Index

binary search, 77
boolean, 77
degree, 245
implementation, 924
index out of range, 74
node array, 245
permutation, 77
set operations, 78
sorting, 77
articulation point, 277, 300
assert, 54
assignment
basics, 32–36
for handle types, 939
for vectors, 118
implementation, 920
LEDA rule, 33
assignment problem, see matchings in graphs
associative array, see sparse array
augmenting path, 365
AVL-tree, 126, see sparse array

BALABAN SECTIONS, 734
balls and bins experiment, 94
barycentric coordinates, 658
BB[a]-tree, 126, see sparse array
Bellman-Ford shortest path algorithm, 338, 345
BELLMAN.FORD, 322, 349
BELLMAN.FORD.B, 339
BF.Gen, 344
BFS, 295
biconnected component, 299
biconnected graph, 277, 299
BICONNECTED.COMPONENTS, 301
bidirected graph, 275, 501–506
vs undirected graph, 574
bidirectedness, test for, 279
bigfloat, see number types

lw, 987
__exportC, 198
<< for class window, 821
>> for class window, 821
(a, b)-tree, 126, see sparse array
ABACUS, 11
abstract data type, 16
access to attributes, LEDA rule, 30
acyclic graph, 244
ACYCLIC.SHORTEST_PATH, 322, 333
adjacency list, 272, 511
affine transformations, 601–604
AGD, 11
algebraic numbers, see number types
Algorithmic Solutions Software GmbH, 8
algorithms, see geometry algorithms, see graph
algorithms, see sorting
alias method, 83
ALL.PAIRS.SHORTEST_PATHS, 323
alternating path, 363
amortized time bound, 24
angle, 603, 666
animation, see window, see GraphWin
animation of strongly connected components, 306
annulus, 703
anonymous object, 16, see object, 21
area
of a simplex, 597
of a triangle, 593
of polygon, 764
arithmetic, see number types
arithmetic demand of geometric computations, 632
arithmetic demand of network algorithms
assignment algorithm, 438
general discussion, 286–292
maximum flow algorithm, 487
weighted matching algorithms, 419, 431
array, 73–77
Index

bigints, see integer
bin_tree, 917
binary heap, 149, see p_queue
binary tree, 126, see sparse array, 917
binomial tree, 953
bipartite cardinality matching, 360, see matchings in graphs
bipartite graph, 277
bitmap, 826, see window
BL PLANAR, 507, 529, 537, 561
blossom, 398
boolean arrays, 77
boolean operations on polygons, 762
Booth–Lueker planarity test, 526
boundary cycle, 516
bounded queue, 60
bounded region wrt. geometric object, 598
bounded stack, 60
braided lines, 110, 610
breadth-first search, 252, 295
bucket heap, 149, see p_queue
bucket sort for lists, 66, 71
buffering drawing operations, see window
built-in types, 931
bush form, 521
button, 831, see panel
cache effects, 69, 268
cache miss, 70
callwidth, 974
cardinality matching, see matchings in graphs
Cartesian coordinates, 583
C++ class and data type, 41–44
CGAL, 11, 655
CHECK MAX CARD MATCHING, 396
CHECK MAX FLOW, 448
CHECK MCB, 362
CHECK MWBM, 416
check.reversal.inf, 503
check.weights, 287
checked priority queue, 165
Chiba et al. embedding algorithm, 529
circle, 585
side of, 598
clipping regions, see window
cmp_key, 917
col, see color
collinear, 593
color, 817, see also window
compare
and linear order, 48
and linearly ordered type, 45
implementation, 916, 917
compare function, 50
instantiation of, 94
compare object
definition, 50
example of use, 665, 742
compare_by_angle, 600, 666
compilation flag, 57
complete graph, 263
COMPONENTS, 299
components in graphs, 296
compressed boolean array, 77
compute S, 289
copy constructor
as member function, 43
implementation, 920, 924
of type parameter, 45
container type, 26
cubic hulls, 637–656
3d-hull, 855
arbitrary dimension, 790
demo, 638
display of 3d-hull, 855
divide-and-conquer algorithm, 656
extreme point, 659
incremental algorithm, 643
incremental algorithm, 643
randomized incremental algorithm, 648
runtime, 652, 653
sweep algorithm, 640
CONVEX HULL, 640, 651
JC, 643
RIG, 651
S, 640
copy, 32–36
deep copy, 35
LEDA concept, 35
LEDA rule, 33, 34, 49
shallow copy, 35
copy constructor
as member function, 43
implementation, 920
of type parameter, 45
CopyGraph, 243, 274
copying a graph, 243
correctness, see program checking, number types,
rational geometry kernel, errors, see program checking
design goal, 12
exact geometric computations, 582
preconditions, 22
rational geometry kernel, 582
safe use of floating point kernel, 632
cost of a path, 317
creation, see object
CRUST, 5, 705
curve reconstruction, 5, 705
CUT VALUE, 492
cycle
minimum cost to profit ratio, 354
negative, 317
cyclic adjacency list, 511
d3.window, 835
d.array, see sparse array
danger of floating point arithmetic, see errors
dart, 658
data member, 41
data structure, see data type
data type
basics, 16–26
built-in type, 20
C++ class, 41
collection of items, 28
default linear order, 48
handle type, 937
hashed type, 45, 49
implementation, see implementation of ... implementation parameter, 51
instantiation, 17, 46
item type, 26
large type, 929
linearly ordered type, 45, 48
non-primitive type, 32, 34
parameterized type, 17, 909
pointer type, 32
primitive type, 32
small type, 929
specification, 17
deep copy, 35
default constructor
as member function, 43
implementation, 924
of type parameter, 45
default linear order, 48
default value of data type, 19
definition, see variable
degeneracy, 638
DELAUNAY
...DWYER, 680
...FLIPPING, 676
Delaunay triangulations, see triangulations, 672-686,
see Voronoi diagrams
arbitrary dimension, 790
checking, 679, 683
definition, 672
Delaunay diagram, 680
demo, 673
diagonal-flip, 674
divide-and-conquer algorithm, 680
Dwyer’s algorithm, 680
dynamic, see dynamic Delaunay triangulations
empty circle property, 672
essential edge, 681
Euclidean minimum spanning tree, 684
flipping algorithm, 673
functionality, 675
furthest site, 675
Guibas–Stolf algorithm, 680
implementation of flipping algorithm, 676
largest angle property, 684
running time, 680
visualization, 890
DELAUNAY.DIAGRAM, 683
delaunay.edge.info, 659
DELAUNAY.TO.VORONOI, 693
DELAUNAY.TRIANG, 675
delaunay.voronoi.kind, 675
deleter-operator, 47, 941
demo
detailed examples
animation of SCCS, 306
comparison of sparse arrays, 130–132
tables for IGX, 96–98
Voronoi demo, 795–812
with GraphWin, see GraphWin
with window, see window
directory, 9
programs
3d hull and 2d Delaunay, 793
animation of a graph algorithm, 306
annuli, 703
basic graph algorithms, 278
bipartite cardinality matching, 361
braided lines, 108
convex hulls, 638
curve reconstruction, 7, 705
Delaunay triangulation, 673
dynamic Delaunay triangulations, 709
Euler’s number, 107
exact arithmetic, 114
genetic weighted matchings, 444
giant component experiment, 159
greedy heuristic, 377
integer multiplication, 101, 102
largest empty circle, 702
line segment intersections, 732
markov chain, 86
matchings in general graphs, 394
maximum flow, 15
minimum cost flows, 490
minimum cut in graphs, 491
minimum spanning tree, 685
minimum spanning trees, 261
minimum cost to profit ratio, 355
planarity, 508, 509
point sets, 709
polygons, 759, 787
random walk, 87
real numbers, 114, 115
run lengths, 81
shortest paths, 321
shortest paths and assignment, 439
smallest enclosing circle, 701
sorted sequences, 183
stacks and expressions, 59
strongly connected components, 306
Voronoi diagrams, 687
weighted bipartite matching, 433
word count, 2
running times, see running time experiments
xllman, 9
dependent item type, see item type
depth-first search, 293–316
descriptor width, 974
design goals, 11
correctness, 12
ease of use, 11
efficiency, 13
extensibility, 12
destruction of object, 36
destructor
as member function, 44
explicit call, 929
implementation, 920
of type parameter, 45
determinant, 120
determinant, fast evaluation, 622
DFS_NUM, 293
dictionaries for geometric objects, 588
dictionary
geometric, see dynamic Delaunay triangulations
dictionary, 146–147
Index

implementation, 917–924
dictionary array, see sparse array, 126–127
animation, 897
dijkstra 3, 100, 150, 255, 283, 322, 323, 336
dijkstra-gen, 155
directed graph, 240
directory tree of LEDA, 8
discrete event simulation, 148
distance, 599
divide-and-conquer paradigm, 884
Delaunay triangulation, 680
documentation, 963–991
code unit, 972, 978
example of a header file, 964
example of a manual page, 965
man, 965
ldoc, 986
leave, 988
literate programming, 982, 986–991
lman, 963–966
options, 983
redirecting output, 982
lweave, 987
Lweb, 987
making a manual, 984–985
manual commands, 967–981
  manpage, 967
  Marop, 976
  Minop, 975
  Minopfunc, 977
  Mconversion, 977
  Mcreate, 971
  Mcreation, 971
  Mdefinition, 969
  Mdestroy, 973
  Mexample, 979
  Mfunc, 977
  Mfuncop, 976
  Mimplementation, 979
  Mname, 971
  Mop, 974
  Moperations, 973
  Moptions, 980
  Mpreamble, 989
  Mstatic, 976
  Msbust, 981
  Mtext, 980
  Mypemember, 970
  Mypes, 970
  Munop, 976
  Munopfunc, 977
  Mvar, 971
  manual comment, 963
  manual directory, 985–986
  manual pages, 966–984
  invisible functions, 978
  structure, 967
  TeX macros, 981
  warnings, 981
  Mkman command, 984
  notangle, 986
  noweave, 986
  noweb, 986
  double, see number types
double click event, 837
drawing of a graph, 499–501
without crossings, see embedded graphs
dual of a map, 573
duality between Voronoi and Delaunay diagrams, 690
dynamic binding, 916
dynamic Delaunay triangulations, 708–731
  checking, 713
class definition, 711
  constructors, 708, 714
del, 709, 722
demo, 709
dim, 708
functionality, 708
implementation, 711–730
insert, 709, 719
locate, 709, 714
lookup, 708, 717
nearest_neighbor, 710, 726
point location, 709
range search, 710, 729
running time, 729, 731
walk through a triangulation, 715
dynamic memory, see heap
dynamic perfect hashing, 127, see sparse array
dynamic random variate, see random.variate
ease of use, 3, 11
edge, 240
array, 245
map, 249
matrix, 245
edge contraction, 399, 404, 493
dulsector, 664, 696
Edmonds' matching algorithm, 397
efficiency
  design goal, 13
  of geometric computing, 613
element of a data type, 17
embedded graph (map), 498–580
forall, 571
basics, 506–511
bidirected, 501–506
boundary cycle, 516
construction
  by drawing, 575–577
  by embedding algorithm, 507
  from graph, 505
definition of embedded graph, 506
definition of map, 502
drawing, 499–501
dual of a map, 573
face (geometric concept), 516
face (programming concept), 571–574
face cycle (combinatorial concept), 512
generation of plane maps and graphs, 569–571
genus, 514
genus of plane map, 515–519
order-preserving embedding, 511–512
planar embedding, 506–511
planar map, 574
planarity, 506–511
plane test, 520–529
plane map, 512, 515–519
number of edges, 518
undirected graph vs bidirected graph, 574
updates, 564–569
equality
for item types, 28
LEDAn rule, 28, 49
of geometric objects, 586
equivalence relation, 48
error analysis, 106
error handler, 54
errors
boolean operations on polygons, 788
braided lines, 108, 610
convex hull with floating point arithmetic, 609, 642
floating point arithmetic and geometry, 609–612, 629
network flow and floating point arithmetic, 487
overflow, 286
overflow and Dijkstra's algorithm, 100
swEEP segments, 756
Euclidean minimum spanning tree, 684
event, see also window, 834–842
button_press_event, 835
button_release_event, 835
configure_event, 835
double click, 837
key_press_event, 835
key_release_event, 835
motion_event, 835
non-blocking event input, 841
position, 835
put back, 839
queue, 835
read, 835, 841
time, 835
type, 835
value, 835
window, 835
exact geometric computations, 582
expected running time, 25
explicit destruction, 923
expression compiler, 622
expression evaluation, 59
extensibility of LEDA, 12
extension packages, 9
extreme point, 659
F. DELAUNAY_TRIANG, 675
F. VORONOI, 689
face (geometric concept), 516
face (programming concept), 571
face cycle (combinatorial concept), 512
Fibonacci heap, see p_queue
implementation, 946–962
assignment, 961
clear, 961
construction, 951
decrease_key, 953
del_item, 961
del_min, 952, 956
heap-ordered forest, 948
insetPosition, 955
item, 948
iteration, 961
simple operations, 951
storage, 948

virtual functions, 952
properties, 149
filter, see floating point filter
FIVE_COLOR, 577
float, see number types
floating point filter, 613–631
definition and correctness, 613–619
determinant, 622
dynamic version, 621
efficacy and efficiency, 623–631
expression compiler, 622
in reads, 112
orientation predicate, 614
original version, 619
specialized arithmetics, 622
static version, 620
floating point integers, 616
floating point kernel, see geometry kernels
floating point numbers, see number types
floor, 290
Fman, see documentation
forall-loop, see iteration
forall_items-loop, see iteration
Ford and Fulkerson matching algorithm, 368
frexp, 289
function
as argument, 38
member, 41
optional argument, 23
overloading, 23
parameter passing, 33, 36–38
value return, 33
Gaussian elimination, 104, 120
GDToolkit, 11
gen_polygons, see polygons
general position assumption, 638
generalized polygons, see polygons
generation of random graphs, 265
generators
geometry, see geometric objects
graph, see graph generators
random source, see random source
generic directory, 8
generic pointer, see GenPtr
generic programming, 635
generics, 916
GenPtr, 918
Genus, 515
genus of a map, 514
genus of plane maps, 515
geometric computation
arithmetic demand, 632
efficiency, 613
exact, 582
go geometric dictionary, see dynamic Delaunay
triangulations
go geometric graph, 664
geometric objects, see also geometry kernels
angle, 600
arithmetic demand, 632
associating information, 588
basics, 583–592
bounded region, 598
Cartesian coordinates, 583
Index

comparing distances, 600
conversion between kernels, 588
distance, 599
drawing them in a window, 821
equality, 586
generators, 604–606
generic names, 606
handle types, 586
homogeneous coordinates, 583
identity, 586
immutability, 589
input and output, 589
intersections, 601
kernel independence, 606
length, 599
mouse input, 590
negative side of geometric object, 597
on-region, 598
polygons, see polygons
positive side, 597
precision of geometric representation, 633
reading them in a window, 821
representation, 587
side of, 597
truncation of precision, 633
unbounded region, 598
window, 589, 821
geometric primitives, 593–601
comparing distances, 600
intersections, 601
length and distance, 599–600
orientation, 593–597
sidedness, 597–599
geometric transformations, 601–604
geometry algorithms
angle order, 600
convex hull with floating point arithmetic, 609
convex hulls, see convex hulls
degeneracy, 638
Delaunay diagram, 625
Delaunay triangulations, see Delaunay triangulations
dynamic Delaunay triangulations, see dynamic Delaunay triangulations
general position assumption, 638
higher-dimensional algorithms, see higher-dimensional geometry
line segment intersection, see line segment intersections
triangulations, see triangulations
verification of geometric structures, see verification of geometric structures
Voronoi demo: the complete program, 795–812
Voronoi diagrams, see Voronoi diagrams
width of a point set, 651
geometry kernels, see also geometric objects, 581–636
basic geometric objects, 583–592
conversion between kernels, 588
danger of floating point kernel, 609–612
floating point filter, see floating point filter, 613
floating point kernel, 581
gemetric primitives, 593–601
higher-dimensional, 634
kernel independence, 606–609
rational kernel, 581
efficiency, see floating point filter
safe use of floating point kernel, 632–634
giant component in a graph, 159
GML-format, 269, 872
graph, 240–282
foral, 571
acyclic, 244
adjacency list, 272, 511
articulation point, 277, 300
associating information with nodes and edges, 245–251
basics, 240–245
biconnected, 277, 299
bidirected, 275, 501–506
bipartite, 277
breadth-first search, 252
connected, 277, 299
construction by drawing, 575
cyclic adjacency list, 511
degree, 242
directed, 240
drawing, 499–501
data, 245
map, 249
embedded graph, 506, see embedded graph
face (programming concept), 571
face cycle (combinatorial concept), 512
five coloring, 577
foral, 241, 271
generation of planar graphs, 569
hiding edges, 273
I/O, 269–271
independence sets, 578
index of a node or edge, 504
is bidirected, 501
is map, 502
isomorphic copy, 243
iterating over edges, 502
iteration, 241, 271–274, 571
list of edges, 272
of nodes, 271
loopfree, 275
make bidirected, 501
make directed, 258
make map, 502, 505
make planar map, 574
make undirected, 258
map, 502, see embedded graph
moving an edge, 565
node, 240
array, 245
data, 245
list, 251–253
map, 249
matrix, 245, 249
partition, 259–263
priority queue, 253–257
opposite node, 257
order-preserving embedding, 511
parameterized, 280
path, 276
planar, 507
Index

handler functions, 879
input and output, 872
layout operations, 872
leaving edit mode, 867
menus, 882
miscellaneous functions, 875
slider, 880, 896
zooming, 866, 874
mouse interaction
creating a node, 858
double click, 859
dragging a node, 859
moving a node, 859
resizing a node, 859
overview, 858–860
panel, 875, see panel parameters, 861–866
change of, 869
default values, 870
edge attributes, 864
global parameters, 865
node attributes, 861
reading them, 869
postscript output, 872
visualizing geometric structures, 890–892
greedy heuristic for matchings, 374

h.array, see sparse array
handle type, 937–941
use in geometry, 586
handle_base, 939
handle_rep, 939
Hash, 45
hashed type
definition, 45
LEDA rule, 49
hashing array, see sparse array, 127–129
hashing with chaining, 127, see sparse array
header file, 56
decoration for manual production, 967
heap (data structure), 149, see p.queue
heap (dynamic memory), 910, 912
hiding edges of a graph, 273
higher-dimensional geometry, 634, 790–795
covex hulls, 790
Delaunay triangulations, 790
demo, 793
higher-dimensional kernel, 790
lifting map, 791
paraboloid of revolution, 791
simplicial complex, 790
higher-dimensional kernel, 790
history of LEDA, 13
homogeneous coordinates, 583
Hopcroft and Karp matching algorithm, 380
HT_PLANAR, 507
HTML, 9

I/O
for graphs, 269–271
for type parameters
implementation, 917
Print, 45
Read, 45
required functions, 45
graphical output, see window
menus, see panel
mouse input, see window
panels, see panel
tables for \LaTeX, 96–98
identity
for handle types, 939
for item types, 28
LEDA rule, 28
of geometric objects, 586
IEEE standard, 104
immutability of attributes, 31
immutability of geometric objects, 589
impl directory, 8
implementation class, 909
implementation of data types, 904–937
abstract class, 909
arrays, 924–926
assignment, 920–924
concrete class, 909
copy constructor, 920–924
default constructor, 924–926
destructor, 920–924
detailed examples
dictionary, 917–924
Fibonacci heap, 946–962
list, 904–917
map, 133–146
sortset, 196–228
stack, 41–44
efficiency, 915
generics, 916
implementation parameters, 934–937
large type, 929
LEDA approach, 909–928
leda.access, 926, 931
leda.cast, 927, 931
leda.clear, 926, 930
leda.copy, 926, 930
leda.create, 926, 930
leda.type_id, 931
optimizations, 929–934
pure virtual functions, 917
small type, 929
summary of LEDA approach, 930
template approach, 906–909
type safety, 915
virtual functions and dynamic binding, 916–919
implementation parameter
basics, 51
implementation, 934–937
in-place new-operator, 929
include directory, 8
incremental construction paradigm
analysis, 648
convex hull, 643
independent item type, see item type
INDEPENDENT_SET, 578, 579
index out of bounds check, 118
infinity and MAXINT, 100
inheritance
and compare objects, 50
and parameterized data types, 935
Index

Jordan sorting, 228
k-heap, 149, see p-queue
Karasuba integer multiplication, 103
kernel independance, 606
K's, 507
Kruskal’s algorithm, 260
K3, 3, 507
KURATOWSKI, 508
Kuratowski subgraph
algorithm, 538–564
definition, 507
running time, 563
KURATOWSKISIMPLE, 538

labels of geometric objects, 588
large type, 929
LARGEST EMPTY CIRCLE, 702
layered network, 381
ldexp, 290
ldoo, see documentation
LEDA_CHECKING.OFF, 57
LEDA extension packages, 9
LEDA rules
access to attributes, 30
assignment, 33
copy, 33, 34, 49
definition with initialization by copying, 20
destruction, 36
equality, 49
equality of items, 28
hashed type, 49
identity of items, 28
independent item types, 31, 32
iteration, 39, 40
linear orders, 48
type parameters, 45
leda.access, 926, 931
leda.assert, 54
leda.cast, 927, 931
leda.clear, 926, 930
leda.cmp.base, 50
leda.copy, 926, 930
leda.create, 926, 930
leda.max, 53
LEDA_MEMORY, 47, 942
leda.min, 53
leda.swap, 53
leda.type.id, 931
LEDAROOT, 6
leftmost depth-first search, 535
leftturn, 593
Lempel–Even–Cederbaum planarity test, 520
length, 586, 599
LEP, 9
lifetime of object, 21, 36
lifting map, 791
line, see geometry objects, 585
line segment intersections, 731–758
asymptotic running times, 734
demo, 732
functionality, 733
running time, 735, 755, 757
sweep line algorithm, see sweep line paradigm

initialization, 924
input, see I/O
installation of LEDA, 8
instance of a data type, 17
instantiations of network algorithms, 283
int, 99, see number types
int_set, 77–79
INT.TYPE, 583
integer, 101, see number types
integer matrix, 117–120
integer vector, 117–120
interpolation, 656
intersections of geometric objects, 601
inverse of a matrix, 120
IO interface, 96–98
Is.Acyclic, 276
Is.Biconnected, 277
Is.Bipartite, 248, 275
Is.Bipartite, 277
Is.C.Increasing, 665
Is.C.Nondecreasing, 665
Is.CC.W.Convex.Face.Cycle, 668
Is.CC.W.Ordered, 666
Is.CC.W.Ordered.Plane.Map, 666
Is.Connected, 277
Is.Convex.Subdivision, 669
Is.Delaunay.Diagram, 683
Is.Delaunay.Triangulation, 679
Is.Planar, 277, 507
Is.Plane.Map, 512
Is.Simple, 275
Is.Triangulation, 669
Is.Triconnected, 277
Is.Voronoi.Diagram, 697
item, see item type
item type, 26–32
assignment, 27
collections of items, 28
dependent item type, 28–30
implementation, 918, 936
equality, 27
forall.item.loop, 39
identity, 28
independent item type, 31–32
implementation, 937
LEDA rule, 51, 32
nil, 30
iteration
addition of objects, 273
basics, 39–41
deletion from data structure, 40, 944
deletion of object under iterator, 272
for graphs, 241, 271–274, 571
for lists, 63
for sparse arrays, 125
implementation, 40, 945–946
insertion into data structure, 944
LEDA rule, 39, 40
macro expansion, 40, 943
over the wedges of a graph, 502
pitfall, 40
STL style, 41
summary of forall.statements, 943–945
iterator, see iteration
Index

linear algebra, 118, 119
linear order, see compare
default order, 48
definition, 47
for points, 587
implementation, 916
LEDAl rule, 48
linear system of equations, 104, 118
linearly ordered type, 43, 48
linker, 909
list, 61–73
basics, 61–64
bucket sort, 71
concatenation and split, 64
for ordered sets, 66–70
forall, 63
implementation, 70–73, 904–917
iteration, 63
list item, 62
merging, 68
permutate, 65
reverse, 65
singly linked, 73
sorting, 66, 68
list heap, 149, see p-queue
literating programming, 42, 982, 986–991
Lman, see documentation
LMDFS, 536
long, 99, see number types
LU decomposition, 120
lweave, 42, see documentation
Lweb, 42, see documentation

machine precision, 105, 616
Make_Biconnected, 277
Make_Bidirected, 276
Make_Connected, 277
manual
directory, 9
how to make one, see documentation
on-line access, 9
manual comment, 963
map (data type), see sparse array
basics, 129–130
gedge map, 249
for geometric objects, 588
implementation, 133–146
access operation, 144
assignment, 142
constructors, 142
destructor, 143
fast access, 136
hash function, 135
header file, 133
implementation class, 134
iteration, 145
manual page, 123
node map, 249
map (embedded graph), see embedded graph
Markov chain, 86
matchings in graphs
alternating path, 363
alternating tree, 397
assignment problem
algorithm, 437
and shortest paths, 438
arithmetic demand, 438
functionality, 413
augmenting path, 363
bipartite graphs, 360–393
All et al. algorithm, 384
basic algorithm, 366
best case, 372
bfs vs dfs, 377
checker, 362
comparison of algorithms, 390
demo, 361
difficult example, 390
directed and undirected view, 367
Ford and Fulkerson algorithm, 368
functionality, 361
greedy heuristic, 374
Hopcroft and Karp algorithm, 380
node cover, 362, 570
running time, 375, 380, 392
bipartite matching
greedy heuristic, 377
bipartite weighted graphs, 413–443
algorithm, 419
arithmetic demand, 419, 431
checker, 416
demo, 433
functionality, 413
implementation, 425
linear program, 417
maximum cardinality, 441
potential function, 415
reduced cost, 415
running time, 433
general graphs, 393–413
algorithm, 397
checking, 396
cover, 394
demo, 394
functionality, 393
implementation, 402
odd-set cover, 400
running time, 395
general weighted graphs, 443
demo, 444
functionality, 443
mate, 361
perfect matching, 361
shortest paths via assignment, 438
matrix, 73, see array
matrix, 117–120
MAX_CARD_BIPARTITE_MATCHING, 361, 367
.AMP, 386
.FB, 369
.FDFS, 373
.HK, 381
MAX_CARD_MATCHING, 393, 403
MAX_FLOW, 445
.BASIC, 456
.GAP, 482
.GRR, 479
.JH, 469
.LRH, 472
INDEX

SCALE.CAPS, 446
MAX.WEIGHT ASSIGNMENT, 415, 437
MAX.WEIGHT Bipartite Matching, 287, 290, 415, 425
MAX.WEIGHT MATCHING, 443
MAXDOUBLE, 105
maximal planar graph, 570
maximal planar map, 569
maximum, see leda max
maximum flow, 443–489
arbitrary-rule, 453
arithmetic demand, 487
checker, 449
current edge, 462
cut, 447
definition of flow, 443
elelible edge, 451
FIFO-rule, 453, 458
flow augmentation, 448
flows with lower bounds, 489
gap heuristic, 479
global relabeling heuristic, 472
highest-level-rule, 453, 458
local relabeling heuristic, 469
low-high distinction, 467
max-flow-min-cut theorem, 446
mincost flow, see minimum cost flow
non-saturating push, 450
optimizations, 465
preflow, 450
preflow-push algorithm, 451
preflow-push implementation, 456
problem generators, 462
residual network, 447
running time, 462, 466, 470, 473, 480, 483, 486
saturating push, 450
summary of experiments, 485
two-phase approach, 475
use of floating point arithmetic, 487
MAXINT, 99
MAXINT and infinity, 100
MCB.EFFECT.OF. HEURISTIC, 376
member function, 41
memory hierarchy, 69
memory leak, 913, 919, 928
memory management, 47, 941–943
memory usage, statistics of, 53
menu, 849, see panel
merge sort
basic algorithm, 68
run generation, 81
with finger search, 191
MIN.AREA. ANNULUS, 704
MIN.COST.FLOW, 489
MIN.COST.MAX.FLOW, 490
MIN.CUT, 491, 493
MIN.SPANNING.TREE, 261, 684
MIN.WEIGHT.ASSIGNMENT, 415
MIN.WIDTH. ANNULUS, 704
MINDOUBLE, 105
minimum, see leda min
minimum cost flow, 489–491
demo, 490
functionality, 489
minimum cost to profit ratio cycle, 354
minimum cuts in graphs, 491–497
algorithm, 492
demo, 491
functionality, 491
heuristic, 496
implementation, 492
running time, 492
minimum spanning tree, 260
MIN. RATIO. CYCLE, 354
MININT, 99
mouse input, see window
MULTILE.SEGMENTS, 734
multiple inheritance and implementation parameters, 935
multiplication of boolean matrices, 78
multiplication of large integers, 101, 103
MW.SCALE.WEIGHTS, 419
MWB.SCALE.WEIGHTS, 291, 419
MWMCB.MATHCING, 415, 441
n- gon, 606
NaN (not a number), 105
nearest neighbor queries, 710
negative cycle, 317
negative orientation, 593
negative side, 597
new-operator, 21, 47, 912, 941
in-place version, 929
new edge operation, 564
nil, 30
node, 240
array, 245
list, 251
map, 249
matrix, 245
partition, 259
priority queue, 253
node cover, 362
non-primitive data type, 32, 34
non-uniform distribution, 83
normalization, 103
notangle, 42, see documentation
noweb, see documentation
noweave, see documentation
number types, 99–117
algebraic numbers, 108–117
demo, 114
efficiency, 114
example for use, 111
implementation, 116
read, 108
separation bound, 112
concept, 45
floating point numbers, 104–108
danger of use, see errors
definition, 104
error analysis, 106
Euler’s number, 106
exponent, 104
IEEE standard, 104
mantissa, 104, 105
NaN (not a number), 105
precision, 105
rounding error, 105
© Cambridge University Press
Index

rounding mode, 105
truncation, 53
integers, 99–103
int, 99
integer, 101
multiplication, 101
overflow, 100
two-complement representation, 99
underflow, 100
rational numbers, 103–104
vectors and matrices, 117–120

object, see variable
anonymous object, 21
concept, 16
construction, 19
copy constructor, 43, 45
creation, 33
default constructor, 43, 45
destruction, 36
destructor, 45
lifetime, 21
named object, 18
object code, 909, 913
optional argument of function, 23
order-preserving embedding, 511
ordered sets, 66
orientation
determinant, 594
in higher dimensions, 595
in space, 596
in the plane, 593
orientation, 593
output, see I/O
overloading, 23

p-queue
implementation, see Fibonacci heap
pair, 94
pairing heap, 149, see p-queue
panel, see also window, 844–854
action function, 846, 849
adding buttons, 850
adding items, 849
bitmap buttons, 851
bool item, 845
button, 848
call back, 846
choice item, 844
item, 844
menu bar, 853
menu button, 849
multiple choice item, 844
open, 849
pixrect buttons, 852
simple item, 844
slider, 845, 850
text item, 844
paraboloid of revolution, 791
parameter passing by value, 37
parameterized data type, see data type
parameterized graph, 280
parametric search, 358
partition, 158–180
basics, 158–161
checking priority queues, 165–180
giant component experiment, 159
implementation, 161–165
node partition, 259
union-find data structure, 161
path, 276
persistence of variables, 132
perturbation, 718, 769
pixel, see window
pixel coordinate system, 814, 821, see window
pixmap, see window
pixrect, see window
PLAN.EMBED, 529, 533
PLANAR, 507
planar embedding
algorithm, 507, 529–538
definition, 507
running time, 563
planar graph, 277, 507
planarity and number of edges, 518
planarity test, 507, 520–529
running time, 563
plane map, 511, 512, 515–519
PLANTEST, 528
point, see geometry objects
Cartesian coordinates, 583
construction, 584
conversion between kernels, 588
equality, 586
homogeneous coordinates, 583
identity, 586
immutability, 589
implementation, 938
input and output, 589
linear order, 587
orientation, 593
point.rep, 587
point.rep, 938
print.statistics, 624
random points, 604
representation, 587
truncation of precision, 589
W, 583
X, 583
xcoord, 583
Y, 583
ycoord, 583
point.on.circle, 604
point.set, see dynamic Delaunay triangulations
pointer type, 32
polygons, 788–789
area, 762, 767
boundary, 762
closure, 762
complement, 761, 762, 771
declaration, 760
demo, 759, 787
determining the orientation, 769
edges, 761
functionality, 758
generalized polygons
area, 764
boundary cycles, 766
reg.n.gen, 606
region.of, 598
regularized set operations, 762
replacement selection, 80
restoring the edges of a graph, 273
rgb-value of a color, 817
rightturn, 593
root operation, 101, 108, see number types
rotating caliper method, 651
rotations, 602
rounding error, 105
rounding mode, 105
run generation for merge sort, 81
running time, 23–26
amortized, 24
asymptotic bounds, 24
expected, 25
of randomized algorithm, 25
worst case, 24
running time experiments
arrays vs maps, 250
basic graph algorithms, 278
bipartite cardinality matching, 375, 380, 392
cache effects, 70
checked priority queue, 178
compact vs non-compact graph representation, 268
comparison of queues implementations, 157
convex hull, 652, 653
curve reconstruction, 6
deletion triangulations, 680
Dijkstra’s, 4
floating point filter, 628
integer multiplication, 102
LEDA versus STL, 14
line segment intersections, 735, 756, 757
matchings in general graphs, 395
maximum flow, 466, 470, 473, 480, 483–486
maximum flow, comparison, 15
memory manager, 943
merge sort, 193
merging, 192
minimum cuts in graphs, 492
minimum spanning trees, 263
nearest neighbor search, 729, 731
optimization for built-in types, 933
planarity, 563
polygons, 788, 789
random graph generation, 267
random variates, 91
real numbers, 115
shortest paths, 4, 338, 346
shortest paths and assignment, 439
sorting and merging, 69
sorting nearly sorted sequences, 186
sorting of arrays, 76
sparse arrays, 127, 128
STL, 14
sweep line algorithm, 756, 757
triangulations, 663
weighted bipartite matching, 433
safe use of floating point kernel, 632
scalar product, 586
scale.weight, 290
scale.weights, 290
scaling weights in network algorithms, 289
screen shot, see window
segment, see geometry objects, 584
SEGMENT.INTERSECTION, 733
set (data type), 146–147
sets and boolean arrays, 77
shallow copy, 35
short, 99, see number types
shortest paths, 316–360
acyclic graphs, 322, 332
all pairs, 323, 352
assignment problem, 438
Bellman–Ford algorithm, 338, 345
checker, 324
comparison of priority queues, 150
deno, 321
general edge costs, 322, 338, 345, 359, 438
generator of difficult graph, 343
generic algorithm, 326
non-negative edge costs, 255, 322
output convention, 320
problem definition, 317
running time, 338, 346, 439
single sink, 323, 333, 339
theory, 328
worst case for Dijkstra’s algorithm, 155
shortest-path tree, 320
SHORTEST_PATH, 322
shrinking strongly connected components, 299
SHRINKEN.GRAPH, 299
side.of, 597
side_of.circle, 632
sidedness, 597–599
sign of an expression, 108, 613
signed area of a triangle, 593
signed int, 99
simplex, 597
simplicial complex, 790
singly linked list, 73
skiplist, see sparse array, 196–228
as implementation of dictionary arrays, 126
assignment, 208
concatenation, 220
constructors, 206
data members, 202
data structure, 196
deletion, 216
deletion of subsequences, 226
destructor, 208
dienger search, 212
header, 197
header file of skiplists, 202
header file of sorted sequences, 203
implementation file, 203
insertion, 216
memory management, 198, 205
merge, 225
optimization of search, 210
probabilistic analysis, 201
reversal of subsequence, 219
search operations, 209
skiplist node, 198
st item, 198
split, 222
tower, 197
validation of data structure, 199
virtual functions, 203
small type, 929
SMALLEST.ENCLOSING.CIRCLE, 702
SORT.EDGES, 575, 666
sorted sequence, see sorted
sorted, 180–228
basics, 181–183
concatenation, 189
deletion of subsequence, 195
finger search, 183–186
finger search and merging, 189
finger search and sorting, 184
implementation by skiplists, 196–228
item arguments, 195
Jordan Sorting, 228–239
merging, 189
reversal of subsequence, 195
running time, 186, 192, 193
split, 186
sorting
bucket sort, 66
for lists, 66
insertion sort with finger search, 184
Jordan sorting, 228
merge sort, 68
merge sort with finger search, 191
nearly sorted sequences, 186
stable sort, 66
straight insertion sort, 74
topological sorting, 244
source code directory, 9
source node, 240
space requirement, 23
sparse array, 121
basics, 121–126
default value, 124
defined, 125
dictionary array (d.array), 126–127
dynamic perfect hashing, 127
experimental comparison, 130–132
first examples, 122
forall, 126
hashing array (h.array), 127–129
hashing with chaining, 127
implementation parameters, 126–132
map, 129–130
implementation, 133–146
performance guarantees, 126–132
persistance of variables, 132
running time, 127, 128
subscriber operator, 124
survey of properties, 122
undefined, 125
sparse arrays, 77
specification of data type, 17
SPRING.EMBEDDING, 500
square root operation, 108, see number types
st-numbering of a graph, 301, 520
ST.NUMBERING, 301
stable sorting, 66
stack, 58–61
bounded, 60
implementation, 41
statistics, 52
STL, 14, 41
straight insertion sort, 74
straight line embedding, 507
STRAIGHT.LINE.EMBEDDING, 508
string, 95–96
STRONG.COMPONENTS, 298, 304
strongly connected component
algorithm, 301
animation, 306
definition, 297
strongly connected graph, 297
subdivision, 664
convex, 669
subgraph, 273
substring operator
for arrays, 73
for lists, 62
for sparse arrays, 124, 132
supermarket checkout simulation, 91
swap, see leda swap
sweep line paradigm
convex hull, 640
line segment intersections, 735–757
algorithm, 736
asymptotic running time, 736
experimental evaluation, 755
implementation, 738–755
running time, 756, 757
Y-structure, 736
triangulations, 650
SWEEP.SEGMENTS, 735
sys directory, 9
target node, 240
template approach to data types, 906
template directory, 8
templates for network algorithms, 283–286
testing, see program checking, 664
thread safety, 9
timer, 842, see window
topological sorting, 244
TOPSORT, 244
transformations (geometric), 601–604
transitive closure, 297
transitive reduction, 297
TRANSITIVE.CLOSURE, 297
translations, 602
TRIANGULATE.POINTS, 660
triangulation.map, 570
triangulations, see Delaunay triangulations, 656–664
barycentric coordinates, 658
checking, 669
dart, 658
definition, 656
interpolation, 656
largest angle property, 684
representation as planar graphs, 658
running time, 663
Index

triconnected graph, 277
triple, 94
TRIVIAL_SEGMENTS, 734
truncate, 53, 633
tuple, 94
two-complement representation, 99
two-tuple, 94
type, see data type
type parameter, 45–46
type safety, 915
typewidth, 974
u_graph, 258
unbounded region wrt. geometric object, 598
undirected edge, 502
undirected graph, 257–258
undirected graph vs bidirected graph, 574
union-find data structure, 161
unsigned int, 99
user time, 52
user coordinate system, 814, 821, see window
value parameter, 33, 35, 37
variable, see object
collection, 16
definition
examples, 18
with default initialization, 19
with initialization by copying, 20
vector, 585
vector, 117–120
verification of correctness, see program checking
verification of geometric structures, 664–672
convex subdivisions, 669
convexity of faces, 667
Delaunay diagrams, 683
Delaunay triangulations, 679
monotonicity of sequences, 665
order of adjacency lists, 666
Voronoi diagrams, 695
verzopfle Geraden, 610
virtual functions, see implementation of data structures,
916, 917, 922
visual debugging, 664
visualization, see window, see GraphWin
visualizing a graph, 271
void*, see GenPtr
volume of a simplex, 597
VORONOI, 689
Voronoi diagrams, see Delaunay triangulations, 686–707
annulus, 703
checking, 695
curve reconstruction, 705
definition, 680
demo, 687, 701–703, 705, 705–812
diagram of line segments, 111, 706
duality to Delaunay diagrams, 690
functionality, 689
furthest site, 689
largest empty circle, 702
representation, 686
smallest enclosing circle, 701
visualization, 890
walk through a triangulation, 715

web-site, 8, 10
weighted matchings, see matchings in graphs
WIDTH, 651
width of a point set, 651
window, 813–856
<<, 821
>>, 821
bitmap, 826
buffering drawing operations, 829
button, 831
clearing a window, 823
clipping region, 828
color, 817, 823
creation, 815
d3_window, 855
drawing a geometric object, 589
drawing a ray, 591
drawing operations, 821–831
drawing section, 813
event, 834, see event
example programs
3d hull, 855
blocking mouse read, 832
bouncing ball, 830
callback functions, 846
clipping, 828
constructing colors, 817
creating a menu bar, 853
creating a panel, 851, 852
event handling, 836
putting back an event, 840
recognizing a double click, 838
slider items, 847
two windows, 842
use of redraw, 847
use of bitmap, 827
use of buffering, 831
use of buttons, 833
use of pixrect, 825
use of timers, 843
use of << and>>, 822
riters, 589
graphics system, 814
input, 821, 831–842
invisible color, 817
<<, 589
menu, 849
mouse cursor, 820
mouse input, 821, 831–842
mouse input of geometric objects, 590
opening and closing a window, 815–816
output, 821
panel, 844, see panel
panel section, 813
parameters, 818–820
change of, 820
pix_to_real, 821
pixel, 813
pixel coordinate system, 814, 821
pixmap, 824
pixrect, 824–827
real_to_pix, 821
rgb-value of a color, 817
scaling factor, 815
Index

screenshot, 822
src_mode, 819
timer, 842
user coordinate system, 814, 821
xlman, 813

xor_mode, 819
xpm data string, 824
word count program, 2, 52, 122, 146, 934
xlman, 9, 813