



## 附录

## 参考文献

- [1] D. E. Knuth. The Art of Computer Programming, Volume 1: Fundamental Algorithms (3rd edn.). Addison-Wesley (1997), ISBN:0-201-89683-1
- [2] D. E. Knuth. The Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd edn.). Addison-Wesley (1997), ISBN:0-201-89684-8
- [3] D. E. Knuth. The Art of Computer Programming, Volume 3: Sorting and Searching (2nd edn.). Addison-Wesley (1998), ISBN:0-201-89685-0
- [4] A. V. Aho, J. E. Hopcroft, J. D. Ullman. The Design and Analysis of Computer Algorithms (1st edn.). Addison-Wesley (1974), ISBN:0-201-00029-0
- [5] J. Bentley. Writing Efficient Programs. Prentice-Hall (1982), ISBN:0-139-70251-2
- [6] J. Bentley. More Programming Pearls: Confessions of a Coder. Addison Wesley (1988), ISBN:0-201-11889-0
- [7] R. L. Graham, D. E. Knuth, O. Patashnik. Concrete Mathematics: A Foundation for Computer Science (2nd edn.). Addison-Wesley (1994), ISBN:0-201-55802-5
- [8] 严蔚敏 等. 数据结构 (C语言版). 北京: 清华大学出版社, 1997年4月第1版, ISBN:7-302-02368-9
- [9] J. Bentley. Programming Pearls (2nd edn.). Addison-Wesley (2000), ISBN:0-201-65788-0
- [10] T. Budd. Classic Data Structures in Java. Addison-Wesley (2000), ISBN:0-201-70002-6
- [11] J. Hromkovic. Design And Analysis Of Randomized Algorithms: Introduction to Design Paradigms. Springer-Verlag (2005), ISBN:3-540-23949-9
- [12] H. Samet. Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann (2006), ISBN:0-123-69446-9
- [13] M. A. Weiss. Data Structures and Algorithm Analysis in C++ (3rd edn.). Addison Wesley (2006), ISBN:0-321-44146-1
- [14] E. Horowitz, S. Sahni, D. Mehta. Fundamentals of Data Structures in C++ (2nd edn.). Silicon Press (2006), ISBN:0-929-30637-6
- [15] A. Drozdek. Data Structures and Algorithms in C++ (2nd edn.). Thomson Press (2006), ISBN:8-131-50115-9
- [16] 殷人昆 等. 数据结构 (C++语言描述). 北京: 清华大学出版社, 2007年6月第2版, ISBN:7-302-14811-1
- [17] P. Brass. Advanced Data Structures. Cambridge University Press, ISBN:0-521-88037-8
- [18] J. Edmonds. How to Think about Algorithms. Cambridge University Press (2008), ISBN:0-521-61410-8
- [19] K. Mehlhorn & P. Sanders. Algorithms and Data Structures: The Basic Tools. Springer (2008), ISBN:3-540-77977-9
- [20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. Introduction to Algorithms (3rd edn.). MIT Press (2009), ISBN:0-262-03384-4

- [21] R. Bird. *Pearls of Functional Algorithm Design*. Cambridge University Press (2010), ISBN:0-521-51338-8
- [22] M. L. Hetland. *Python Algorithms: Mastering Basic Algorithms in the Python Language*. Apress (2010), ISBN:1-430-23237-4
- [23] M. T. Goodrich, R. Tamassia, D. M. Mount. *Data Structures and Algorithms in C++* (2nd edn.). John Wiley & Sons (2011), ISBN:0-470-38327-5
- [24] R. Sedgewick & K. Wayne. *Algorithms* (4th edn.). Addison-Wesley (2011), ISBN:0-321-57351-X
- [25] Y. Perl, A. Itai and H. Avni, Interpolation Search: A  $\log(\log(n))$  Search, *Commun. ACM*, 21 (1978), pp. 550-553
- [26] A. C. Yao & F. F. Yao. The Complexity of Searching an Ordered Random Table. 17th Annual Symposium on Foundations of Computer Science (1976), 173-177
- [27] A. C. Yao & J. M. Steele. Lower Bounds to Algebraic Decision Trees. *Journal of Algorithms* (1982), 3:1-8
- [28] A. C. Yao. Lower Bounds for Algebraic Computation Trees with Integer Inputs. *SIAM J. On Computing* (1991), 20:655-668
- [29] L. Devroye. A Note on the Height of Binary Search Trees. *J. of ACM* (1986), 33(3):489-498
- [30] P. Flajolet & A. Odlyzko. The Average Height of Binary Trees and Other Simple Trees. *Journal of Computer and System Sciences* (1982), 25(2):171-213
- [31] J. B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proc. of the American Mathematical Society*, 7(1):48-50
- [32] B. W. Arden, B. A. Galler, R. M. Graham. An Algorithm for Equivalence Declarations. *Communications ACM* (1961), 4:310-314
- [33] B. A. Galler, M. J. Fisher. An Improved Equivalence Algorithm. *Communications ACM* (1964), 7:301-303
- [34] R. E. Tarjan. Efficiency of a Good but not Linear Set Union Algorithm. *Journal of the ACM* (1975), 22:215-225
- [35] R. Seidel & M. Sharir. Top-Down Analysis of Path Compression. *SIAM Journal Computing* (2005), 34:515-525
- [36] G. Adelson-Velskii & E. M. Landis. An Algorithm for the Organization of Information. *Proc. of the USSR Academy of Sciences* (1962), 146:263-266
- [37] D. S. Hirschberg. An Insertion Technique for One-Sided Heightbalanced Trees. *Comm. ACM* (1976), 19(8):471-473
- [38] S. H. Zweben & M. A. McDonald. An Optimal Method for Deletion in One-Sided Height-Balanced Trees. *Commun. ACM* (1978), 21(6):441-445
- [39] K. Culik, T. Ottman, D. Wood. Dense Multiway Trees. *ACM Transactions on Database Systems* (1981), 6:486-512
- [40] E. Gudes & S. Tsur. Experiments with B-tree Reorganization. *SIGMOD* (1980), 200-206
- [41] D. D. Sleator & R. E. Tarjan. Self-Adjusting Binary Trees. *JACM* (1985), 32:652-686

- [42] R. E. Tarjan. Amortized Computational Complexity. *SIAM. J. on Algebraic and Discrete Methods* 6(2):306-318
- [43] R. Bayer & E. McCreight. Organization and Maintenance of Large Ordered Indexes. *Acta Informatica* (1972), 1(3):173-189
- [44] R. Bayer. Symmetric Binary B-Trees: Data Structure and Maintenance Algorithms. *Acta Informatica* (1972), 1(4):290-306
- [45] L. J. Guibas & R. Sedgewick. A Dichromatic Framework for Balanced Trees. *Proc. of the 19th Annual Symposium on Foundations of Computer Science* (1978), 8-21
- [46] J. L. Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM* (1975), 18(9):509-517
- [47] H. J. Olivie. A New Class of Balanced Search Trees: Half Balanced Binary Search Trees. *ITA* (1982), 16(1):51-71
- [48] J. L. Bentley. Decomposable Searching Problems. *Information Processing Letters* (1979), 8:244-251
- [49] J. H. Hart. Optimal Two-Dimensional Range Queries Using Binary Range Lists. Technical Report 76-81, Department of Computer Science, University of Kentucky (1981)
- [50] D. E. Willard. New Data Structures for Orthogonal Range Queries. *SIAM Journal on Computing* (1985), 14:232-253
- [51] H. Samet, An Overview of Quadrees, Octrees, and Related Hierarchical Data Structures, in R. Earnshaw, ed., *Theoretical Foundations of Computer Graphics and Cad*, Springer Berlin Heidelberg, 1988, pp. 51-68
- [52] W. Pugh. Skip Lists: a Probabilistic Alternative to Balanced Trees. *Lecture Notes in Computer Science* (1989), 382:437-449
- [53] R. de la Briandais. File Searching Using Variable Length Keys. *Proc. of the Western Joint Computer Conference* 1959, 295-298
- [54] E. H. Sussenguth. Use of Tree Structures for Processing Files. *Communications of the ACM* (1963), 6:272-279
- [55] D. R. Morrison. PATRICIA - Practical Algorithm to Retrieve Information Coded in Alphanumeric. *Journal of the ACM* (1968), 15:514-534
- [56] J. L. Bentley & R. Sedgewick. Fast Algorithms for Sorting and Searching Strings. *Proc. of 8th ACM-SIAM Symposium on Discrete Algorithms* (1997), 360-369
- [57] R. W. Floyd. Algorithm 113: Treesort. *Communications of the ACM* (1962), 5:434
- [58] C. A. Crane. Linear Lists and Priority Queues as Balanced Binary Trees. PhD thesis, Stanford University (1972)
- [59] E. M. McCreight. Priority Search Trees. *SIAM J. Comput.* (1985), 14(2):257-276
- [60] D. E. Knuth, J. H. Morris, V. R. Pratt. Fast Pattern Matching in Strings. *SIAM Journal of Computing* (1977), 6(2):323-350

- [61] R. S. Boyer & J. S. Moore. A Fast String Searching Algorithm. *Communications of the ACM* (1977), 20:762-772
- [62] L. J. Guibas & A. M. Odlyzko. A New Proof of the Linearity of the Boyer-Moore String Search Algorithm. *SIAM Journal on Computing* (1980), 9(4):672-682
- [63] R. Cole. Tight Bounds on the Complexity of the Boyer-Moore Pattern Matching Algorithm. *SIAM Journal on Computing* 23(5):1075-1091
- [64] C. A. R. Hoare. Quicksort. *Computer Journal* (1962), 5(1):10-15
- [65] D. L. Shell. A High-Speed Sorting Procedure. *Communications of the ACM* (1959), 2(7):30-32
- [66] R. Sedgewick, A New Upper Bound for Shellsort, *J. Algorithms*, 7 (1986), pp. 159-173

## 插图索引

图x1.1 过直线外一点作其平行线 .....	2
图x1.2 《海岛算经》算法原理 .....	3
图x1.3 fib()算法中递归实例fib(k)的两种出现可能 .....	13
图x1.4 使用85块L形积木,可以恰好覆盖缺失一角的16×16棋盘 .....	18
图x1.5 采用分治策略,将大棋盘的覆盖问题转化为四个小棋盘的覆盖问题 .....	18
图x1.6 借助reverse()算法在O(n)时间内就地移位的过程及原理 .....	21
图x1.7 reverse()算法的递归跟踪 .....	24
图x1.8 power2()算法的递归跟踪 .....	24
图x1.9 二重循环执行时间的对应图形 .....	25
图x1.10 联合递归函数F(n)和G(n)的递归跟踪图 .....	31
图x2.1 permute()算法中第k + 1个就位元素,应等概率地随机选自当时的前n - k个元素 .....	38
图x2.2 Brian W. Kernighan和Dennis M. Ritchie所设计随机数发生器的原理 .....	39
图x2.3 无序向量删除算法remove(lo, hi)中,采用自后向前的次序移动可能造成数据丢失 .....	41
图x2.4 从问题A到问题B的线性归约 .....	43
图x2.5 输入规模为4时的归并排序过程 .....	45
图x2.6 二分查找binSearch()算法版本A所对应的比较树,在向量规模递增后的结构变化 .....	47
图x2.7 二分查找失败情况的递归分类 .....	49
图x2.8 Fibonacci查找失败情况的递归分类 .....	50
图x2.9 马鞍查找算法的原理及过程 .....	51
图x2.10 B[]和C[]中的元素均未耗尽,且已转入A[]的元素总数 $i \leq lb$ .....	59
图x2.11 B[]和C[]中的元素均未耗尽,且已转入A[]的元素总数 $i > lb$ .....	59
图x2.12 B[]中的元素先于C[]耗尽 .....	59
图x2.13 C[]中的元素先于B[]耗尽 .....	59
图x2.14 通过引入两个等长的向量,在O(1)时间内初始化Bitmap对象 .....	64
图x2.15 Eratosthenes算法的实例 .....	67
图x2.16 Eratosthenes算法:每次迭代中所筛选的整数,恰好就是重排矩形的最右侧一列 .....	67
图x2.17 Eratosthenes算法的改进 .....	67
图x2.18 4×5的矩阵实例:经逐列排序再逐行排序后,每行、每列均各自有序 .....	70
图x2.19 只需考查沿纵向捉对有序的任意两行 .....	70
图x2.20 起泡排序的每一步,都是考查一对相邻元素 .....	70
图x2.21 只需考查仅有一行进行交换的两种情况:(a) a和x交换;(b) b和y交换 .....	71
图x2.22 假设逐行排序之后,沿纵向出现一对逆序元素a和b .....	71
图x4.1 迷宫算法低效的实例 .....	99
图x5.1 $n_h$ 个底层(叶)节点删除后,(次底层叶)节点至少增加 $\lceil n_h/2 \rceil$ 个 .....	105
图x5.2 迭代式先序遍历实例(出栈节点以深色示意) .....	108

图x5.3 BinNode::succ()的情况一：t拥有右后代，其直接后继为右子树中左分支的末端节点s .....	110
图x5.4 BinNode::succ()的情况二：t没有右后代，其直接后继为以其为直接前驱的祖先s .....	110
图x5.5 二叉树中序遍历过程中对succ()接口的调用 .....	111
图x5.6 频率最低的兄弟节点合并之后，最优编码树必对应于合并之前的最优编码树 .....	117
图x5.7 字符串集{ "how", "many", "roads", "must", "a", "man", "walk", "down" }对应的键树 .....	118
图x5.8 键树的紧凑表示与实现 .....	118
图x6.1 (a)平面图、(b)三角剖分以及(c)外面亦为三角形的三角剖分 .....	121
图x6.2 将5×5的对称矩阵压缩至长度为15的一维向量 .....	122
图x6.3 图BFS搜索，等效于BFS树的层次遍历 .....	124
图x6.4 偏心率实例：(a) 单个顶点；(b) 两个顶点的完全图；(c)周长为奇数( 5 )的环形；(d) Petersen图( Petersen's Graph )；(e) 13个顶点、12条边构成的树；(f) 13个顶点、18条边构成的图 .....	126
图x6.5 无论如何，BFS在树中最后访问的顶点u必是边缘点（还有一些情况，比如a可能就是v） .....	127
图x6.6 构造有向图的欧拉环路：各子环路加粗示意，删除的边不再画出，删除的顶点以灰色示意 .....	128
图x6.7 最小支撑树（粗线条）可能反复地穿越于割的两侧 .....	132
图x6.8 借助最小支撑树，构造近似的旅行商环路（严格地说，w还不是一条旅行商环路——它经过各顶点至少两次，而非恰好一次。为此只需再次遍历该环路，沿途一旦试图再次访问某节点，则将相邻的两条边替换为一段“捷径”。如此不仅能够得到一条严格意义上的环路apprTST（虚线），而且总长度亦不致增加。） .....	135
图x6.9 含负权边时，即便不存在负权环路，Dijkstra算法依然可能出错：设起点为S，算法将依次确定A、B对应的最短距离分别为2、3。而实际上，从S绕道B通往A的距离为 $3 + (-2) = 1 < 2$ 。这里的关键在于，在顶点A所对应的最短路径上，有另一顶点B被算法发现得更晚。 .....	136
图x6.10 完全由极短跨越边构成的支撑树，未必是极小的 .....	137
图x6.11 在各边权重未必互异时，Prim算法依然正确 .....	138
图x6.12 Krusal算法的正确性 .....	139
图x6.13 Krusal算法的最坏情况 .....	139
图x6.14 并查集：初始时每个元素逻辑上自成一个子集，并分别对应于一棵多叉树，parent统一取作-1 .....	140
图x6.15 并查集：经过union(D, F)和union(G, B)，F所属的子集（树）被归入D所属的子集（树），沿用标识D；B所属的子集（树）被归入G所属的子集（树），沿用标识G .....	140
图x6.16 并查集：再经union(D, A)和union(F, H)，A和H被归入子集D中；再经union(E, C)，C被归入子集E中 .....	141
图x6.17 并查集：再经union(B, A)，A所属的子集（树）D，被归入B所属的子集（树）G中 .....	141
图x6.18 并查集：再经union(C, F)，F所属的子集（树）G，被归入C所属子集(树) E中（树高未能有效控制） .....	141
图x6.19 并查集：经union(C, F)操作之后（“低者归入高者”以控制树高） .....	142
图x6.20 各边权重互异时，最短路径树依然可能不唯一 .....	142
图x6.21 平面点集的三角剖分（a），及其特例Delaunay三角剖分（b） .....	143
图x6.22 平面点集的邻近图：(a)Gabriel图，(b) RNG图，(c) 欧氏最小支撑树 .....	144
图x7.1 AVL树中最浅的叶节点 .....	150
图x7.2 Fib-AVL树 .....	150
图x7.3 从AVL-树中删除节点之后，需要重平衡的祖先未必相邻 .....	154
图x7.4 将31个关键码按单调次序插入，必然得到一棵高度为4的满树 .....	155

图x8.1 高度 $h = 4$ 、由53个节点组成的一棵5阶B-树 .....	161
图x8.2 高度 $h = 4$ 、由79个节点组成的一棵5阶B-树 .....	161
图x8.3 按递增插入 $[0, 52)$ 而生成的5阶B-树 .....	162
图x8.4 高度(计入扩充的外部节点)为10的红黑树,至少包含62个节点 .....	168
图x8.5 高度(计入扩充的外部节点)为9的红黑树,至少包含46个节点 .....	169
图x8.6 统计与查询区域边界相交的子区域(节点)总数 .....	171
图x8.7 每次切分之后,都即将子区域(实线)替换为包围盒(虚线),以加速此后的查找 .....	172
图x8.8 通过递归地将平面子区域均分为四个象限(左),构造对应的四叉树(右) .....	173
图x8.9 四叉树的空间利用率可能极低 .....	173
图x8.10 利用范围树,可以实现更加高效的范围查询 .....	176
图x8.11 通过分散层叠,进一步提高范围树的查找性能 .....	178
图x9.1 表长与公差有非平凡公因子时,会出现大量的冲突 .....	182
图x9.2 双向平方试探法 .....	186
图x9.3 将关键词{ 2012, 10, 120, 175, 190, 230 },依次插入长度为11的散列表 .....	188
图x9.4 删除关键词2012,并做懒惰删除标记 .....	188
图x9.5 PATRICIA树(PATRICIA tree) .....	194
图x9.6 三叉键树(ternary trie) .....	194
图x10.1 字符权重已排序时,可在线性时间内构造出Huffman编码树 .....	200
图x10.2 三叉堆:(a)逻辑结构及(b)物理结构 .....	202
图x10.3 优先级搜索树 .....	204
图x10.4 基于优先级搜索树的半无穷范围查询算法 .....	205
图x10.5 高效支持getMax()接口的栈 .....	206
图x10.6 高效支持getMax()接口的队列 .....	208
图x10.7 合并AVL树S和T:不妨假定 $g \geq h$ .....	208
图x10.8 合并AVL树S和T:删除T中的最小节点m,在S的最右侧通路上找到与树T'高度接近的节点u .....	209
图x10.9 合并AVL树S和T:以m为结合点合并S'和T',在整体接入至S .....	209
图x10.10 以任意关键为界,分裂AVL树(这里只是示意性地绘出了各子树,并未严格地反映其高度) .....	210
图x11.1 BM算法的最坏情况 .....	216
图x12.1 轴点构造算法(版本C) .....	220
图x12.2 在向量X和Y各自排序后,对齐元素之间的次序依然保持 .....	226
图x12.3 (a)排序前有 $Y[0, 3) \leq X[4, 7)$ , (b)排序后仍有 $Y'[0, 3) \leq X'[4, 7)$ .....	226
图x12.4 g-有序的向量A[]再经h-排序后, $A[i + g]$ 必然来自阴影区域 .....	227
图x12.5 g-有序的向量A[]按照h列重排之后, $A[i]$ 所属列的前缀,必然与 $A[i + g]$ 所属列的后缀,逐个元素地对应有序 .....	227



表格索引

表x1.1 函数F(n)中变量i和j随迭代不断递增的过程 ..... 28

表x1.2 函数F(n)中变量i和r随迭代不断递增的过程 ..... 28

表x3.1 列表{ 61, 60, 59, ..., 5, 4, 3, 2, 0, 1, 2 }的插入排序过程 ..... 80

表x4.1 表达式求值算法实例 ..... 94

表x4.2 (左)括号数固定时,运算符栈的最大规模 ..... 96

表x4.3 非法表达式"(12)3+!4\*+5"的“求值”过程 ..... 96

表x9.1 n个人中存在生日巧合的概率..... 183

表x9.2 对序列{ 5a, 2a, 3, 2b, 9a, 5b, 9b, 8, 2c }的直接计数排序 ..... 190

表x9.3 借助散列表对{ 5a, 2a, 3, 2b, 9a, 5b, 9b, 8, 2c }的计数排序(凡“-”项均与其上方项相等) .. 192

# 算法索引

算法x1.1 过直线外一点作其平行线..... 2

算法x1.2 缺角棋盘的覆盖算法 ..... 18

算法x2.1 马鞍查找 ..... 51

算法x4.1 确认不含任何禁形的序列都是栈混洗 ..... 89

算法x6.1 基于“反复删除零入度节点”策略的拓扑排序算法 ..... 133

算法x9.1 整数向量的计数排序算法..... 191

算法x10.1 基于优先级搜索树的半无穷范围查询算法..... 205

## 代码索引

代码x1.1 《海岛算经》中计算海岛高度的算法 .....	3
代码x1.2 《海岛算经》中计算海岛距离的算法 .....	3
代码x1.3 包含循环、分支、子函数调用甚至递归结构，但具有常数时间复杂度的算法 .....	6
代码x1.4 countOnes()算法的改进版 .....	8
代码x1.5 countOnes()算法的再改进版 .....	9
代码x1.6 power2BF_I()算法的递归版 .....	10
代码x1.7 power2()算法的迭代版 .....	10
代码x1.8 通用的迭代版幂函数算法 .....	11
代码x1.9 数组最大值算法（迭代版） .....	11
代码x1.10 数组最大值算法（线性递归版） .....	11
代码x1.11 数组最大值算法（二分递归版） .....	12
代码x1.12 Fib类的实现 .....	16
代码x1.13 Hanoi塔算法 .....	17
代码x1.14 运用“中华更相减损术”的最大公约数算法 .....	20
代码x1.15 借助reverse()算法在O(n)时间内就地移位 .....	21
代码x1.16 计算Hailstone(n)序列长度的“算法” .....	23
代码x2.1 基于遍历实现向量的decrease()功能 .....	44
代码x2.2 基于遍历实现向量的double()功能 .....	45
代码x2.3 向量的起泡排序（改进版） .....	56
代码x2.4 单趟扫描交换（改进版） .....	56
代码x2.5 有序向量二路归并算法的简化 .....	60
代码x2.6 增加注释后，Python的bisect模块中bisect_right接口的源代码 .....	60
代码x2.7 位图Bitmap类 .....	62
代码x2.8 可快速初始化的Bitmap对象（仅支持set()操作） .....	63
代码x2.9 可快速初始化的Bitmap对象（兼顾set()和clear()操作） .....	65
代码x2.10 Eratosthenes素数筛选算法 .....	66
代码x3.1 基于遍历实现列表的increase()功能 .....	76
代码x3.2 基于遍历实现列表的half()功能 .....	76
代码x3.3 向量的选择排序算法 .....	77
代码x3.4 列表倒置算法的第一种实现 .....	85
代码x3.5 列表倒置算法的第二种实现 .....	85
代码x3.6 列表倒置算法的第三种实现 .....	85
代码x4.1 由List类派生Stack类 .....	88
代码x4.2 操作数的解析 .....	92
代码x4.3 运算符优先级关系的判定 .....	93

代码x4.4 将操作数或操作符统一接至RPN表达式末尾.....	93
代码x4.5 PostScript语言的绘图程序.....	98
代码x5.1 二叉树先序遍历算法（迭代版#1）.....	108
代码x5.2 二叉树中序遍历算法（迭代版#4）.....	112
代码x6.1 基于PFS框架的BFS优先级更新器.....	134
代码x6.2 基于PFS框架的DFS优先级更新器.....	135
代码x7.1 二叉搜索树searchIn()算法的迭代实现.....	147
代码x7.2 将任意一棵二叉搜索树等价变换为单分支列表.....	152
代码x12.1 轴点构造算法（版本C）.....	221

关键词索引

(按关键词中各汉字的声母及各英文单词的首字母排序，比如“大O记号”对应于“DOJH”)

A

凹函数 (concave function) ..... 7, 159

埃拉托斯特尼的筛子 (the sieve of Eratosthenes) ..... 66

B

并查集 (union-find set) ..... 140

八叉树 (octree) ..... 175

半径 (radius) ..... 126

比较树 (comparison tree) ..... 45, 46, 47, 68, 107

伯努利实验 (Bernoulli trial) ..... 217

半平衡二叉搜索树 (half-balanced binary search trees) ..... 167

包围盒 (bounding-box) ..... 172

半无穷范围查询 (semi-infinite range query) ..... 203

$B^*$ -树 ( $B^*$ -tree) ..... 166

边缘点 (peripheral vertex) ..... 126, 127

标志 (tag) ..... 91

C

差分约束系统 (system of difference constraints) ..... 120

常数代价准则 (uniform cost criterion) ..... 15

超限数学归纳法 (transfinite induction) ..... 22

插值查找 (interpolation search) ..... 53

D

多层搜索树 (multi-level search tree) ..... 178

队堆 (queap) ..... 207

递归跟踪 (recursion trace) ..... 12, 24, 31

独立集 (disjoint set) ..... 140

独立性 (independence) ..... 40

对数代价准则 (logarithmic cost criterion)	15
Dijkstra算法 (Dijkstra Algorithm)	136, 202
Delaunay三角剖分 (Delaunay triangulation)	143
代数判定树 (algebraic decision tree, ADT)	68
队头 (front)	101
递推方程 (recurrence equation)	13, 23, 31, 55, 69, 84
队尾 (rear)	101

E

二分查找 (binary search)	46, 47, 48
二分递归 (binary recursion)	32, 107
$\epsilon$ -间距问题 ( $\epsilon$ -closeness)	144

F

Fib-AVL树 (Fibonacci AVL tree)	150, 153
封底估算 (back-of-the-envelope calculation)	5, 66
分而治之 (divide-and-conquer)	32, 225
费马平方和定理 (Two-Square Theorem of Fermat)	187
分散层叠 (fractional cascading)	178
Floyd算法 (Floyd Algorithm)	201
分摊分析 (amortized analysis)	36, 37, 158, 201
范围树 (range tree)	176, 204

G

割 (cut)	131, 132, 137, 138, 139
归并排序 (mergesort)	45
高度 (height)	105
关联树 (associative tree)	176
关联矩阵 (incidence matrix)	120
Gabriel图 (Gabriel graph)	143

H

合成数 (composite number)	136, 201
多叉堆 (d-heap)	202

好后缀 (good suffix) ..... 214

后进先出 (last-in-first-out, LIFO) ..... 108

活跃期 (active duration) ..... 129, 130

J

减而治之 (decrease-and-conquer) ..... 51, 225

几何分布 (geometric distribution) ..... 199, 217

键树 (trie) ..... 118, 193, 194

计数排序 (counting sort) ..... 190

禁形 (forbidden pattern) ..... 89

基于比较式算法 (comparison-based algorithm, CBA) ..... 68, 107, 199

K

kd-树 (kd-tree) ..... 52

咖啡罐游戏 (Coffee Can Game) ..... 22

Kruskal算法 (Kruskal Algorithm) ..... 139, 140, 141, 143

快速排序 (quicksort) ..... 36

k-选取 (k-selection) ..... 222

空圆性质 (empty-circle property) ..... 143

L

鲁棒性 (robustness) ..... 96

路径压缩 (path compression) ..... 142

邻接表 (adjacency list) ..... 122, 189

邻接矩阵 (adjacency matrix) ..... 120

邻近图 (proximity graph) ..... 143

良序 (well order) ..... 22

旅行商环路 (traveling salesman tour) ..... 135

M

马鞍查找 (saddleback search) ..... 51

末节点 (last node) ..... 74, 86, 111

模式枚举 (pattern enumeration) ..... 215

N

NP完全的 (NP-complete)	12
逆序对 (inversion)	4, 79

O

欧拉环路 (Eulerian tour)	128
欧拉通路 (Eulerian path)	128
欧氏最小支撑树 (Euclidean Minimum Spanning Tree, EMST)	143

P

PATRICIA树 (PATRICIA tree)	194
Prim算法 (Prim Algorithm)	131, 136, 138, 143, 202
Petersen图 (Petersen's Graph)	126
偏心率 (eccentricity)	126

Q

起泡排序 (bubblesort)	5
期望值的线性律 (linearity of expectation)	33, 78, 83
前沿集 (frontier)	124

R

RNG图 (relative neighborhood graph)	144
------------------------------------	-----

S

哨兵节点 (sentinel node)	74, 196
三叉键树 (ternary trie)	194
四叉树 (quadtree)	52, 173, 194
树堆 (treap)	204
双端队列 (deque)	101
随机存储器 (Random Access Machine, RAM)	20
首节点 (first node)	74, 86, 153
数据局部性 (data locality)	76, 166



三角剖分 (triangulation) .....	121, 143
随机生成 (randomly generated by) .....	148
随机算法 (randomized algorithm) .....	33
随机组成 (randomly composed of) .....	148
散列表 (hashtable) .....	62, 182, 183, 184, 185, 188, 189
势能 (potential) .....	158
势能分析法 (potential analysis) .....	36, 158
输入敏感的 (input sensitive) .....	79
伸展树 (splay tree) .....	36

## T

凸函数 (convex function) .....	7
图灵机 (Turing Machine, TM) .....	20
跳转表 (skip list) .....	180

## W

尾递归 (tail recursion) .....	223
稳定性 (stability) .....	191
外面 (outer face) .....	121
完全二叉堆 (complete binary heap) .....	196
完全二叉树 (complete binary tree) .....	146

## X

斜堆 (skew heap) .....	200
循环节 (cycle) .....	82
循环列表 (Circular list) .....	86
先进先出 (first-in-first-out, FIFO) .....	149, 153, 180
小 $\omega$ 记号 (small-omega notation) .....	43
小 $O$ 记号 (small-O notation) .....	43
循位置访问 (call-by-position) .....	74, 78
线性递归 (linear recursion) .....	29, 30, 31
线性规划 (linear programming) .....	120
线性时间归约 (linear-time reduction) .....	43, 75, 199
循秩访问 (call-by-rank) .....	62, 74

## Y

叶节点平均深度 (average leaf depth) .....	107
掩码 (mask) .....	62
页面缓冲池 (buffer pool of pages) .....	166
元素唯一性 (Element Uniqueness) .....	44, 75
优先级搜索 (Priority-First Search, PFS) .....	202, 203
优先级搜索树 (priority search tree, PST) .....	204
元字符 (meta-character) .....	97

## Z

最短路径树 (shortest-path tree) .....	134, 136, 142
字典序 (lexicographical order) .....	45, 46, 136
真二叉树 (proper binary tree) .....	105, 107, 116
直径 (diameter) .....	126
最近点对 (nearest pair) .....	144
最近邻图 (nearest neighbor graph) .....	144
主树 (main tree) .....	176
众数 (majority) .....	222
指数查找 (exponential search) .....	50
自调整列表 (self-adjusting list) .....	76
中位点 (median point) .....	170, 173
中位数 (median) .....	222
中心点 (central vertex或center) .....	126
中缀表达式 (infix) .....	97
最左侧通路 (leftmost path) .....	151
最左最低点 (leftmost-then-lowest point) .....	22

习题汇总

第1章	第2章	第3章	第4章	第5章	第6章	第7章	第8章	第9章	第10章	第11章	第12章
[1-1]	[2-1]	[3-1]	[4-1]	[5-1]	[6-1]	[7-1]	[8-1]	[9-1]	[10-1]	[11-1]	[12-1]
[1-2]	[2-2]	[3-2]	[4-2]	[5-2]	[6-2]	[7-2]	[8-2]	[9-2]	[10-2]	[11-2]	[12-2]
[1-3]	[2-3]	[3-3]	[4-3]	[5-3]	[6-3]	[7-3]	[8-3]	[9-3]	[10-3]	[11-3]	[12-3]
[1-4]	[2-4]	[3-4]	[4-4]	[5-4]	[6-4]	[7-4]	[8-4]	[9-4]	[10-4]	[11-4]	[12-4]
[1-5]	[2-5]	[3-5]	[4-5]	[5-5]	[6-5]	[7-5]	[8-5]	[9-5]	[10-5]	[11-5]	[12-5]
[1-6]	[2-6]	[3-6]	[4-6]	[5-6]	[6-6]	[7-6]	[8-6]	[9-6]	[10-6]	[11-6]	[12-6]
[1-7]	[2-7]	[3-7]	[4-7]	[5-7]	[6-7]	[7-7]	[8-7]	[9-7]	[10-7]	[11-7]	[12-7]
[1-8]	[2-8]	[3-8]	[4-8]	[5-8]	[6-8]	[7-8]	[8-8]	[9-8]	[10-8]	[11-8]	[12-8]
[1-9]	[2-9]	[3-9]	[4-9]	[5-9]	[6-9]	[7-9]	[8-9]	[9-9]	[10-9]	[11-9]	[12-9]
[1-10]	[2-10]	[3-10]	[4-10]	[5-10]	[6-10]	[7-10]	[8-10]	[9-10]	[10-10]	[11-10]	[12-10]
[1-11]	[2-11]	[3-11]	[4-11]	[5-11]	[6-11]	[7-11]	[8-11]	[9-11]	[10-11]		[12-11]
[1-12]	[2-12]	[3-12]	[4-12]	[5-12]	[6-12]	[7-12]	[8-12]	[9-12]	[10-12]		[12-12]
[1-13]	[2-13]	[3-13]	[4-13]	[5-13]	[6-13]	[7-13]	[8-13]	[9-13]	[10-13]		[12-13]
[1-14]	[2-14]	[3-14]	[4-14]	[5-14]	[6-14]	[7-14]	[8-14]	[9-14]	[10-14]		[12-14]
[1-15]	[2-15]	[3-15]	[4-15]	[5-15]	[6-15]	[7-15]	[8-15]	[9-15]	[10-15]		
[1-16]	[2-16]	[3-16]	[4-16]	[5-16]	[6-16]	[7-16]	[8-16]	[9-16]	[10-16]		
[1-17]	[2-17]	[3-17]	[4-17]	[5-17]	[6-17]	[7-17]	[8-17]	[9-17]	[10-17]		
[1-18]	[2-18]	[3-18]	[4-18]	[5-18]	[6-18]	[7-18]	[8-18]	[9-18]	[10-18]		
[1-19]	[2-19]	[3-19]	[4-19]	[5-19]	[6-19]	[7-19]	[8-19]	[9-19]	[10-19]		
[1-20]	[2-20]		[4-20]	[5-20]	[6-20]	[7-20]	[8-20]	[9-20]	[10-20]		
[1-21]	[2-21]		[4-21]	[5-21]	[6-21]			[9-21]	[10-21]		
[1-22]	[2-22]		[4-22]	[5-22]	[6-22]			[9-22]	[10-22]		
[1-23]	[2-23]		[4-23]	[5-23]	[6-23]			[9-23]			
[1-24]	[2-24]		[4-24]	[5-24]	[6-24]			[9-24]			
[1-25]	[2-25]		[4-25]	[5-25]	[6-25]			[9-25]			
[1-26]	[2-26]		[4-26]	[5-26]	[6-26]			[9-26]			
[1-27]	[2-27]			[5-27]	[6-27]						
[1-28]	[2-28]			[5-28]	[6-28]						
[1-29]	[2-29]			[5-29]	[6-29]						
[1-30]	[2-30]			[5-30]	[6-30]						
[1-31]	[2-31]				[6-31]						
[1-32]	[2-32]				[6-32]						
	[2-33]				[6-33]						
	[2-34]										
	[2-35]										
	[2-36]										
	[2-37]										
	[2-38]										
	[2-39]										
	[2-40]										
	[2-41]										