *See* priority search tree
  representative point rectangle representation. *See* representative point rectangle representation
  two-dimensional interval tree. *See* two-dimensional interval tree
  two-dimensional segment tree. *See* two-dimensional segment tree
enclosure set. *See* containment set
ENCW(*e*) edge around a vertex, *319n*
ENCW(*e*) edge in a face, *319n*
endif, *744*
ENQUEUE, *518*
envelope, 395
EPCCW(*e*) edge around a vertex, *319n*
EPCCW(*e*) edge in a face, *319n*
EPCW(*e*) edge around a vertex, *319n*
EPCW(*e*) edge in a face, *319n*
EPICT. *See* efficient pixel interleaving compression technique
epsilon query, *664*, *See also* range query
ε-nearest neighbor, 580–581
equator, 231–232
equivalence relation
  approximate *k* nearest neighbor finding. *See* approximate *k* nearest neighbor finding
  fuzzy logic. *See* fuzzy logic
error-bound method, *256*, 257
Escher staircase, 218
Euclidean distance, 55
Euclidean embedding, *687*
Euclidean matching problem, 190, 473, 776
Euler's formula, 242, 789
exact match query, *3*, 192, 428
  BANG file. *See* BANG file
  BD-tree. *See* BD-tree
  doubly chained tree (DCT). *See* doubly chained tree (DCT)
  grid file. *See* grid file
  kB-tree. *See* kB-tree
  LSD tree. *See* LSD tree
  multidimensional B-tree (MDBT). *See* multidimensional B-tree (MDBT)
  multidimensional extendible hashing (MDEH). *See* multidimensional extendible hashing (MDEH)
  nested interpolation-based grid file (NIBGF). *See* nested interpolation-based grid file (NIBGF)
  point k-d tree. *See* point k-d tree
  quintary tree. *See* quintary tree
  sequential list. *See* sequential list
  Spatial Approximation tree (sa-tree). *See* Spatial Approximation tree (sa-tree)
EXCELL, xxii–xxiii, 46, 130–131, 135, 137–140, 153n, 160–163, 186–189, 370, 775–776
  bucket merging, 140
  directory merging, 140
excluded middle vantage point forest, *612*
EXHASH, 137, 140, 163
expanded MX-CIF quadtree, 256, 481, *481*, 483
  construction cost, 481, 483
  rectangle intersection problem, 481, 483
  space requirements, 481, 483
  window query, 483
expansion index, *148*
explicit aggregation. *See* explicit representation
explicit representation, *194*
explicit winged-edge data structure. *See* non-oriented winged-edge data structure
exponent of a floating-point number, 2
extended binary tree, 98, *98n*, 102–103, 105–106, *718n*, 751, 872
extended neighborhood (EN), 638–639, 641
extended octree, *370*
extended pyramid technique, *589*
extendible hashing, 46, 135, 140, 730
  bounded-index. *See* bounded-index extendible hashing
Extensible and Flexible Library (XXL), 271
extension parameters, *454*, 455
extent-based object aggregation techniques, 282–289, 791–792
external directory page, *103*
external methods in SP-GiST, 188
external searching, *717*
EXTRACTEDGESINCIDENTATVERTEX, *795*
EXTRACTEDGESOFFACE, *322*
EXTRACTORIENTEDEDGESINCIDENTATVERTEX, *796*
EXTRACTORIENTEDEDGESOFFACE, *326*

F

face decimation, 395–399
face octree, 361–362, *361*, 365, 371, 803
  space requirements, 362, 373, 809
face simplification, *395*
face-based object representations using laths, 330, 344
face-edge octree, *362*, 362
  space requirements, 362
face-edge relation, *316*, 317
  multiple shells and multiply connected faces, 317
face-edge table, *318*
face-lath table, *332*
face-loop relation, 317
FACEEDGETABLE, *320*
fair-split tree, 58–59, 186
false dismissal, *663*
false hit, *273*, *663*
far-reinsert, *290*
farthest neighbor query, 667
  approximate. *See* approximate farthest neighbor query
  incremental algorithm. *See* incremental farthest neighbor finding
  k farthest. *See k* farthest neighbor finding
farthest-point clustering, *623*
Fast Fourier Transform (FFT), 684, 867
fast nearest-neighbor classifier, 486. *See also* nearest neighbor query; nearest object query
FastMap, 686, 690, 695–711, 867–871
  deriving first coordinate value, 699–700, 867–868

FastMap *(continued)*
deriving remaining coordinate values, 701–703, 868–869
expansion of distances, 707–708, *707*, 870–871
farthest pair of objects. *See* diameter of a point set
heuristics for non-Euclidean metrics, 708–709
pivot object selection, 698–699, 867
pivot objects, *697*
projected distance, 700–701
pruning property, 703–707, 869–870
spread between pivot objects, *698*
fat region, *64*, 580
FCCW(*e*), *319*
FCW(*e*), *319*
feature detection, 425
feature extraction, 425
feature query, *191*
feature vector, *485*
feature-based query, *192*
Fibonacci heap, *20n*
Fibonacci number, 223, 767
field, *1n*
fieldtree, 259
cover. *See* cover fieldtree
partition. *See* partition fieldtree
file, *1*
filial set, *7*, 8
filter tree, *255*, 255–256, 259
filter-and-refine method, xvii, *663*
nearest neighbor finding, *666*
filtering, *663*
FINDCANDIDATE, *753*
FINDCHILD, *175*
FINDDMINIMUM, *53*, *762*
FINDFATHER, *763*
findpath problem, 425
FINDY, *252*
finite metric space, *598*
finite-element analysis, xv, 211, 425
first normal form (1nf), 317n
fixed-grid method, xxiii, 10–12, 14, 57, 66, 91, 136, 186, 189, 776
fixed-height fixed-queries tree, 611–613, *611*
fixed-queries array, 601, *611*, 612–613
fixed-queries tree, 601, *611*, 612–613, 856
FLEX, xx, 743
floating-point arithmetic, 197
floating-point representation
exponent. *See* exponent of a floating-point number
mantissa. *See* mantissa of a floating-point number
FNDFTRAV, *523*
FNDFTRAVFARTHEST, *529*, *839*
foldover, *397*
forced reinsertion, 285, 289–290, *289*, 292n, 293–294, 300, 303
forest, *722n*
FORMNODEFORMINENCLOSINGBLOCK-WITHKITEMS, *175*
four-connected, *200n*
four-connected component, *200n*
four-connected region. *See* four-connected component
four-dimensional representative point rectangle representation, 460–463, 826–827
balanced four-dimensional k-d tree. *See* balanced four-dimensional k-d tree
grid file. *See* grid file
k-d tree. *See* k-d tree
LSD tree. *See* LSD tree
fq-array. *See* fixed-queries array
fq-tree. *See* fixed-queries tree
fractional cascading
point location query, 127, 251, 435n
dynamic, 253
front-to-back algorithm, 236, 787
FRONTPRIORITYQUEUE, 501
fuzzy logic
approximate *k* nearest neighbor finding, 651
equivalence relation, 651

G

g-degrees rotated *x*-range, 64n, 195
Gabriel graph (GG), *640*, 640–643, 860–862
nearest neighbor query, 643
game programming, xv, xvii, xix, xxii, 259
gap, *245*
gap tree, 245, *245*, 248–254, 788–789
point location query, 246–248, 788–789
GBI. *See* Generalized Bulk Insertion (GBI)
GBT. *See* generalized balanced ternary (GBT)
generalization, *383*
generalized balanced ternary (GBT), *231*
Generalized BD-tree (GBD-tree), *82*
generalized bintree, *71*, *221*, 222, 782
Generalized Bulk Insertion (GBI), 297–298, *298*
window query, 298
generalized hyperplane partitioning, 598, *598*, 603, 613–624, 856–858
generalized hyperplane tree. *See* gh-tree
generalized k-d tree, *50*
regions, 224–225, 305, 311
generalized k-d tree bounding-box cell-tree pyramid, *306*, *See also* k-d-B-tree for regions
generalized k-d trie, *71*, 110, 112–113
generalized k^+-d tree, *98*, 102–103, 105–108, 108n, 110, 768–769
generalized k^+-d trie, 112n
generalized pseudo k-d tree, *58*, 61, 70, 97–98, 103, 129, 186
deletion, 65
range query, 65
Generalized Search Tree (GiST), 189, 271, 590
geographic information system (GIS), xv, xix, xxii, 164
geometree, *370*
geometric join query, *428*
geometric modeling, xix
Geometric Near-neighbor Access Tree (GNAT), 529, 591–592, 616–617, 622–624, 622n, 630, 646, 856
Dirichlet domain, 617
incremental nearest neighbor query, 617
range query, 617
split point, 616
geometric probability, 369, 376
geometric simplification. *See* surface simplification
GETNETWORKDISTINTERVAL, *511*
GG. *See* Gabriel graph (GG)
gh-tree, xxiii, 599, 609, 616–619, 621–622, 630, 636n, 649
augmented. *See* augmented gh-tree
incremental nearest neighbor query, 614–616
peer-to-peer (P2P), 614
range query, 614, 616
search hierarchy, 614–615
GIS. *See* geographic information system (GIS)
GiST. *See* Generalized Search Tree (GiST)
global, *743*
GNAT. *See* Geometric Near-neighbor Access Tree (GNAT)
GPU. *See* graphics processing unit (GPU)
graph
object. *See* object graph
planar. *See* planar graph
graph drawing, 64
graphical search-and-replace operation, 459
graphics processing unit (GPU), xviii
Gray code, 93–95, *93n*, *212*
gray level, 424
gray node, *214*
Gray order, *94*, 94, 199–202, *199*, 216, 776–779
partial range query, 95, 768
range query, 95
GRAY value
NODETYPE field
node record in MX quadtree for points, *41*
node record in PM quadtree, *367*
node record in PR quadtree, *46*
greedy insertion technique, 393
greedy set cover algorithm, 716
greedy triangulation, *240*
grid cell, *130*
grid directory, 12, *96*, *130*
grid file. *See* grid file
grid directory methods, 130–163, 773–775
grid file, xxii–xxiii, 46, 110n, 115, 130–

137, 140, 146–148, 160, 162–163, 186–189, 210, 594, 773, 776
bucket merging, 135
buddy system, *135*
merging threshold, *135*
neighbor system, *135*
bulk insertion, *136*
bulk loading, *136*
directory merging, 135
exact match query, 136
four-dimensional representative point rectangle representation, 462
grid directory, *131*
grid refinement, 134
linear scale. *See* linear scale
multilayer. *See* multilayer grid file
partial range query, 134, 136
range query, 132, 136
rectangle intersection problem, 462
space requirements, 136
two-level. *See* two-level grid file
grid refinement
grid file. *See* grid file
grid-partition index, *582*
grouping process, *165*
growth factor, *735*
guard, *118*

H

H-rectangle, *449*
h-refinement, *405*
Haar Transform, 676
HAL tree. *See* hierarchical approximation and localization tree (HAL tree)
half-edge, *328*
companion. *See* companion half-edge
half-edge data structure, 328, 331, 333, 343–345
half-edge-face. *See* half-edge-face data structure
half-edge-vertex. *See* half-edge-vertex data structure
nonmanifold object, 316
half-edge lath data structure. *See* split-vertex lath data structure
half-edge-face data structure, *328*, 329
half-edge-vertex data structure, *328*, 329, 346, 796, 799
half-quad data structure, *343*
halfplanar range query, 88
halfspace
linear. *See* linear halfspace
halting point, *429*
Ham-Sandwich cut, *89*, 768
Ham-Sandwich Theorem, *88*
Ham-Sandwich tree, *88*
Hamming edit distance, *488n*, 603
Hamming space, *712*
Hamming space mapping, 712
hashing, 466
linear. *See* linear hashing
linear probing. *See* linear probing
separate chaining method. *See* separate chaining method
spiral. *See* spiral hashing
Hausdorff distance, 392, *392n*, 488
directed. *See* directed Hausdorff distance
hB-tree, 12, 82, 97, 106–110, 114, 116–118, 125, 128–129, 167, 171, 187, 581, 768–769
hB$^{\Pi}$-tree, *108n*, 108n, 109
sibling pointer. *See* sibling pointer
heap, *20n*, 187
binary. *See* binary heap
binomial. *See* binomial heap
d-heap. *See* *d*-heap
Fibonacci. *See* Fibonacci heap
sequential. *See* sequential heap
HEAPMAX field
node record in priority search tree, *21*
height-balanced binary search tree, 7–8, 66, 280n
exact match query, 13, 747
hexagonal grid. *See* hexagonal tiling
hexagonal tiling, *197*, 198, 205, 230–231
4-shape. *See* 4-shape
7-shape. *See* 7-shape
9-shape. *See* 9-shape
hexahedron, *232*
HICHILD field
node record in point k-d tree, *52*
hidden-line elimination, 423
hidden-surface elimination, 234, 236, 423, 787
hierarchical approximation and localization tree (HAL tree), 389–391, *389*, 813
construction, 391, 813
polysurface. *See* polysurface
space requirements, 391, 813
hierarchical Delaunay triangulation (HDT), *402*, 402
hierarchical face clustering, 395–399, *395*
circularity. *See* irregularity in hierarchical face clustering
irregularity. *See* irregularity in hierarchical face clustering
hierarchical probe model, *377*
hierarchical rectangular decomposition. *See* quaternary hierarchical rectangular decomposition
hierarchical space indexing method, *370*
hierarchical triangular decomposition
quaternary. *See* quaternary hierarchical triangular decomposition
ternary. *See* ternary hierarchical triangular decomposition
hierarchy
ordinary. *See* ordinary hierarchy
reflection. *See* reflection hierarchy
rotation. *See* rotation hierarchy
High variable
layered dag, *252*
high-energy physics, 211
Hilbert B-tree, *280n*
Hilbert packed R-tree, *276*, 276–278, 280–281, 280n, 294, 296–297, 301, 587
Hilbert R-tree, 275n, *280*, 280, 280n, 285
Hilbert tree, *280n*
hinted quadtree, 482–483
construction cost, 482
hinted distance, *482*
owner pointer, *482*
space requirements, 482
window query, 483
hole, *200n*
holey brick tree (hB-tree). *See* hB-tree
homogeneous image, *423*
homogeneous k-d tree, 58n
horizontal rectangle, *449*
Hough Transform, xxii, 62–63, 70
HV/VH tree, *476*, *See also* bucket PR-CIF k-d tree
hybrid k-d trie, *74*
hybrid tree, 101–102, 295, 465, 572–573, 585, 592, 610n

I

icosahedron, *231*
IER nearest neighbor algorithm. *See* Incremental Euclidean Restriction (IER) nearest neighbor algorithm
if, *744*
image, *196*
homogeneous. *See* homogeneous image
image processing, xv, xx, xxii, 423
image understanding, 424
image-based decomposition, *191*
image-based query. *See* location-based query
image-space methods, 183, 191
iMinMax method, 589–592, *589*, 854
window query, 591
implicit aggregation. *See* implicit representation
implicit representation, *195*
implicit winged-edge data structure. *See* oriented winged-edge data structure
INCFARTHEST, *507*, *835*
INCFARTHESTDUP, *507*, 835
inclusion set. *See* containment set
INCNEAREST, *494*
INCNEARESTDUP, *500*
INCNEARESTSPATIALNETWORK, *512*
Incremental Euclidean Restriction (IER) nearest neighbor algorithm, *509*, 514
incremental farthest neighbor finding, 502
vp-tree. *See* vp-tree
incremental nearest neighbor finding, 490–517, 833–835
AESA. *See* AESA
duplicate object instances, 499–502, 834–835
PM quadtree, 499
PMR quadtree, 500

incremental nearest neighbor finding *(continued)*
duplicate object instances *(continued)*
region quadtree, 499
R^+-tree, 499–500
general, 493–499, 833–834
r-optimal. *See* r-optimal algorithm
space requirements, 495–496
Geometric Near-neighbor Access Tree (GNAT). *See* Geometric Near-neighbor Access Tree (GNAT)
gh-tree. *See* gh-tree
kNN graph. *See* kNN graph
LAESA. *See* LAESA
M-tree. *See* M-tree
maximum distance, 502, 507
maximum result, 502, 507. *See also k* nearest neighbor finding
minimum distance, 502, 507
monotonous bisector tree (mb-tree). *See* monotonous bisector tree (mb-tree)
mvp-tree. *See* mvp-tree
R-tree. *See* R-tree
search hierarchy, 492–493, 666–667, 833
MX-CIF quadtree. *See* MX-CIF quadtree
R-tree. *See* R-tree
Spatial Approximation tree (sa-tree). *See* Spatial Approximation tree (sa-tree)
spatial network, 508–516, 835
Incremental Euclidean Restriction (IER) nearest neighbor algorithm. *See* Incremental Euclidean Restriction (IER) nearest neighbor algorithm
Incremental Network Expansion (INE) nearest neighbor algorithm. *See* Incremental Network Expansion (INE) nearest neighbor algorithm
TLAESA. *See* TLAESA
vp-tree. *See* vp-tree
vp^{sb}-tree. *See* vp^{sb}-tree
Incremental Network Expansion (INE) nearest neighbor algorithm, *513–514*
independent vertex, 238
index, xv
indexed-sequential file organization (ISAM), *717*
indirect uniform grid, *211*
INE nearest neighbor algorithm. *See* Incremental Network Expansion (INE) nearest neighbor algorithm
influence set, *658*
inherent dimensionality, *488*
initial network distance interval, *511*
inner approximation, *378*
inner box in balanced box-decomposition tree (BBD-tree), *83*
INRANGE, 17
INSERTAXIS, *829*
INSERTCHILDSOFNODE, *176*
INSERTL, *519*
INSERTNODE, *176*
INSERTQUADRANT, *35*
instantiation methods, 170–171, 183–184
integrated polytree, *371*
interface parameters in SP-GiST, 188
interior-based object representation, *192*, 193–312, 776–794
arbitrary objects, 254–264, 789–790
blocks, 204–232, 778–787
hierarchical, 264–312, 790–794
disjoint object-based, 304–312, 794
image-based, 266–270, 790
object-based, 270–303, 790–794
nonorthogonal blocks, 232–254, 787–789
unit-size cells, 194–204, 776–778
internal directory page, *103*
intersection query, *427*
intersection vertex, *514*
INTERSECTLINESQUARE, *368*
interval tree, xxii–xxiii, 434–439, 445–447, 473–474, 820–823
containment set, 446
definition as a Cartesian tree, 438
deletion, 437
dynamic. *See* dynamic interval tree
enclosure query, 439, 820
insertion, 437
one-dimensional containment query, 439, 820
primary structure. *See* primary structure
rectangle intersection problem, 244n, 434–439, 442, 460, 820–821
secondary structure. *See* secondary structure
spatial join query, 439, 820
stabbing counting query, 439, 820
stabbing query, 439, 820
tertiary structure. *See* tertiary structure
two-dimensional. *See* two-dimensional interval tree
window query, 444
inverse priority search tree, 190, 776, 821
intervals
one-dimensional containment query, 443, 821
points, *23*, 443
inverted file, 5, *5*, 37, 134
partial range query, 136
range query, 5, 136
inverted list, *5*, 37, 65, 91, 186–187, 189, 755, 776
IQ-tree, 592, 595–597
irregular cell-tree pyramid, 63
irregular grid, 210–211, *210*
irregular grid array pyramid, *269*
irregular grid bounding-box pyramid, *304*, 304–305
irregular grid pyramid, *269*, 304
irregularity in hierarchical face clustering, *397*
ISAM. *See* indexed-sequential file organization (ISAM)
isohedral tiling, *197*
isolated vertex, *368*
isometric embedding, *688*, *See also* distance-preserving embedding
ISPOINT, *180*
iterative end-point fit method. *See* Douglas-Peucker line generalization algorithm

J

Java, xx, 743
Johnson-Lindenstrauss Lemma, *715*, *871*
join
all nearest neighbor. *See* all nearest neighbor join (ANN join)
distance. *See* distance join
spatial. *See* spatial join query

K

k farthest neighbor finding
depth-first algorithm
basic, 520–521, 836
MINFARTHESTDIST, 548, 846
pruning rules, 529, 839
k nearest neighbor finding
AESA. *See* AESA
best-first algorithm, 502, 507
basic, 548–550
MAXNEARESTDIST, 550–552
space requirements, 548
depth-first algorithm, 517–557, 835–848
basic, 518–521, 835–836
MAXNEARESTDIST, 538–548, 845–847
point k-d tree. *See* point k-d tree
pruning rules, 521–529, 836–840
space requirements, 548
M-tree. *See* M-tree
k-center, *623*
approximate. *See* approximate *k*-center
k-center problem, 623
k-cut, *65*
k-cuttable, *65*, 70
k-d tree, xxii–xxiii, 11, 14, 37, 48–89, 761–768. *See also* point k-d tree
adaptive. *See* adaptive k-d tree
balanced four-dimensional. *See* balanced four-dimensional k-d tree
bucket PR. *See* bucket PR k-d tree
bucket PR-CIF. *See* bucket PR-CIF k-d tree
exact match query, 51
four-dimensional representative point rectangle representation, 462
generalized. *See* generalized k-d tree
minimum-ambiguity. *See* minimum-ambiguity k-d tree
MX. *See* MX k-d tree
MX-CIF. *See* MX-CIF k-d tree

open-sliding midpoint. *See* open-sliding midpoint k-d tree
path-compressed PR. *See* path-compressed PR k-d tree
path-level compressed PR. *See* path-level compressed PR k-d tree
peer-to-peer (P2P), 49
PK PR. *See* PK PR k-d tree
PMR. *See* PMR k-d tree
point. *See* point k-d tree
PR. *See* PR k-d tree
skip list. *See* skip list
sliding-midpoint. *See* sliding-midpoint k-d tree
spatial. *See* spatial k-d tree
superkey. *See* superkey
VAMSplit. *See* VAMSplit k-d tree
k^+-d tree
generalized . *See* generalized k^+-d tree
k-d trie, 49, *71*, 110, 112n, 142
bucket generalized. *See* bucket generalized k-d trie
generalized. *See* generalized k-d trie
hybrid. *See* hybrid k-d trie
k-d-B-tree, 12, 93, 96–103, 106, 110, 112, 112n, 114, 116, 127–130, 187, 295, 768
point page, *98*
region page, *98*
regions, 224, 306–311, *306*
resplitting. *See* k-d-B-tree resplitting
space requirements, 312, 794
k-d-B-tree resplitting, 309–310, *309*
k-d-B-trie, 12, 110–116, *110*, 129, 183–184, 187, 769–770
point page, *110*
range query, 112
region page, *110*
k-deinstantiated node, *170*
k-DOP, 195
k-instantiable node. *See* *k*-instantiated node
k-instantiated node, *168–170*
k-instantiation methods, 183
k-instantiation value, *164*
k-mean, 623
k-means problem, 533, 594
k-median problem, 623
K-structure, xxii, 69, 224, 237–242, 250, 347, 394, 787–788
construction algorithm, 238–239
dynamic edge insertion, 239
independent vertex. *See* independent vertex
K-vertex. *See* K-vertex
map overlay problem, 242, 788–789
neighborhood of a K-vertex. *See* neighborhood of a K-vertex
point location query, 239–241
visible vertex. *See* visible vertex
K-vertex, *238*, 241
independent. *See* independent vertex
neighborhood of. *See* neighborhood of a K-vertex
Karhunen-Loève Transform (KLT), 591, 671–673, 685–686, 690, 697–698, 704, 710
VA^+-file. *See* VA^+-file
kB-tree, 6, 8–9, 66n
exact match query, 136
partial range query, 9, 136
range query, 136
kB^+-tree, *9*
KD2-tree, 68, *68*, 464n, 610n
KD2B-tree, *68*, 68
KDCOMPARE, *761*
KDDELETE, *762*
KDDELETE1, *762*
KDINSERT, *761*
key, *1n*
primary retrieval. *See* primary key retrieval
secondary retrieval. *See* secondary key retrieval
KFARTHEST, 521, *836*
Klein bottle, 316, 353, 372n
KLT. *See* Karhunen-Loève Transform (KLT)
KNEAREST, *519*
kNN graph, 582, 599, 637–643, *637*, 658, 662, 859–862
extended neighborhood (EN), *638*
k nearest neighbor query, 641
best-first algorithm, 641
nearest neighbor query, 638–639, 714, 716
best-first algorithm, 639
incremental algorithm, 641
Monte Carlo algorithm, 711
range query, 641
reverse. *See* reverse kNN graph
seed, *638*
τ-closer, *638*

L

L field
rectangle record in MX-CIF quadtree, *470*
LAESA, 599, 601, 623, 643, 646–650, 650n, 862
incremental nearest neighbor query, 647
search hierarchy, 646, 650
lambda problem. *See* range query
Lambert's cylindrical projection, 232
landmark, *515*
Las Vegas randomized algorithm, *711*
approximate *k* nearest neighbor finding
SASH (Spatial Approximation Sample Hierarchy), 712
nearest neighbor query
Delaunay graph, 712
LATERALNEIGHBOR, *218*
lath, *330*
lath data structure
companion transition. *See* companion transition
corner lath data structure. *See* corner lath data structure
half-edge lath data structure. *See* split-vertex lath data structure
manifold objects, 331–336, 797
meshes with boundaries, 336–340, 797–799
primitive transition. *See* primitive transition
split-edge lath data structure. *See* split-face lath data structure
split-face lath data structure. *See* split-face lath data structure
split-vertex lath data structure. *See* split-vertex lath data structure
layered dag, xxii, 69, 224, 248–254, *248*, 347, 789
ldnode. *See ldnode*
map overlay problem, 254, 789
point location query, 250–254
three-dimensional point location query, 254
x-interval. *See* x-interval
layered tree, 823
LAYEREDDAGPOINTLOCATION, *252*
ldnode record
layered dag, *251*
LEAF value
NODETYPE field
node record in PM quadtree, *367*
least recently used (LRU) page replacement policy, *723*
least square quadtree, *360*
LEFT field
line record in PM quadtree, *367*
node record in one-dimensional range tree, *15*
node record in priority search tree, *21*
node record in two-dimensional range tree, *16*
TYPE field value vertex
ldnode record in layered dag, *251*
Lemma 2.1 of Feustel and Shapiro, *525*
Lemma 2.2 of Feustel and Shapiro, *528*, *See also* Pruning Rule 2 for $k = 1$
Lemma 4.1, *600*
Lemma 4.2, *601*
Lemma 4.3, *602*
Lemma 4.4, *603*
LEN field
square record in PM quadtree, *367*
level compression, *77*, 169, 183
level decompression, *79*
Level of Detail (LOD) methods, xix, *391*
Levenshtein edit distance, *488n*, 603
lexicographic ordering, *200*
LHRBC. *See* linear hashing with reversed bit concatenation (LHRBC)

LHRBI. *See* linear hashing with reversed bit interleaving (LHRBI)
limited tiling, *197*, 198
line intersection problem, 445
line quadtree, 356–357, *356*, 803
  revised. *See* revised line quadtree
line record
  PM quadtree, *367*
Linear Approximating and Eliminating Search Algorithm (LAESA). *See* LAESA
linear halfspace, *67n*
linear hashing, xxii–xxiii, 130–131, 137, 140–146, 152, 157–163, 187–188, 735, 739, 773–775
  asymptotic analysis, 734, 874
  basics, 729–734, 873–874
  merging, 731, 733, 874
  overflow bucket. *See* overflow bucket
  primary bucket. *See* primary bucket
  recursive. *See* recursive linear hashing
  reversed bit concatenation. *See* linear hashing with reversed bit concatenation (LHRBC)
  reversed bit interleaving. *See* linear hashing with reversed bit interleaving (LHRBI)
  splitting, *731*
  storage utilization factor. *See* storage utilization factor
linear hashing with partial expansions (LHPE), 140, 152, 774
linear hashing with reversed bit concatenation (LHRBC), *143*, 146n, 148, 150, 152
linear hashing with reversed bit interleaving (LHRBI), 142–146, *142*, 148, 150, 152, 160–162, 773, 775
  range query, 143, 146
linear homogeneity heuristic, *228*
linear least squares problem, 671
linear prefix, *415*, 415, 815
linear probing, 740
linear programming problem
  minimum bounding hyperspheres, 852
  Voronoi site reconstruction problem, 801
linear scale, *11*, 130, *131*, 186, 210
linear scan method. *See* sequential scan method
linear tree, *411*, 415, 814–815
link-cut tree, 253
Lipschitz embedding, 515, 608, 686, 690–697, *691*, 709–711, 716
  reference sets of the embedding. *See* reference sets
LISP, 743–744
list, *194n*
list, *743*
LITMAX, *92*
Lloyd's algorithm, 594
Local Split Decision tree (LSD tree). *See* LSD tree
locality sensitive hashing (LSH), 490, 664, 686
  approximate k nearest neighbor finding, 711–716, 871
    Monte Carlo randomized algorithm, 713
  collision set, *714*
localization, *389*, 392
location of a cell in space, *192*, 195
location parameters, *454*, 455
location query, *191*
location-based query, *192*
locational code, 72, 115, 166–167, 172, 183, 187, *214*, 255–256, 479, 510, 587
LOCHILD field
  node record in point k-d tree, *52*
LOD methods. *See* Level of Detail (LOD) methods
loop on a face, *316*
loop-edge relation, 317
loop-face relation, 317, 794
loose octree, 259
loose quadtree, 257–259, *257*, 475. *See also* cover fieldtree
Low variable
  layered dag, *252*
LRU. *See* least recently used (LRU) page replacement policy
LSD tree, 12, 61, 97, 102–106, 110, 110n, 129–130, 187, 585, 768
  collection of rectangles, 103n
  data-dependent strategy, *103*, 463
  distribution-dependent strategy, *103*, 463
  exact match query, 102
  four-dimensional representative point rectangle representation, 463
  paged. *See* paged LSD tree
  regions, 224, 308, 312
LSD^h tree, 592
LSH. *See* locality sensitive hashing (LSH)

M

M-tree, 529, 533, 540, 547, 551, 553, 557, 561, 592, 599, 609, 622, 624–629, 841, 845, 854
  covering radius, *624*
  incremental nearest neighbor query, 626–629
  k nearest neighbor query, 628–629
  parent object, *624*
  range query, 626
  routing object, *624*
  search hierarchy, 626–627
MAKENODEANDADDASCHILD, *175*
Manhattan distance, 32. *See also* City Block distance
manifold with boundary, *396*
mantissa of a real number, 2
map, *356*
map overlay problem, 241
  K-structure. *See* K-structure
  layered dag. *See* layered dag
  PM quadtree. *See* PM quadtree
mapping. *See* embedding
mark, *430*
MAT. *See* medial axis transformation (MAT); multiattribute tree
matrix manipulation
  MX k-d tree. *See* MX k-d tree
  MX quadtree. *See* MX quadtree
MAXDIST, *526*
  axis-aligned minimum bounding hyperrectangle, *840*
  SR-tree, *840*
MAXDISTINTER, *840*
MAXDISTOBJECT, *535*
MAXDISTSR, *840*
maximal block, *208*
maximal leaf node, *779*
maximum clique problem, *448*, 452, 776, 824
  region quadtree. *See* region quadtree
  segment tree. *See* segment tree
maximum distance query. *See* nearest neighbor query
maximum result query. *See* nearest neighbor query
MAXMINDIST, *535n*
MAXNEARESTDIST, *526*
  axis-aligned minimum bounding hyperrectangle, *840*
  minimum bounding hypersphere, *530*
  minimum bounding spherical shell, *531*
  SR-tree, *841*
MAXPRIORITYQUEUE, *519*
mb-tree, xxiii. *See* monotonous bisector tree (mb-tree)
mb*-tree. *See* monotonous bisector* tree (mb*-tree)
MD-tree, 98n
MDBT. *See* multidimensional B-tree (MDBT)
MDEH. *See* multidimensional extendible hashing (MDEH)
mean gray level, 424
mean square error, *690*
MEANORPIVOTDIST, *522*
measure problem, *428*, 447–453, *447*, 823–824
  multidimensions, 448–451
  segment tree. *See* segment tree
medial axis transformation (MAT), 6n, 208–210, *208*, 212, 364
medical image processing, 425
merging, 779
  linear hashing. *See* linear hashing
  spiral hashing. *See* spiral hashing
mesh, 400
  conforming. *See* conforming mesh
mesh generation, 425
metric, *488*
  Chamfer. *See* Chamfer metric
  City Block. *See* City Block distance

Euclidean. *See* Euclidean distance
Manhattan. *See* Manhattan distance
Minkowski. *See* Minkowski distance metric L_p
Octagonal. *See* Octagonal metric
metric space
finite. *See* finite metric space
metric tree, 598
ball partitioning. *See* vp-tree
generalized hyperplane partitioning. *See* gh-tree
MetricMap, 697, 711, *711*
pseudo-Euclidean distance. *See* pseudo-Euclidean distance
reference objects, *711*
Metro system, 392
MIDRANGE field
node record in one-dimensional range tree, *15*
node record in priority search tree, *21*
node record in two-dimensional range tree, *16*
MINDIST, *524*
axis-aligned minimum bounding hyperrectangle, *840*
SR-tree, *840*
MINDISTINTER, *840*
MINDISTOBJECT, *535*
MINDISTSR, *840*
MINFARTHESTDIST, *839*
axis-aligned minimum bounding hyperrectangle, *842*
minimum bounding hypersphere, *842*
minimum bounding spherical shell, *842*
SR-tree, *843*
minimal spanning tree (MST), 639, 641–643, 859, 861
nearest neighbor query, 643
minimax facilities location problem, 852
minimum angle property, *347*
minimum bounding box. *See* bounding box
minimum bounding n-corner, 195
minimum bounding object. *See* bounding box
minimum bounding polybox, 195
minimum distance query. *See* nearest neighbor query
minimum-ambiguity k-d tree, 58–61, 70
approximate nearest neighbor query, 70, 763
minimum-weight triangulation (MWT), *240*
Minkowski distance metric L_p, *64*
two dimensions, *209n*
MINMAXDIST, *535n*
MINNETWORKDISTBLOCK, *511*
MINPRIORITYQUEUE, *836*
MLOPLH. *See* multi-level order preserving linear hashing (MLOPLH)
mobility, 260
Möbius strip, 316, 353, 372
modified empty circle property, *351*
molecular tile, *197*
monotonous bisector tree (mb-tree), 580, 599, 618–622, *618*, 649, 856–857
incremental nearest neighbor query, 621
range query, 621
monotonous bisector* tree (mb*-tree), *618*
Monte Carlo randomized algorithm, *711*, 716
approximate *k* nearest neighbor finding
locality preserving hashing (LSH), 713
SASH (Spatial Approximation Sample Hierarchy), 712
nearest neighbor query
kNN graph, 711
Morton block, *215*, 510
Morton location ordering, 412–416, *412*, 816
Morton number, 93, *199*, *410*, 479
Morton order, 93, 150–152, 172, 199–203, *199*, 216, 256, 275, 277, 280, 280n, 296, 301, 303, 410, 413, 423, 469, 670, 776–779, 815, 818. *See also* N order; Z order
pruning property, 670, 865
Morton packed R-tree, *277*, 280
Morton size ordering, 410, *410*, 412–414, 814
most recently used (MRU) page replacement policy, *724n*
moving objects database, xvii, xix, 286, 445, 487, 567
MRU. *See* most recently used (MRU) page replacement policy
MST. *See* minimal spanning tree (MST)
multi-level order preserving linear hashing (MLOPLH), 162–163, *162*, 775
Multi-scale Line Tree, 384, *384*
multiattribute tree, 6–7, *6*, 9, 208n
multibody problem, 426. *See also N*-body problem
multicolored quadtree, *211*
multidimensional B-tree (MDBT), 6, 8–9, *8*, 14, 66, 748
exact match query, 136
partial range query, 8–9, 14, 136, 748
range query, 8, 136, 748
multidimensional directory (MDD), 9–10, *9*, 66
hybrid variant, *10*
multidimensional extendible hashing (MDEH), 140, 146–150, 152–153, 153n, 155–156, 159–162, 774–775
exact match query, 150
partial expansion, 152
range query, 150–151
multidimensional histogramming, 62, 269
multidimensional measure problem
region quadtree. *See* region quadtree
trellis. *See* trellis
multidimensional order-preserving linear hashing with partial expansion (MOLHPE), *152*, 152
multidimensional range tree, 5–6, 9, 14, *18*, 187, 190, 776
construction cost, 18, 750
range query, 18, 750
space requirements, 18, 187, 750
multidimensional scaling (MDS), 689–690
multidimensional searching, xix, xxii
multidimensional trie, *3*
multidistance function, *640*
multilayer grid file, *256*, 256
multilevel grid file, 97, 110, 113–116, 115n, 465
multilist, *5*
multimedia database, xv–xvii, xix, xxi–xxii
multiobject BSP tree, *233*
multipaging, 10–11, *10*, 130, 136–137, *136*
axial array. *See* axial array
dynamic. *See* dynamic multipaging
static. *See* static multipaging
multiple coverage problem, 128
multiple posting problem, *108*, 125, 128
multiple resolution, 255, 424
multiple storage quadtree, *481*
multiresolution. *See* multiple resolution
multiresolution model, 391, 394, 399
continuous. *See* continuous multiresolution model
discrete. *See* discrete multiresolution model
multiresolution radiosity, 398
multiresolution representation, *266*
multitriangulation, 399
multiway tree, 8, 97–98, *717*
mutual intersection rule, *261*, 261, 261n
mvp-tree, 608–609, *608*, 611, 613, 856
incremental nearest neighbor query, 609
range query, 609
MWT. *See* minimum-weight triangulation (MWT)
MX bintree, *72*
MX k-d tree
points, 72, 182
matrix manipulation, 72
MX octree, xxiii
regions, *357*, 361, 363, 369–370, 803
MX quadtree, xxiii, 369
PK-tree. *See* PK MX quadtree
points, 11, 38–42, *38*, 72, 172, 174–175, 180, 182, 186, 188–189, 473, 756–758, 776
collapsing, *40*
comparison with point quadtree, 47–48, 761
comparison with PR quadtree, 47–48, 761
deletion, 40–41, 756–757
insertion, 39–41, 756
matrix manipulation, 40, 42, 72, 757–758
range query, 40–42, 757
space requirements, 48
splitting, *40*

MX quadtree *(continued)*
regions, 40, 357–360, *357*, 803
space requirements, 357–359, 376, 803
MX quadtrie, *38*
MX-CIF k-d tree, 475–476
MX-CIF quadtree, xxiii, 115, 255–257, 259, 466–476, *466*, 478–479, 481, 827–830, 832
axes. *See* axes
axis lines. *See* axis lines
collapsing, *468*
construction cost, 481, 483
deletion, 468–471, 830
expanded. *See* expanded MX-CIF quadtree
insertion, 468, 470–471, 829, 832
one-dimensional. *See* one-dimensional MX-CIF quadtree
overlap test, 470, 828–829
overlay query, 469
peer-to-peer (P2P), 259, 517
range query, 469, 472
rectangle intersection problem, 481, 483
expected cost, 473
plane-sweep methods, 469, 473
tree traversal, 469
search hierarchy, 493, 833
space requirements, 481, 483
splitting, *468*
window query, 469, 472, 474, 480
MXCOMPARE, *756*
MXDELETE, *756*
MXINSERT, *756*

N

N order, *200*, *See also* Morton order; Z order
N-body problem, 59, 426. *See also* multibody problem
N-tree, *93*
NAME field
rectangle record in MX-CIF quadtree, *470*
natural correspondence between a forest and a binary tree, 7, 722, *722n*
nearest common ancestor, *435*
nearest foreign neighbor problem, *577*
nearest neighbor query, 192n. *See also* post office problem
approximate. *See* approximate nearest neighbor query
augmented Gabriel graph (AGG). *See* augmented Gabriel graph (AGG)
best-first incremental algorithm. *See* incremental nearest neighbor finding
best-first *k* nearest neighbors. *See k* nearest neighbor finding
Delaunay graph. *See* Delaunay graph
depth-first *k* nearest neighbors. *See k* nearest neighbor finding
ε. *See* ε-nearest neighbor
feature, *485*
Gabriel graph (GG). *See* Gabriel graph (GG)
incremental algorithm. *See* incremental nearest neighbor finding
k nearest neighbors. *See k* nearest neighbor finding
kNN graph. *See* kNN graph
minimal spanning tree (MST). *See* minimal spanning tree (MST)
partial match. *See* partial match nearest neighbor query
probably approximately correct (PAC). *See* probably approximately correct (PAC) nearest neighbor query
relative neighborhood graph (RNG). *See* relative neighborhood graph (RNG)
reverse. *See* reverse nearest neighbor query (RNN)
R*-tree. *See* R*-tree
sequential scan method. *See* sequential scan method
vector approximation file (VA-file). *See* vector approximation file (VA-file)
X-tree. *See* X-tree
nearest object query, *192*, *See* nearest neighbor query
ordered R-tree. *See* ordered R-tree
R-tree. *See* R-tree
neighbor finding in a region quadtree
diagonal neighbor, 220, 781–782
lateral neighbor, 217
neighborhood of a K-vertex, *238*
nested interpolation-based grid file (NIBGF), 115, *115*, 479
exact match query, 116, 769
network distance, *509*
network distance interval, *511*
NEWROOT, 504, *754*
NEXT field
edgelist record in PM quadtree, *368*
node record in one-dimensional range tree, *15*
NEXTVERTEXSHORTESTPATH, *511*
node
directory. *See* directory node
gray. *See* gray node
k-deinstantiated. *See k*-deinstantiated node
k-instantiable. *See k*-instantiated node
k-instantiated. *See k*-instantiated node
nonleaf. *See* nonleaf node
separator. *See* separator node
node record
MX quadtree for points, *41*
one-dimensional range tree, *15*
PM quadtree, *367*
point k-d tree, *52*
point quadtree, *30*
PR quadtree, *46*
priority search tree, *21*
two-dimensional range tree, *16*
NODETYPE field
node record in MX quadtree for points, *41*
node record in PM quadtree, *367*
node record in PR quadtree, *46*
non-oriented winged-edge data structure, *324*
nondegenerate Delaunay triangulation, *347*
nondegenerate Voronoi diagram, *347*, 367
nonhomogeneous k-d tree, 58n
nonleaf node, *214*
nonmanifold object
half-edge data structure. *See* half-edge data structure
partial-entity structure. *See* partial-entity structure
radial-edge data structure. *See* radial-edge data structure
nonpolygonal tiling, 377–380
nonprimitive topological entities, 316
NP problem, *206n*
NP-complete problem, *206n*
Steiner point tetrahedralization. *See* Steiner point tetrahedralization problem
traveling salesman. *See* traveling salesman problem

O

OBB. *See* oriented bounding box (OBB)
OBBTree, 195, *386*, 389, 393, 398
object aggregation techniques, 271–272, 274–275
extent-based. *See* extent-based object aggregation techniques
ordering-based. *See* ordering-based object aggregation techniques
object B^+-tree, *279*, 279–280, 283, 301
object graph, *316*
object hierarchy, 189
object number, *275*
object pyramid, *272*, 272, 274, 304
object representation
boundary-based. *See* boundary-based object representation
interior-based. *See* interior-based object representation
object-based decomposition, *191*
object-based query. *See* feature-based query
object-space methods, 183, 191
object-tree pyramid, *272*, 273–274, 275n, 276–279, 282–283, 294, 296, 298–299, 301, 304
bulk insertion, 296–299, *297*
bulk loading, *297*, 299–301, 794
Octagonal metric, 210
octahedron, *232*
octree, *211*, 393
extended. *See* extended octree
face. *See* face octree
MX. *See* MX octree

regions. *See* MX octree
PM. *See* PM octree
PM_1. *See* PM_1 octree
PM_2. *See* PM_2 octree
PM_3. *See* PM_3 octree
region. *See* region octree
vector. *See* vector octree
OLAP. *See* online analytic processing (OLAP)
Ω, *xx*
one-cut, *65*
one-dimensional bucket PR-CIF k-d tree, 476–478, 480, 482
one-dimensional containment query
interval tree. *See* interval tree
inverse priority search tree
intervals. *See* inverse priority search tree
priority search tree
intervals. *See* priority search tree
one-dimensional MX-CIF quadtree, 255, 467, 470, 473–475
comparison, 470–471, 828, 832
one-dimensional ordering, 90–95, 187, 768
one-dimensional partial overlap query
priority search tree
intervals. *See* priority search tree
one-dimensional point quadtree, 474–475
one-dimensional PR quadtree, 474–475
one-dimensional R-file, 478
one-dimensional range tree, 5, 9, *14*, 19–20, 433, 440–441, 444–446, 821
range query, 14–15
rectangle intersection problem, 433
online analytic processing (OLAP), 297
OPDIRECTION, *470*
open-sliding midpoint k-d tree, 74, *74*, 766
space requirements, 74, 77, 764–766
OPLH. *See* order preserving linear hashing (OPLH)
OPQUAD, *32*
OPTBFTRAV, *550*
OPTBFTRAVFARTHEST, *553*, *847*
OPTDFTRAV, *541*
OPTDFTRAVFARTHEST, *548*, 846
optimized k-d tree, 58, 65–66, 69, 763
deletion, 65
range query, 65
optimized point quadtree, *30*, 30–31, 58, 65, 190, 752–753, 776
construction cost, 31, 752
total path length (TPL), 31, 752
OPTINSERTL, *542*
order
admissible. *See* admissible order
B-tree. *See* B-tree
B^+-tree. *See* B^+-tree
Cantor-diagonal. *See* Cantor-diagonal order
column. *See* column order
double Gray. *See* double Gray order
Gray. *See* Gray order
Morton. *See* Morton order
N. *See* N order
Peano-Hilbert. *See* Peano-Hilbert order
R-tree. *See* R-tree
row. *See* row order
row-prime. *See* row-prime order
spiral. *See* spiral order
stable. *See* stable order
U. *See* U order
Z. *See* Z order
order preserving linear hashing (OPLH), *142*, 148, 150, 152–153, 153n, 162
partial expansion, 152
ordered R-tree, *301*, 303
nearest object query, 303
point query, 303
rectangle intersection problem, 303
window query, 303
ordered R-tree node-splitting policies
seed-picking
linear cost, 303
quadratic cost, 303
ordering
partial. *See* partial ordering
total. *See* total ordering
ordering-based object aggregation techniques, 275–281, 790–791
representative point, 275
ordinary hierarchy, *197*
organization method
data. *See* data organization methods
embedding space. *See* embedding space organization methods
ORIENT field
edge_orientation_pair record in ORIENTEDFACEEDGETABLE and ORIENTEDVERTEXEDGETABLE, *326*
oriented bounding box (OBB), *195*
oriented bounding box tree (OBBTree). *See* OBBTree
oriented edge, *324*
oriented winged-edge data structure, 324–329, *325*, 331, 343–344
ORIENTEDFACEEDGETABLE, *325*
ORIENTEDVERTEXEDGETABLE, *325*
orphan object, *652*, 659n, 661–662
orthogonal partition tree, *450*, 452
orthogonal polygon, *409n*
os-tree. *See* overlapped-split tree (os-tree)
OTHERAXIS, *470*
outer approximation, *378*
outer box in balanced box-decomposition tree (BBD-tree), *83*
OUTOFRANGEY, *22*
overflow bucket, *130*, *731*
overflow file, *733*
overlap, *274*
overlapped-split tree (os-tree), 68, 573–575, 610n
cover, *573*
failure probability, *575*
probably correct. *See* probably correct os-tree
overlapping pyramid, 257, *424*
overlay query
MX-CIF quadtree. *See* MX-CIF quadtree
PR quadtree. *See* PR quadtree
oversized shelf, *567*
overwork, *37*, 37, 755

P

P-tree, 195, 570
p^+-order, *151*, 151–152
P1 field
line record in PM quadtree, *367*
P2 field
line record in PM quadtree, *367*
PAC. *See* probably approximately correct (PAC) nearest neighbor query
packed R-tree, *277*, 277–278, 281, 296–297, 301
Hilbert. *See* Hilbert packed R-tree
Morton. *See* Morton packed R-tree
point query, 296
window query, 296
page, *3*, *12*
buddy. *See* buddy page
directory. *See* directory page
external directory. *See* external directory page
internal directory. *See* internal directory page
point. *See* point page
region. *See* region page
page fault, 717
paged LSD tree, *105*
parallel data structure, 49
parallel processing, xviii
parametric R-tree, *286*, 445
Pareto operator, *503*
Parseval's theorem, *680*
partial expansion
multidimensional extendible hashing (MDEH). *See* multidimensional extendible hashing (MDEH)
order preserving linear hashing (OPLH). *See* order preserving linear hashing (OPLH)
partial match nearest neighbor query, *844*
partial match query. *See* partial range query
partial ordering, 506, *506n*
skyline query. *See* skyline query
partial range query, *3*
BD-tree. *See* BD-tree
bit interleaving. *See* bit interleaving
combined index. *See* combined index
double Gray order. *See* double Gray order
Gray order. *See* Gray order
grid file. *See* grid file
inverted file. *See* inverted file
kB-tree. *See* kB-tree

partial range query *(continued)*
multidimensional B-tree (MDBT). *See* multidimensional B-tree (MDBT)
point k-d tree. *See* point k-d tree
point quadtree. *See* point quadtree
PR k-d tree. *See* PR k-d tree
PR quadtree. *See* PR quadtree
quintary tree. *See* quintary tree
U order. *See* U order
partial-entity structure, 316
partition fieldtree, 257–259, *257*, 401, 475, 480
partition number, *146*
partition process, *165*
partitioning box in balanced box-decomposition tree (BBD-tree), *83*
PASCAL, xx, 743
path compression, *76*, 169, 182–183
path planning, 373
path-compressed PR k-d tree, 75–82, *76*, 169, 182, 188, 581, 619, 765–766
space requirements, 76–77, 766
path-level compressed PR k-d tree, 77–79, *78*, 82, 165, 169, 182, 767
path-level compression, *78*
Patricia trie, *76*, 183, 188
pattern matching, 425
pattern recognition, xv, xx, 423
PCA. *See* Principal Component Analysis (PCA)
Peano-Hilbert location ordering, 415–416
Peano-Hilbert order, 145, 151–152, 172, 199–202, *199*, 216, 275–277, 280–281, 296–297, 301, 303, 410, 423, 670, 768, 774, 776–779, 793, 815
Peano-Hilbert size ordering, *410*, 414
peer-to-peer (P2P)
gh-tree. *See* gh-tree
k-d tree. *See* k-d tree
MX-CIF quadtree. *See* MX-CIF quadtree
R-tree. *See* R-tree
perfect point quadtree, 37, 57, 755
perimeter computation problem
segment tree. *See* segment tree
persistent search trees, 445
phasing, 46, *46*
PHYSICAL, *874*
piano movers problem, 425
pick query in computer graphics, 4, 192, 486. *See also* nearest neighbor query; nearest object query
Piecewise Linear Complex (PLC), *352*, 352n
piecewise linear order preserving (PLOP) hashing, 146, 153–156, 159, 161, 187–188, 774
pivot, *521*, *598*, *See also* pivot object
pivot object, *598*, *See also* pivot
pivot-based distance-based indexing methods, 598–599, 625
pixel, *192*
PK MX quadtree, *172*, 172
PK point k-d tree, 184, 775
PK point quadtree, 184, 775
PK PR k-d tree, *168*, 169–171, 183
PK PR octree, *170*
PK PR quadtree, *168*, 168–170, 173–174, 177–178, 181
PK pseudo k-d tree, 775
PK pseudo quadtree, 775
PK-tree, xxiii, 12, 117, 127n, 164–185, 190, 216, 775
deletion, 176, 178–181, 775
directory node, 117, *167*
insertion, 172–178
MX quadtree. *See* PK MX quadtree
point k-d tree. *See* PK point k-d tree
point node, 117, *167*
point quadtree. *See* PK point quadtree
PR k-d tree. *See* PK PR quadtree
PR octree. *See* PK PR octree
PR quadtree. *See* PK PR quadtree
pseudo k-d tree. *See* PK pseudo k-d tree
pseudo quadtree. *See* PK pseudo quadtree
space requirements, 183
PKDELETE, *180*
PKINSERT, *176*
plan, 424
planar graph, *315*
adjacent faces, *315*
face, *315*
frontier of a face, *315*
Planar Straight Line Graph (PSLG), *351*, 352, 352n
plane-sweep methods, xxiii, 291, 428–453, *428*, 819–824
skyline query. *See* skyline query
plane-sweep triangulation, 238
Platonic solids, *232*
PLC. *See* Piecewise Linear Complex (PLC)
PLOP hashing. *See* piecewise linear order preserving (PLOP) hashing
PM octree, xxiii, 363, 369–374, *369*, 808–809, *809*
black hole, 371
Boolean node, *372*
Boolean set operations, 370–372
conversion between boundary model, 370
nasty node, *371*
nearby node, *372*
space requirements, 373
terminal gray node, *371*
PM_1 octree, *369*, 371–373, 808
point location query, 374
space requirements, 373, 809
PM_2 octree, 371–373, *371*
black hole, 372
PM_3 octree, *371*, 372
PM quadtree, xxiii, 363, 365–369, 481, 584, 801, 803–808
bucket. *See* bucket PM quadtree
bucket rectangle. *See* bucket rectangle PM quadtree
edge-based, 365, 375
map overlay problem, 369
vertex-based, 365–366, 374
PM_1 quadtree, xxiii, *365*, 365–369, 372–377, 513, 516, 809–810
bucket. *See* bucket PM_1 quadtree
deletion, 368–369, 805–806
insertion, 368, 803–805
point location query, 369, 808–809
space requirements, 376
vertex-based, 366
PM_2 quadtree, xxiii, *366*, 367, 369, 371, 375–377, 584–585, 801, 810, 853
bucket. *See* bucket PM_2 quadtree
deletion, 369, 807
insertion, 369, 806–807
vertex-based, 366
PM_3 quadtree, xxiii, *366*, 367, 370–372, 375–376, 809–810
bucket. *See* bucket PM_3 quadtree
deletion, 369, 808
insertion, 369, 807–808
vertex-based, 366
PM-CSG tree, *373*
PM1CHECK, *804*
PM2CHECK, *806*
PM3CHECK, *807*
PMDELETE, *805*
PMINSERT, *803*
PMR k-d tree, xxiii, 74–75, 182–183
PMR polygon quadtree, *262*, 262
PMR quadtree, xxiii, 74n, 182–183, 262, 366, 369, 374–377, *375*, 513, 516
deletion, 377
insertion, 377
polygon. *See* PMR polygon quadtree
space requirements, 376
PMR rectangle k-d tree, xxiii
PMR rectangle quadtree, xxiii. *See also* PMR rectangle quadtree
point bucket, *97*
point dominance problem, 88, 444. *See also* dominance relation
POINT field
node record in priority search tree, *21*
node record in two-dimensional range tree, *17*
point k-d tree, xxiii, 49–72, *50*, 88, 96, 98, 101, 183, 186, 189, 418, 576, 585, 761–764, 776
bounds array. *See* bounds array
bucket adaptive. *See* bucket adaptive k-d tree
bucket generalized pseudo. *See* bucket generalized pseudo k-d tree
deletion, 52–55, 65, 762–763
analogy to binary search tree deletion, 52–53
depth-first k nearest neighbor algorithm, 538, 844

divided. *See* divided k-d tree
double rotation, 52
generalized pseudo. *See* generalized pseudo k-d tree
homogeneous. *See* homogeneous k-d tree
insertion, 50–52, 761
MX-CIF. *See* MX-CIF k-d tree
nonhomogeneous. *See* nonhomogeneous k-d tree
optimized. *See* optimized k-d tree
partial range query, 56–57, 72, 763–764
PK-tree. *See* PK point k-d tree
pseudo. *See* pseudo k-d tree
range query, 55–57, 65, 763
regions, 223–225, 260, 305, 314, 783
search, 55–57, 763
single rotation, 52
spatial. *See* spatial k-d tree
total path length (TPL), 52–53, 58, 65, 762
trie-based. *See* trie-based k-d tree
VAMSplit k-d tree. *See* VAMSplit k-d tree
point location problem. *See* point location query
point location query, 192, 435n, 489
fractional cascading. *See* fractional cascading
gap tree. *See* gap tree
K-structure. *See* K-structure
layered dag. *See* layered dag
PM_1 quadtree. *See* PM_1 quadtree
R-tree. *See* R-tree
separating chain. *See* gap tree
point node
PK-tree. *See* PK-tree
point page
BV-tree. *See* BV-tree
k-d-B-tree, 103. *See also* k-d-B-tree
k-d-B-trie. *See* k-d-B-trie
point quadtree, xxiii, 11, 14, 28–37, 57, 70, 88, 96, 164, 183, 186–189, 269, 418, 425, 751–755, 775–776
comparison with MX quadtree, 47–48, 761
comparison with PR quadtree, 47–48, 761
construction cost, 31, 753
deletion, 31–36, 504, 753–754
analogy to binary search tree deletion, 31
double rotation, 31, 753
insertion, 28–31, 751–752
one-dimensional. *See* one-dimensional point quadtree
optimized. *See* optimized point quadtree
partial range query, 37, 48, 755, 761
PK-tree. *See* PK point quadtree
pseudo. *See* pseudo quadtree
range query, 36–37, 755
region query, 37
regions, 222–223, 260, 305, 782–783
single rotation, 31, 753
space requirements, 48
total path length (TPL), 30–31, 34–35, 755
point quadtree bounding-box cell-tree pyramid, *306*
point query
feature, *485*
objects, *428*
balanced four-dimensional k-d tree. *See* balanced four-dimensional k-d tree
corner stitching. *See* corner stitching
ordered R-tree. *See* ordered R-tree
packed R-tree. *See* packed R-tree
R-file. *See* R-file
R-tree. *See* R-tree
representative point rectangle representation. *See* representative point rectangle representation
R^+-tree. *See* R^+-tree
points, 192, 428. *See* exact match query
two-dimensional representative point rectangle representation
PR k-d tree. *See* PR k-d tree
PR quadtree. *See* PR quadtree
spatial k-d tree. *See* spatial k-d tree
point record
layered dag, *252*
PM quadtree, *367*
two-dimensional range tree, *17*
point set query, *428*
pointer, *743*
polar coordinates, 377–380, 740
polar quadtree, 379–380, *379*
pole, 231–232
polygon
convex. *See* convex polygon
polygon quadtree
bucket. *See* bucket polygon quadtree
PMR. *See* PMR polygon quadtree
vertex bucket. *See* vertex bucket polygon quadtree
polygon representation, *313*
polygon tree, *36*, 37
polygonal map, *261*, 356
regularization. *See* regularization
y-monotone subdivision. *See* *y*-monotone subdivision
polygonal tiling, 197–199, 776
polyhedral approximation, 362, 386–389, 813
adjustment step, 387–388, 813
polyline, *68*, 224, *313*
polysurface, *390*
polytree, *370*
population, *45*
population model, 45–46
POSSIBLEPM1MERGE, *806*
POSSIBLEPM23MERGE, *807*
post office problem, *4*, 486. *See also* nearest neighbor query
post office tree, *609*
PR bintree, *71*
PR k-d tree, xxiii, 70–74, *71*, 76, 80, 82–84, 110, 112–114, 168–171, 182–183, 186, 189, 619, 764, 776
bucket. *See* bucket PR k-d tree
partial range query, 72, 764
path-compressed. *See* path-compressed PR k-d tree
path-level compressed. *See* path-level compressed PR k-d tree
PK-tree. *See* PK PR k-d tree
regions, *221*
two-dimensional representative point rectangle representation, 464
point query, 464
PR octree, 170
PK-tree. *See* PK PR octree
PR quadtree, xxiii, 11, 38, 42–47, *42*, 70–71, 80, 163, 166, 168–173, 175, 178, 180–183, 186, 188–189, 366, *368*, 511, 580, 599, 619, 758–761, 776
bucket. *See* bucket PR quadtree
collapsing, *43*
comparison with MX quadtree, 47–48, 761
comparison with point quadtree, 47–48, 761
deletion, 43, 47, 759–760
four-dimensional representative point rectangle representation, 458
insertion, 43, 47, 758–759
one-dimensional. *See* one-dimensional PR quadtree
overlay query, 45
partial range query, 48, 761
PK-tree. *See* PK PR quadtree
range query, 44, 47, 761
skip list. *See* skip list
space requirements, 44, 47–48, 760
splitting, *43*
triangulation, 241
two-dimensional representative point rectangle representation, 458, 464
point query, 464
PR quadtrie, *42*, 163
PR-file, 384
PRCOMPARE, *758*
PRDELETE, *759*
precision, 664, *664*
predictive region quadtree construction. *See* region quadtree
preference function, *505*
prefix B^+-tree, 415, *726*, 815
preload, *743*
preprocessing cone, 424
PREV field
node record in one-dimensional range tree, *15*
primary bucket, *731*
primary key, 91

primary key retrieval, *1n*, 5
primary structure, 435, *435*
  active node, *436*
  inactive node, *436*
prime meridian, 232
primitive topological entities, 316
primitive transition, *331*
principal axis, *66*
principal axis tree (PAT), *9*, 66
Principal Component Analysis (PCA), 398, 591, 671–673, 685–686, 690, 697–698, 704, 710
  probably correct os-tree. *See* probably correct os-tree
PRINSERT, *758*
priority queue, 20, 493
priority R-tree, 286n
priority search tree, xxii–xxiii, 6, 9, 13, 19–27, *19*, 187, 189–190, 439–443, 447, 750–751, 776, 821
  containment set, 446
  intervals, xxiii, 435n, 439–443, 821
    enclosure query, 443, 821
    one-dimensional containment query, 443, 821
    one-dimensional partial overlap query, 443, 821
    rectangle intersection problem, 439–443, 821
    stabbing query, 443, 821
  inverse. *See* inverse priority search tree
  points, xxiii, 19–27, 440, 750–751
    construction cost, 19, 26, 750
    semi-infinite range query, 20–23, 25–26, 440, 750
    space requirements, 19, 25, 750
    two-dimensional range query, 19
  two-dimensional. *See* two-dimensional priority search tree
PRIORITYSEARCH, *21*
PRIORITYSEARCH2, *26*
prism tree, 362, 386–389, *387*, 391, 393, 398, 401–402, 813
  object intersection, 389
  regular. *See* regular prism tree
probably approximately correct (PAC) nearest neighbor query, 561–563, *562*, 579, 596–597
probably correct os-tree, 574–580, *575*, 853
  Principal Component Analysis (PCA), 575–576, 580
  support vector machines (SVM) method, 576–577, 579
  support vectors, *576*
projection space, *712*
property sphere, *231*
proximity query, 4
proximity search, 36
proximity-preserving embedding, 686–687
proximity-preserving property, 665, *665*, 667, 687, 864–865
pruning property, 664–665, *664*, 667. *See also* contractive embedding; contractive mapping
  space-ordering methods. *See* space-ordering methods
Pruning Rule 1
  k farthest neighbors, 529, *838*
  k nearest neighbors, *524*, *See also* Lemma 4.2
Pruning Rule 2
  k farthest neighbors
    nonobjects, 548, *846*
    objects, 529, *838*
  k nearest neighbors. *See also* Lemma 4.1
    nonobjects, *540*
    objects, *524*
Pruning Rule 3
  k farthest neighbors, 529, *838*
  k nearest neighbors, *524*, *See also* Lemma 4.2
Pruning Rule 4
  k farthest neighbors
    nonobjects, 548, *846*
    objects, 529, *838*
  k nearest neighbors. *See also* Lemma 4.1
    nonobjects, *540*
    objects, *524*
Pruning Rule 5
  farthest neighbor, 529, *839*
  k nearest neighbors, 541, *541*, 543–546
  nearest neighbor, *526*, *See also* Lemma 4.2
pseudo k-d tree, *58*, 65–66, 69
  PK-tree. *See* PK pseudo k-d tree
pseudo quadtree, 34, *34*, 36, 58, 65, 69, 88, 190, 755, 775–776
  PK-tree. *See* PK pseudo quadtree
pseudo-Euclidean distance, *711*
PSLG. *See* Planar Straight Line Graph (PSLG)
PTCOMPARE, *751*
PTDELETE, *753*
PTINSERT, *752*
PTINSQUARE, *368*
puzzletree, 66, 225–230, *225*, 260, 572, 783–786. *See also* treemap; X-Y tree
pyramid, xvi, xx, 63, 266–270, 384, *424*, 776, 790
  array. *See* array pyramid
  bintree bounding-box cell-tree. *See* bintree bounding-box cell-tree pyramid
  bounding-box cell-tree. *See* bounding-box cell-tree pyramid
  cell. *See* cell pyramid
  cell-tree. *See* cell-tree pyramid
  disjoint object. *See* disjoint object pyramid
  generalized k-d tree bounding-box cell-tree. *See* generalized k-d tree bounding-box cell-tree pyramid
  irregular grid. *See* irregular grid pyramid
  irregular grid array. *See* irregular grid array pyramid
  irregular grid bounding-box. *See* irregular grid bounding-box pyramid
  object. *See* object pyramid
  overlapping. *See* overlapping pyramid
  point quadtree bounding-box cell-tree. *See* point quadtree bounding-box cell-tree pyramid
  region representation, 424–425
  truncated-tree. *See* truncated-tree pyramid
pyramid technique, 587–592, *587*, 854
  extended. *See* extended pyramid technique
  pyramid value, *588*
  window query, 591

Q

q-edge, *366*
q-face, *366*, *370*
Q-tree, 423
QLQT. *See* quad list quadtree (QLQT)
QMAT. *See* quadtree medial axis transform (QMAT)
QSF-tree
  simple. *See* simple QSF-tree
quad list quadtree (QLQT), 482–483
  construction cost, 483
  space requirements, 477–478, 482–483
  window query, 477–478, 482–483
quad-edge data structure, 322–324, *322*, 329–330, 341–343, 345, 353–354
quadgrid, 165, *165*, 170
quadtree, xviii, *211*, 423
  0-balanced. *See* 0-balanced quadtree
  1-balanced. *See* 1-balanced quadtree
  1-irregular. *See* 1-irregular quadtree
  adaptive. *See* adaptive quadtree
  balanced. *See* balanced quadtree
  binary. *See* binary quadtree
  bounded. *See* bounded quadtree (BQT)
  bucket PM. *See* bucket PM quadtree
  bucket PM_1. *See* bucket PM_1 quadtree
  bucket PM_2. *See* bucket PM_2 quadtree
  bucket PM_3. *See* bucket PM_3 quadtree
  bucket polygon. *See* bucket polygon quadtree
  bucket pseudo. *See* bucket pseudo quadtree
  bucket rectangle. *See* bucket rectangle PM quadtree
  bucket rectangle PM. *See* bucket rectangle PM quadtree
  edge. *See* edge quadtree
  expanded MX-CIF. *See* expanded MX-CIF quadtree
  hinted. *See* hinted quadtree
  least square. *See* least square quadtree
  line. *See* line quadtree
  multicolored. *See* multicolored quadtree

multiple storage. *See* multiple storage quadtree
MX. *See* MX quadtree
MX-CIF. *See* MX-CIF quadtree
expanded. *See* expanded MX-CIF quadtree
PK MX. *See* PK MX quadtree
PK point. *See* PK point quadtree
PK PR. *See* PK PR quadtree
PK pseudo. *See* PK pseudo quadtree
PM. *See* PM quadtree
PM_1. *See* PM_1 quadtree
PM_2. *See* PM_2 quadtree
PM_3. *See* PM_3 quadtree
PMR. *See* PMR quadtree
PMR polygon. *See* PMR polygon quadtree
point. *See* point quadtree
polar. *See* polar quadtree
PR. *See* PR quadtree
pseudo. *See* pseudo quadtree
quad list (QLQT). *See* quad list quadtree (QLQT)
rectangle. *See* rectangle quadtree
region. *See* region quadtree
restricted. *See* restricted quadtree
revised line. *See* revised line quadtree
shortest path. *See* shortest-path quadtree
trie-based. *See* trie-based quadtree
uniform. *See* uniform quadtree
vector. *See* vector quadtree
vertex bucket polygon. *See* vertex bucket polygon quadtree
Quadtree Complexity Theorem, 357–359, *358*, 376, 803
quadtree medial axis transform (QMAT), 212, *212*, 220, 257, 365, 365n, 425, 782
quadtree skeleton, *212*, 220, 782
quadtrie, 423
MX. *See* MX quadtrie
PR. *See* PR quadtrie
quantile hashing, 153–156, 155n, 159, 161, 187–188, 774
quaternary hierarchical decomposition, *403*
rectangle. *See* quaternary hierarchical rectangular decomposition
triangle. *See* quaternary hierarchical triangular decomposition
quaternary hierarchical rectangular decomposition, 402–405, 408
quaternary hierarchical triangular decomposition, *400*, 402–405, 407
query
all closest pairs. *See* all closest pairs query
Boolean. *See* Boolean query
containment. *See* containment query
containment set. *See* containment set
enclosure. *See* enclosure query
epsilon. *See* epsilon query
exact match. *See* exact match query
farthest neighbor. *See* farthest neighbor query
feature. *See* feature query; feature-based query
geometric join. *See* geometric join query
intersection. *See* intersection query
location. *See* location query; location-based query
maximum distance. *See* nearest neighbor query
maximum result. *See* nearest neighbor query
measure problem. *See* measure problem
minimum distance. *See* nearest neighbor query
nearest neighbor. *See* nearest neighbor query
nearest object. *See* nearest object query
partial match. *See* partial range query
partial match nearest neighbor. *See* partial match nearest neighbor query
partial range. *See* partial range query
pick in computer graphics. *See* nearest object query
point
objects. *See* point query
points. *See* exact match query
point set. *See* point set query
probably approximately correct (PAC) nearest neighbor. *See* probably approximately correct (PAC) nearest neighbor query
proximity. *See* proximity query
range. *See* range query
range aggregation. *See* range aggregation query
rectangle intersection. *See* rectangle intersection problem
reverse nearest neighbor (RNN). *See* reverse nearest neighbor query (RNN)
skyline. *See* skyline query
spatial join. *See* spatial join query
stabbing. *See* stabbing query
stabbing counting. *See* stabbing counting query
window. *See* window query
quicksort, 661
QUILT quadtree GIS, xviii
quintary tree, 6–9, *7*
construction cost, 13, 748
exact match query, 13, 748
partial range query, 7, 13, 748
range query, 13, 748
space requirements, 8, 13, 747

R

R-file, 115, 256, 478–480
deletion, 480
insertion, 480
one-dimensional. *See* one-dimensional R-file
point query, 479
two-dimensional. *See* two-dimensional R-file
window query, 480
r-optimal algorithm, 494, 497–498
R-tree, xvi, xviii, xxii–xxiii, 12, 61, 68, 96–97, 101, 112, 117, 127–130, 165, 171, 187, 189, 280n, 282–290, *282*, 292–296, 301–302, 308, 376, 382, 398, 427, 488–489, 509–511, 515, 535, 537–538, 548, 556, 566–574, 579, 585, 595–596, 604, 606, 614, 622, 624–626, 629, 663, 674–676, 685, 791–792, 841
bulk insertion. *See* object-tree pyramid
bulk loading. *See* object-tree pyramid
deletion, 287, 791
Hilbert. *See* Hilbert R-tree
incremental nearest neighbor finding, 494–495, 666–667
insertion, 287
nearest object query, 289, 792
node-splitting policies. *See* R-tree node-splitting policies
order, *282*
ordered. *See* ordered R-tree
overflow, *283*
packed. *See* packed R-tree
parametric. *See* parametric R-tree
peer-to-peer (P2P), 271
point location query, 61
point query, 288
priority. *See* priority R-tree
rectangle intersection problem, 289
restructuring
incremental refinement, *302*, 302
postoptimization, *302*, 302
search hierarchy, 492–493, 666–667, 833
skyline query, 505
spatial sampling, 780
underflow, *283*
Voronoi diagram, 573
window query, 289
rectangular, 289, 792
R^*-tree, 283, 285, 289–296, *289*, 300–303, 465, 488, 548, 566–570, 572, 586, 629, 674–675, 681–682, 685, 791–793
forced reinsertion. *See* forced reinsertion
nearest neighbor query, 594
reinsertion
close-reinsert. *See* close-reinsert
far-reinsert. *See* far-reinsert
R^+-tree, 12, 96, 130, 311–312, *311*, 376, 382, 773
deletion, 312
insertion, 312
point query, 312
rectangle intersection problem, 312
space requirements, 312, 794
window query, 312

R-tree node-splitting policies, 284–289, 791–792
exhaustive, 284–285, 291
reducing overlap, 285–288, 291, 294
seed-picking, 285
linear cost, 285, 287–288, 290–292, 294, 296
quadratic cost, 285, 287–288, 290–292, 294, 296, 793
radial-edge data structure, 316
radix searching, 2
random access structure, *202*
random lines model, 369, 376
randomized algorithm, *711*
Las Vegas. *See* Las Vegas randomized algorithm
Monte Carlo. *See* Monte Carlo randomized algorithm
range aggregation query, *485n*
range priority tree, 9, 23–25, *23*, 187
construction cost, 23, 27, 751
space requirements, 23, 27, 751
two-dimensional range query, 25–27, 751
updates, 23, 27
range query, *3*, 4. *See also* window query
AESA. *See* AESA
aggregation. *See* range aggregation query
BD-tree. *See* BD-tree
bisector tree. *See* bisector tree
bit concatenation. *See* bit concatenation
bit interleaving. *See* bit interleaving
buddy-tree. *See* buddy-tree
double Gray order. *See* double Gray order
dynamic p hashing. *See* dynamic p hashing
dynamic z hashing. *See* dynamic z hashing
ECDF tree. *See* ECDF tree
feature, *485*
generalized pseudo k-d tree. *See* generalized pseudo k-d tree
Geometric Near-neighbor Access Tree (GNAT). *See* Geometric Near-neighbor Access Tree (GNAT)
gh-tree. *See* gh-tree
Gray order. *See* Gray order
grid file. *See* grid file
inverted file. *See* inverted file
k-d-B-trie. *See* k-d-B-trie
kB-tree. *See* kB-tree
kNN graph. *See* kNN graph
linear hashing with reversed bit interleaving (LHRBI). *See* linear hashing with reversed bit interleaving (LHRBI)
M-tree. *See* M-tree
monotonous bisector tree (mb-tree). *See* monotonous bisector tree (mb-tree)
multidimensional B-tree (MDBT). *See* multidimensional B-tree (MDBT)
multidimensional extendible hashing (MDEH). *See* multidimensional extendible hashing (MDEH)
mvp-tree. *See* mvp-tree
MX-CIF quadtree. *See* MX-CIF quadtree
one-dimensional range tree. *See* one-dimensional range tree
optimized k-d tree. *See* optimized k-d tree
priority search tree. *See* priority search tree
quintary tree. *See* quintary tree
SASH (Spatial Approximation Sample Hierarchy). *See* SASH (Spatial Approximation Sample Hierarchy)
semi-infinite. *See* semi-infinite range query
Spatial Approximation tree (sa-tree). *See* Spatial Approximation tree (sa-tree)
two-dimensional range tree. *See* two-dimensional range tree
U order. *See* U order
uniform grid. *See* uniform grid
vp-tree. *See* vp-tree
vp^{sb}-tree. *See* vp^{sb}-tree
range tree, xxii–xxiii, 5, 13–18, 187, 189–190, 749–750, 776
multidimensional. *See* multidimensional range tree
one-dimensional. *See* one-dimensional range tree
range query
one-dimensional range tree. *See* one-dimensional range tree
two-dimensional range tree. *See* two-dimensional range tree
space requirements, 15
two-dimensional. *See* two-dimensional range tree
RANGESEARCH, *15*
RANGETREE field
node record in two-dimensional range tree, *17*
ranking
ECDF tree. *See* ECDF tree
raster data, 367
raster-scan order, *779*, *See also* row order
RASTERTOVLIST, *421*, 423, 818
ray tracing, 398–399
recall, 664, *664*
recognition cone, 424
RECT field
bnode record in MX-CIF quadtree, *470*
rectangle intersection problem, 428–445, *428*, 819–822
balanced four-dimensional k-d tree. *See* balanced four-dimensional k-d tree
bin method. *See* bin method
dynamic interval tree. *See* dynamic interval tree
expanded MX-CIF quadtree. *See* expanded MX-CIF quadtree
grid file. *See* grid file
interval tree. *See* interval tree
MX-CIF quadtree. *See* MX-CIF quadtree
one-dimensional range tree. *See* one-dimensional range tree
ordered R-tree. *See* ordered R-tree
priority search tree. *See* priority search tree
R-tree. *See* R-tree
representative point rectangle representation. *See* representative point rectangle representation
R^+-tree. *See* R^+-tree
segment tree. *See* segment tree
tile tree. *See* tile tree
rectangle placement problem, 449, 452
rectangle quadtree, xxiii, *480*, 481
rectangle record
MX-CIF quadtree, *470*
rectangular coding, 208
RECTINTERSECT, *470*
recursive linear hashing, 733–734, *733*, 874
overflow file. *See* overflow file
red-black balanced binary tree, 19, 440–443, 722–723, 726–727, 873
2-3 tree, *722*, 722n, 726, 872. *See also* 2-3 tree
bottom-up insertion algorithm, 722–723, 726
deletion, 726
top-down insertion algorithm, 726
2-3-4 tree, *722*, 722n, 726, 872. *See also* 2-3-4 tree
bottom-up insertion algorithm, 723, 726
deletion, 726
top-down insertion algorithm, 726–727
black edge, *722*
red edge, *722*
reduced combined index, *6*
Reduced Overhead AESA (ROAESA). *See* ROAESA
reduction factor, *65*
reference, *743*
reference sets, *691*
selection, 692–693
refinement, *511*, *663*
surface. *See* surface simplification
REFLECT, *218*
reflection hierarchy, *198*
region, *200n*
α-balanced. *See* α-balanced region
aspect ratio. *See* aspect ratio of region
canonical. *See* canonical region
eight-connected. *See* eight-connected component
fat. *See* fat region
four-connected. *See* four-connected component
nonobject (white), *200n*
object, *200n*
skinny. *See* skinny region

region bucket, *97*
region capacity, *476*
region octree, 211–220, *212*, 314, 358, 361, 363, 370, 399, 409, 418, 425, 779–782
  Boolean set operations, 372
  space requirements, 373
region page
  k-d-B-tree, 103. *See also* k-d-B-tree
  k-d-B-trie. *See* k-d-B-trie
region quadtree, xvii, 37, 44, 141n, 211–223, *211*, 226–227, 233, 257, 259, 266–267, 305, 314, 356, 358, 380–381, 399, 402, 409–415, 423–425, 448, 466, 473, 510, 515–516, 779–782, 803, 815–816
  construction
    bottom-up, *779*
    predictive, *779*
    top-down, *779*
  maximum clique problem, 449, 452
  multidimensional measure problem, 448, 452–453
    surface area, 452
  optimal position, 219, 780
  spatial sampling, 780
  translation, 425
region tree, *243*
  separator node. *See* separator node
regular B-partition, *112*
regular decomposition, *3*, *211*
regular prism tree, *389*, 389
regular tiling, *197*
regularization
  polygonal map, *242*, 244, 247–249
    plane-sweep, 788
regularized Boolean set operations, 314n, 393
relative neighborhood graph (RNG), 639–643, *639*, 859–861
  nearest neighbor query, 643
REMOVECHILDSOFNEWNODEFROMFATHER, *176*
REMOVENODEFROMFATHER, *176*
REMOVEPRIORITYQUEUE, *518n*
REPARTITION, *786*
representative point, 99n, 103n, *453*
representative point rectangle representation, 453–466, 824–827
  Cartesian coordinates, 453
  Cartesian product, 454
  extension parameters. *See* extension parameters
  four-dimensional point, 454–463, 824–827
    balanced four-dimensional k-d tree. *See* balanced four-dimensional k-d tree
    containment query, 456, 458
    enclosure query, 456, 458
    grid file. *See* grid file
    k-d tree. *See* k-d tree
    LSD tree. *See* LSD tree
    point query, 456, 458
    PR quadtree. *See* PR quadtree
    rectangle intersection problem, 456
    window query, 456, 458
  location parameters. *See* location parameters
  polar coordinates, 453
  two-dimensional point
    BANG file. *See* BANG file
    PR k-d tree. *See* PR k-d tree
    PR quadtree. *See* PR quadtree
    spatial k-d tree. *See* spatial k-d tree
resolution
  multiple. *See* multiple resolution
restricted bintree, *64*, 406–408, *406*, 814
restricted d-dimensional quadtree, 408
restricted quadtree, *64*, 381–382, 405–408, *405*
restricted rectangle quadtree. *See* restricted quadtree
restricted triangular quadtree, *408*
RETURNTREETOAVAIL, *806*
reverse kNN graph, *658*, 662, 864
reverse nearest neighbor query (RNN), 658, *658n*
  influence set. *See* influence set
reversed bit concatenation, *143*
reversed bit interleaving, 140, *142*, 146, 159
revised line quadtree, *357*, 357, 803
RIGHT field
  line record in PM quadtree, *367*
  node record in one-dimensional range tree, *15*
  node record in priority search tree, *21*
  node record in two-dimensional range tree, *16*
  TYPE field value vertex
    ldnode record in layered dag, *251*
RMD-tree, 98n
RNG. *See* relative neighborhood graph (RNG)
RNN. *See* reverse nearest neighbor query (RNN)
ROAESA, *649*, 649–650
  alive, *649*
  not alive, *649*
robot, 425
robotics, xv, 373, 425
rosette, 231
rotation
  point k-d tree. *See* point k-d tree
  point quadtree. *See* point quadtree
rotation hierarchy, *198*
routing object, *521*
row order, 199–202, *200*, 278, 296, 423, 776–778, *779*, 818–819
row-prime order, 199–202, *199*, 776–778
RR_1 quadtree, *261*
RR_2 quadtree, *261*
runlength code, 42. *See also* runlength encoding
runlength encoding, 42, 205, *206*, 409–417, *409*, 420, 422, 814–816
  change-based. *See* change-based runlength encoding
  Morton order, 414
  Peano-Hilbert order, 414
  value-based. *See* value-based runlength encoding
runlength representation. *See* runlength encoding

S

S-tree, *301*, 303, 411, *411*, 411n, 414
  skew factor. *See* skew factor
S^*-tree, 412
S^+-tree, 411–412, *411*, 414–415, 815
sa-tree. *See* Spatial Approximation tree (sa-tree)
SAIL, xx, 743
SAND Internet Browser, xviii
SASH, *651*
SASH (Self Adaptive Set of Histograms), 650n
SASH (Spatial Approximation Sample Hierarchy), 557, 582, 599, 650–662, 863–864
  approximate k nearest neighbor finding, 655–658, 662, 716, 863
    Las Vegas randomized algorithm, 712
    Monte Carlo randomized algorithm, 712
  construction, 651–653
  orphan object. *See* orphan object
  range query, 662
SATREEINCNEAREST, *634*
Schönhardt Twisted Prism, 353
SDFT. *See* Sorted Discrete Fourier Transform (SDFT)
search hierarchy. *See* nearest neighbor query
  AESA. *See* AESA
  gh-tree. *See* gh-tree
  LAESA. *See* LAESA
  M-tree. *See* M-tree
  MX-CIF quadtree. *See* MX-CIF quadtree
  R-tree. *See* R-tree
  vp-tree. *See* vp-tree
searching, xix
second normal form (2nf), 329
secondary key, 91
secondary key retrieval, *1n*
secondary structure, 435, *435*
sector tree, 377–381, *378*, 811
  complete. *See* complete sector tree
  intersection, 379
  rotation, 380–381
  scaling, 379
  translation, 380
  union, 379
segment tree, xxii–xxiii, 430–434, 445–447, 451–452, 473, 819–823
  containment set, 444–445

segment tree *(continued)*
deletion, 432
insertion, 432
maximum clique problem, 449, 452
measure problem, 447, 452
multidimensional. *See* multidimensional segment tree
perimeter computation problem, 447, 452
rectangle intersection problem, 430–434, 439, 460, 819–820
space requirements, 432
stabbing counting query, 434
stabbing query, 434, 822
two-dimensional. *See* two-dimensional segment tree
window query, 444
segmentation, 424
selectivity estimation, 259
semi-infinite range query, *19*, 440
semiclosed boundary, *429*
semijoin
distance. *See* distance semijoin
separate chaining method, 729, *729n*
separating chain, 69, 224, 242–254, *243*, 788–789
dynamic edge insertion, 245–246, 248, 789
point location query. *See* gap tree
separation factor, *582*
separator node, *243*
septree, *231*
sequential heap, *106*
sequential list, *5*, 185, 776
exact match query, 5, 13, 747
sequential scan method, 592–597, 854
nearest neighbor query, 594
serial data structure, 49
set, *194n*
set cover algorithm
greedy. *See* greedy set cover algorithm
shading, 197
SHAREPM12VERTEX, *804*
SHAREPM3VERTEX, *808*
shell in an object, *316*
shortest-path algorithm
Dijkstra. *See* Dijkstra's shortest-path algorithm
shortest-path map, *510*
shortest-path quadtree, *510*
shrink operation in balanced box-decomposition tree (BBD-tree), *83*
shuffle order, *143*
sibling pointer, *108n*
sign language, 40
silhouette, 361
similar tiling, *197*
similarity retrieval. *See* similarity searching
similarity searching, xv, xvii, 164, *485*, 685
simple QSF-tree, 465
simplex, 353, *353*, 354n
simplicial complex, 353, *353*, 354n, *869*
simplification
view-dependent. *See* view-dependent simplification
simplification envelope method, 395
simplification methods, *391*, *See* surface simplification
SIMULA, xx, 743
single balance, *753*
Singular Value Decomposition (SVD), xvii, 66n, 591, 671–677, *672*, 681, 683–686, 683n, 690, 697–698, 704, 710–711, 866
Approximate SVD transform matrix. *See* Approximate SVD transform matrix
transform matrix, *672*
site, *69*, *346*, 581, 658n
SIZE, *518*
size-bound method, *256*, 257, 259
skeleton, 208–209, *209*
adaptively sampled distance field (ADF). *See* adaptively sampled distance field (ADF)
quadtree. *See* quadtree skeleton
skew factor, *302*
skinny region, *64*
skip list, 27n, 661
k-d tree
half-space range query, 27n
PR quadtree
approximate nearest neighbor query, 27n
approximate range query, 27n
point location, 27n
skycube, *503n*
skyline cube, *503n*
skyline points, *503*
skyline query, 502–507, *503*
block-nested loop strategy, 503–504
divide-and-conquer strategy, 503
partial ordering, 506–508
plane-sweep methods, 504–507
preference function. *See* preference function
R-tree. *See* R-tree
sort-first block-nested loop strategy, 503, 505
top-*k*. *See* top-*k* skyline query
total ordering, 502–506
slab, *449*
slice, *146*
slice-and-dice, *225*
sliding-midpoint k-d tree, 72–74, 77, 764–766
space requirements, 74, 76–77, 766
sliding-midpoint rule, *72*
Slim-tree, *625*
Slim-down algorithm, 625
Small-Tree-Large-Tree (STLT) method, 297–298
solid modeling, xv–xvi, xix, xxii, 425
solid tile, *459*
sort-tile-recurse method (STR method), 277–278
Sorted DFT (SDFT). *See* Sorted Discrete Fourier Transform (SDFT)
Sorted Discrete Fourier Transform (SDFT), 684, *684*
sorting, xv
SP-GiST. *See* spatial partitioning generalized search tree (SP-GiST)
space hierarchy, 189
space planning, 425
space requirements
balanced four-dimensional k-d tree. *See* balanced four-dimensional k-d tree
bounded quadtree (BQT). *See* bounded quadtree (BQT)
bucket PR-CIF k-d tree. *See* bucket PR-CIF k-d tree
bucket rectangle PM quadtree. *See* bucket rectangle PM quadtree
expanded MX-CIF quadtree. *See* expanded MX-CIF quadtree
hinted quadtree. *See* hinted quadtree
quad list quadtree (QLQT). *See* quad list quadtree (QLQT)
segment tree. *See* segment tree
space tile, *459*, 460
space-filling curve, *90*, 199, 444n, 776
space-ordering methods, 199–202, 275, 776–778
admissible. *See* admissible order
pruning property, 669–670
stable. *See* stable order
sparse matrix, 423
SparseMap, 686, 690, 694–697, 709–710
cluster preservation ratio (CPR). *See* cluster preservation ratio (CPR)
greedy resampling, *694*
Spatial Approximation tree (sa-tree), 598–599, 629–637, 639, 641, 643, 858–859
ancestor neighbors, *634*
exact match query, 633
incremental nearest neighbor query, 634–637
nearest neighbor query, 650
neighbor set, *631*
range query, 633–634
spatial database, xv–xvi, xix, xxii, 164, 166
spatial distance, *510*
spatial indexing, 366
spatial join query, xx, *428*, 438, 465, 485
dynamic interval tree. *See* dynamic interval tree
interval tree. *See* interval tree
spatial k-d tree, *62*, 68, 101, *464*, 610n
space requirements, 102
two-dimensional representative point rectangle representation, 464
point query, 464
spatial network, *508*, 516
incremental nearest neighbor finding.

See incremental nearest neighbor finding
spatial partitioning generalized search tree (SP-GiST), 188–190, 271, 590
external methods. *See* external methods in SP-GiST
interface parameters. *See* interface parameters in SP-GiST
spatial range query. *See* window query
spatial sampling
R-tree. *See* R-tree
region quadtree. *See* region quadtree
spatiotemporal database, xvii, xix, 445
sphere tree, 195, 383, 389, 567–568, 622, 625
Spherical Pyramid Technique (SPY-TEC), *587*
spherical triangle, 231
spiral hashing, xxii, 130–131, 153, 156–160, 187–188, 190, 774
asymptotic analysis, 742, 875
basics, 735–742, 874–875
growth factor. *See* growth factor
merging, 735, 741, 874
overflow bucket. *See* overflow bucket
primary bucket. *See* primary bucket
splitting, *735*
storage utilization factor. *See* storage utilization factor
spiral order, 199–202, *199*, 776–778
spiral storage, 130. *See also* spiral hashing
split operation in balanced box-decomposition tree (BBD-tree), *83*
split-and-merge segmentation algorithm, *424*
grouping step, *424*
merge step, *424*
pyramid. *See* pyramid
split step, *424*
split-edge lath data structure. *See* split-face lath data structure
split-face lath data structure, 333, 336–344, 797
split-quad data structure, *343*
split-vertex lath data structure, 333–344, 798
SPLITPMNODE, *805*
splitting
linear hashing. *See* linear hashing
MX quadtree. *See* MX quadtree
MX-CIF quadtree. *See* MX-CIF quadtree
PR quadtree. *See* PR quadtree
spiral hashing. *See* spiral hashing
splitting threshold, *75*, *375*, 375
SPY-TEC. *See* Spherical Pyramid Technique (SPY-TEC)
SQUARE field
node record in PM quadtree, *367*
square record
PM quadtree, *367*
square tiling, *197*
squarecode, 208
SR-tree, 164, 195, 533, 569–573
SS-tree, 561, 567–572, 622, 625, 674–675, 852
stabbing counting query, *434*
interval tree. *See* interval tree
segment tree. *See* segment tree
stabbing query, *434*
interval tree. *See* interval tree
multidimensional segment tree. *See* multidimensional segment tree
priority search tree. *See* priority search tree
segment tree. *See* segment tree
two-dimensional interval tree. *See* two-dimensional interval tree
two-dimensional segment tree. *See* two-dimensional segment tree
stable order, *199*
standard deviation, 423
star-vertex data structure, *355*
static database, 2
static multipaging, *136*
statistical analysis, *46*
Steiner point tetrahedralization problem
NP-complete, 353
Steiner point, *262*, 352–353
STLT method. *See* Small-Tree-Large-Tree (STLT) method
storage utilization factor, *731*
STR method. *See* sort-tile-recurse method (STR method)
streaming, *451*
stress, 689–690, *689*
strip tree, 360, 362, 382–386, *382*, 389, 395, 812–813
area, 385
area-area intersection, 386
construction
bottom-up, 385
top-down, 382, 385, 812
curve-area intersection, 385
curve-curve intersection, 385
three-dimensional, 388
subSASH, *651*
subset relation, 427
subsubSASH, *651*
subsumed, *220*
subsumption property, *220*
superface algorithm, 398
superkey, *50*, 52, 54, 59
surface decimation, 393–399
face decimation. *See* face decimation
vertex clustering. *See* vertex clustering
vertex decimation. *See* vertex decimation
surface simplification, 238, 391–399, 813
decimation, *392*, *See* surface decimation
geometric, 392
refinement, *392*, 393
topological, 392
SVD. *See* Singular Value Decomposition (SVD)
symmetric structure, *317*

T

(t,ε)-AVD, 581–585, *581*
telephonic transmission, 40
telescoping vector. *See* TV-tree
ternary hierarchical triangular decomposition, *400*, 401–402
tertiary structure, 435, *436*
tessellation, *237*, *See also* polygonal tiling
hexagonal. *See* hexagonal tiling
test-concatenation techniques, *79*
tetrahedralization
Delaunay. *See* Delaunay tetrahedralization
tetrahedron, *232*
tetrahedron table, 354
then, *744*
Thiessen polygon. *See* Voronoi diagram
third normal form (3nf), 329
threshold, 423
TID, *208*
tile
atomic. *See* atomic tile
molecular. *See* molecular tile
solid. *See* solid tile
space. *See* space tile
tile tree, 435, 435n, 438n, 473–474
rectangle intersection problem, 438n
tiling
adjacency number. *See* adjacency number of a tiling
hexagonal. *See* hexagonal tiling
isohedral. *See* isohedral tiling
limited. *See* limited tiling
nonpolygonal. *See* nonpolygonal tiling
polygonal. *See* polygonal tiling
regular. *See* regular tiling
similar. *See* similar tiling
square. *See* square tiling
trapezoidal. *See* trapezoidal tiling
triangular. *See* triangular tiling
uniform orientation. *See* uniform orientation of a tiling
uniformly adjacent. *See* uniformly adjacent tiling
unlimited. *See* unlimited tiling
time-parametric rectangle, *286*
time-series databases, 681–683, 685
TLAESA, 618, *649*, 649–650
incremental nearest neighbor query, 650
top-down region quadtree construction. *See* region quadtree
top-k skyline query, *504*, 505–506
topological entities
primitive. *See* primitive topological entities
topological simplification. *See* surface simplification
topological sort, 243
total grid partition, *141*

total ordering, 506, *506n*
  skyline query. *See* skyline query
total path length (TPL)
  k-d tree. *See* k-d tree
  optimized point quadtree. *See* point quadtree
  point quadtree. *See* point quadtree
touching rule, *261*, 261, 261n
TPR-tree, *286*, 445, 488
TPR*-tree, *286*, 445, 488
trafficability, 260
training set, *60*
transformation method, 671
trapezoidal tiling, *197*
traveling salesman problem, *202*
  NP-complete, 202
treap, *27n*, *See also* Cartesian tree
tree directory, 12, *96*
tree directory methods, 96–130, 165, 768–773
tree-based representations, 184, 189
tree-based searching methods, *2*
treemap, 66, 225–226, *225*, 572. *See also* puzzletree; X-Y tree
trellis, *449*, 452
  multidimensional measure problem, 449–452
tri-cell, *455*, 462
triacon hierarchy, *232*, 401
trial midpoint split operation in balanced box-decomposition tree (BBD-tree), *84*
triangle collapse, *394*
triangle decimation, 394
triangle inequality, 514–515, 599, 650, 658
triangle intersection problem, 447
triangle measure problem, 453
triangle strip, 399
triangle table, 353–355, *354*, 367, 584, 802
triangular cell. *See* tri-cell
triangular decomposition
  hierarchical
    quaternary. *See* quaternary hierarchical triangular decomposition
    ternary. *See* ternary hierarchical triangular decomposition
Triangular Irregular Network (TIN), *400*, 400
triangular tiling, *197*, 231–232
triangulation, 262, 367
  bucket PR octree. *See* bucket PR octree
  conforming Delaunay. *See* conforming Delaunay triangulation
  constrained Delaunay. *See* constrained Delaunay triangulation
  Delaunay. *See* Delaunay triangulation (DT)
  greedy. *See* greedy triangulation
  hierarchical
    adaptive. *See* adaptive hierarchical triangulation (AHT)
  minimum weight. *See* minimum-weight triangulation (MWT)
  plane-sweep. *See* plane-sweep triangulation
  PR quadtree. *See* PR quadtree
trie, *2*, 6, 142, 423, 467, 729–730
  k-d. *See* k-d trie
  quad. *See* quadtrie
trie-based k-d tree, 70–87, 764–767
trie-based quadtree, 37–47, 756–761
trie-based representations, 184, 189
trie-based searching methods, *2*
trivial split, *72*
truncated-tree pyramid, *268*
TRYTOMERGEPM1, *806*
TRYTOMERGEPM23, *807*
TR*-tree, *303*
TV-tree, 572, *572*
twin grid file, *137*, 137, 773
two-dimensional bucket PR-CIF k-d tree, 476–478, 480, 482
two-dimensional interval tree, 446
  containment query, 446
  enclosure query, 446
  stabbing query, 446
  window query, 446
two-dimensional priority search tree, *447*
two-dimensional R-file, 478
two-dimensional range tree, xxiii, 15–18, 444, 821, 823
  construction cost, 16, 18, 749
  range query, 16, 18, 749–750
  space requirements, 16, 18, 749
two-dimensional representative point rectangle representation, 463–464, 827
  BANG file. *See* BANG file
  PR k-d tree. *See* PR k-d tree
  PR quadtree. *See* PR quadtree
  spatial k-d tree. *See* spatial k-d tree
two-dimensional runlength encoding, 413–416, *413*, 815–816
two-dimensional segment tree, *445*, 446–447
  containment query, 446
  enclosure query, 446
  stabbing query, 446
  window query, 446
two-level grid file, *132*
two-manifold surface, *315n*, *372*
TYPE field
  ldnode record in layered dag, *251*

U

U order, 94, 199–202, *199*, 216, 776–779
  partial range query, 95, 768
  range query, 95
unambiguous boundary model, *316*
uniform extendible array of exponential varying order (UXAE), 147n
uniform grid, 10, *10*, *210*, 393
  range query, 10, 14, 749
uniform hashing, 653
uniform orientation of a tiling, *198*
uniform quadtree. *See* PR quadtree
uniformly adjacent tiling, *198*
union neighborhood graph (UNG), *642*, 642, 860
unit-segment tree, 430, *430*, 434, 448
unit-size cells
  array access structure, 202–203
  tree access structure, 203–204
unlimited tiling, *197*
UP field
  TYPE field value edge
    ldnode record in layered dag, *252*
  TYPE field value gap
    ldnode record in layered dag, *252*

V

V-rectangle, *449*
VA-file. *See* vector approximation file (VA-file)
value, *743*
VALUE field
  node record in one-dimensional range tree, *15*
value-based runlength encoding, *409*, 414, 416
VAMSplit k-d tree, 58–59, *129*, 129–130, 186–187, 773
VAMSplit R-tree, 129–130, *130*, 187, 279, 568, 570–571, 586
VAMSplit R+-tree, *130*
vantage object, *688*
vantage point, *598*
vantage point tree (vp-tree). *See* vp-tree
variable resolution, 424
variable-resolution representation, *266*
Variance DFT (VDFT). *See* Variance Discrete Fourier Transform (VDFT)
Variance Discrete Fourier Transform (VDFT), 684, *684*
VASCO. *See* Visualization and Animation of Spatial Constructs and Operations (VASCO)
VA+-file, 592, 594–597, *594*
  Karhunen-Loève Transform (KLT), 594, 596–597
VDFT. *See* Variance Discrete Fourier Transform (VDFT)
vector approximation file (VA-file), 592–597, *592*, 854
  nearest neighbor query, 594
vector data, 366
vector octree, 367
vector quadtree, 367
vector quantization problem, 486. *See also* nearest neighbor query; nearest object query
VEND, *319*
vertex
  cone, *418*
  vertex representation, *418*
  weight. *See* weight of a vertex
vertex bucket polygon quadtree, *262*, 264, 376–377, 809–810
vertex clustering, 393–394

vertex decimation, 394–395
triangle decimation. *See* triangle decimation
vertex removal. *See* vertex removal
vertex simplification. *See* vertex simplification
vertex enlargement, 263–264, *263*, 790
vertex list, 421–423, *421*, 818
vertex removal, 394, *394*
vertex representation, 195, 417–423, 817–819
vertex. *See* vertex
vertex simplification, 394–395, *394*
memoryless, 395
optimal placement, 394
vertex-based object representations using laths, 329–346, 796–799
vertex-edge relation, 317
vertex-edge table, *318*
vertex-lath table, *332*
VERTEXEDGETABLE, *320*
vertical rectangle, *449*
very large-scale integration (VLSI), xvii, xxii, 211, 260, 417, 427, 443, 481–482
view-dependent simplification, 399
visibility number, *236*, *787*
visible vertex, *238*
visual computing, *xix*
visualization, xviii
Visualization and Animation of Spatial Constructs and Operations (VASCO), xviii, *271*
VLSI. *See* very large-scale integration (VLSI)
volume computation problem, 448
volume graphics, 363
volume rendering, 363
Voronoi cell, *346*
Voronoi Diagram, xxii, *69*, 322, 346–348, *346*, 350, 352–354, 367, 486, 515, 563, 566, 573–580, 598, 603, 616, 623, 629–631, 633, 658n, 799, 853
approximate. *See* approximate Voronoi diagram (AVD)
discrete. *See* discrete Voronoi diagram
nondegenerate. *See* nondegenerate Voronoi diagram
R-tree. *See* R-tree
site. *See* site
site reconstruction problem, 350, 800–802
Voronoi graph, *629n*
Voronoi property, *630*, 630, 858
Voronoi region, *346*
Voronoi tree, *618*, 621–622, *622*, 857
voxel, *192*
vp-tree, xxiii, 555, 599, 604–614, 616, 621–622, 636n, 855
approximate nearest neighbor query, 607
dynamic, 605
incremental farthest neighbor query, 607
incremental nearest neighbor query, 606–607
correctness criterion, 607
range query, 606–607
search hierarchy, 606
vp^s-tree, *607*
vp^{sb}-tree, 607–608, *608*, 612, 855
incremental nearest neighbor query, 608
range query, 608
VSTART, *319*

W

warping process, *63*
wavelets, 676
weakly edge visible polygons, *239*
weakly proximity preserving embedding, *686*
weight balancing, 7
weight of a vertex, 418, *418*
weighted randomized search tree, *27n*
well-separated pair decomposition (WSPD), *582*, 582
well-separation, *582*
WHITE value
NODETYPE field
node record in MX quadtree for points, *41*
node record in PR quadtree, *46*
WHITEBLOCKS, 816
window query, *36*, 192, *428*, *444*, 444
balanced four-dimensional k-d tree. *See* balanced four-dimensional k-d tree
bounded quadtree (BQT). *See* bounded quadtree (BQT)
bucket PR-CIF k-d tree. *See* bucket PR-CIF k-d tree
bucket rectangle PM quadtree. *See* bucket rectangle PM quadtree
corner stitching. *See* corner stitching
Generalized Bulk Insertion (GBI). *See* Generalized Bulk Insertion (GBI)
hinted quadtree. *See* hinted quadtree
iMinMax method. *See* iMinMax method
interval tree. *See* interval tree
MX-CIF quadtree. *See* MX-CIF quadtree
ordered R-tree. *See* ordered R-tree
packed R-tree. *See* packed R-tree
pyramid technique. *See* pyramid technique
quad list quadtree (QLQT). *See* quad list quadtree (QLQT)
R-tree. *See* R-tree
representative point rectangle representation. *See* representative point rectangle representation
R^+-tree. *See* R^+-tree
segment tree. *See* segment tree
two-dimensional interval tree. *See* two-dimensional interval tree
two-dimensional segment tree. *See* two-dimensional segment tree
winged-edge data structure, xxii, 317–330, *318*, 342–345, 353, 367, 584, 795–796
explicit. *See* non-oriented winged-edge data structure
implicit. *See* oriented winged-edge data structure
non-oriented. *See* non-oriented winged-edge data structure
oriented. *See* oriented winged-edge data structure
winged-edge-face data structure, *319*, 322, 324, 329, 341
winged-edge-vertex data structure, *319*, 322, 324, 329, 341

X

x-discriminator, *50*
x-interval, *248*
x-partition, *476*
X-tree, 164, 301, 566–567, 587, 589, 595–596
nearest neighbor query, 594
supernode, *566*
X-Y tree, 66, 225–230, *225*, 260, 572, 783–786. *See also* puzzletree; treemap
XCOORD field
node record in point quadtree, *31*
node record in PR quadtree, *46*
point record in layered dag, *252*
point record in PM quadtree, *367*
point record in two-dimensional range tree, *17*
XVAL field
TYPE field value vertex
ldnode record in layered dag, *251*
XXL. *See* Extensible and Flexible Library (XXL)

Y

y-discriminator, *50*
y-monotone subdivision, 69, *242*
y-partition, *476*
YCOORD field
node record in point quadtree, *31*
node record in PR quadtree, *46*
point record in layered dag, *252*
point record in PM quadtree, *367*
point record in two-dimensional range tree, *17*
YSEARCH, *22*

Z

z buffer algorithm, 236
Z order, 93–94, 172, *200*, *See also* Morton order; N order
z^+-order, 150–152, *150*, 774
zkd Btree, *93*, 97, 257
zkd tree, *93*