

---

# Appendix

## Mathematical Background

# A

---

### Sum Formulas

Each sum of the form

$$\sum_{x=1}^n x^k = 1^k + 2^k + 3^k + \dots + n^k,$$

where  $k$  is a positive integer has a closed-form formula that is a polynomial of degree  $k + 1$ . For example,<sup>1</sup>

$$\sum_{x=1}^n x = 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

and

$$\sum_{x=1}^n x^2 = 1^2 + 2^2 + 3^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6}.$$

An *arithmetic progression* is a sequence of numbers where the difference between any two consecutive numbers is constant. For example,

$$3, 7, 11, 15$$

is an arithmetic progression with constant 4. The sum of an arithmetic progression can be calculated using the formula

$$\underbrace{a + \dots + b}_{n \text{ numbers}} = \frac{n(a+b)}{2}$$

---

<sup>1</sup>There is even a general formula for such sums, called *Faulhaber's formula*, but it is too complex to be presented here.

where  $a$  is the first number,  $b$  is the last number, and  $n$  is the amount of numbers. For example,

$$3 + 7 + 11 + 15 = \frac{4 \cdot (3 + 15)}{2} = 36.$$

The formula is based on the fact that the sum consists of  $n$  numbers and the value of each number is  $(a + b)/2$  on average.

A *geometric progression* is a sequence of numbers where the ratio between any two consecutive numbers is constant. For example,

$$3, 6, 12, 24$$

is a geometric progression with constant 2. The sum of a geometric progression can be calculated using the formula

$$a + ak + ak^2 + \cdots + b = \frac{bk - a}{k - 1}$$

where  $a$  is the first number,  $b$  is the last number, and the ratio between consecutive numbers is  $k$ . For example,

$$3 + 6 + 12 + 24 = \frac{24 \cdot 2 - 3}{2 - 1} = 45.$$

This formula can be derived as follows. Let

$$S = a + ak + ak^2 + \cdots + b.$$

By multiplying both sides by  $k$ , we get

$$kS = ak + ak^2 + ak^3 + \cdots + bk,$$

and solving the equation

$$kS - S = bk - a$$

yields the formula.

A special case of a sum of a geometric progression is the formula

$$1 + 2 + 4 + 8 + \cdots + 2^{n-1} = 2^n - 1.$$

A *harmonic sum* is a sum of the form

$$\sum_{x=1}^n \frac{1}{x} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

An upper bound for a harmonic sum is  $\log_2(n) + 1$ . Namely, we can modify each term  $1/k$  so that  $k$  becomes the nearest power of two that does not exceed  $k$ . For example, when  $n = 6$ , we can estimate the sum as follows:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} \leq 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}.$$

This upper bound consists of  $\log_2(n) + 1$  parts ( $1, 2 \cdot 1/2, 4 \cdot 1/4$ , etc.), and the value of each part is at most 1.

## Sets

A *set* is a collection of elements. For example, the set

$$X = \{2, 4, 7\}$$

contains elements 2, 4, and 7. The symbol  $\emptyset$  denotes an empty set, and  $|S|$  denotes the size of a set  $S$ , i.e., the number of elements in the set. For example, in the above set,  $|X| = 3$ . If a set  $S$  contains an element  $x$ , we write  $x \in S$ , and otherwise we write  $x \notin S$ . For example, in the above set,  $4 \in X$  and  $5 \notin X$ .

New sets can be constructed using set operations:

- The *intersection*  $A \cap B$  consists of elements that are in both  $A$  and  $B$ . For example, if  $A = \{1, 2, 5\}$  and  $B = \{2, 4\}$ , then  $A \cap B = \{2\}$ .
- The *union*  $A \cup B$  consists of elements that are in  $A$  or  $B$  or both. For example, if  $A = \{3, 7\}$  and  $B = \{2, 3, 8\}$ , then  $A \cup B = \{2, 3, 7, 8\}$ .
- The *complement*  $\bar{A}$  consists of elements that are not in  $A$ . The interpretation of a complement depends on the *universal set*, which contains all possible elements. For example, if  $A = \{1, 2, 5, 7\}$  and the universal set is  $\{1, 2, \dots, 10\}$ , then  $\bar{A} = \{3, 4, 6, 8, 9, 10\}$ .
- The *difference*  $A \setminus B = A \cap \bar{B}$  consists of elements that are in  $A$  but not in  $B$ . Note that  $B$  can contain elements that are not in  $A$ . For example, if  $A = \{2, 3, 7, 8\}$  and  $B = \{3, 5, 8\}$ , then  $A \setminus B = \{2, 7\}$ .

If each element of  $A$  also belongs to  $S$ , we say that  $A$  is a *subset* of  $S$ , denoted by  $A \subset S$ . A set  $S$  always has  $2^{|S|}$  subsets, including the empty set. For example, the subsets of the set  $\{2, 4, 7\}$  are

$$\emptyset, \{2\}, \{4\}, \{7\}, \{2, 4\}, \{2, 7\}, \{4, 7\} \text{ and } \{2, 4, 7\}.$$

Some often used sets are  $\mathbb{N}$  (natural numbers),  $\mathbb{Z}$  (integers),  $\mathbb{Q}$  (rational numbers), and  $\mathbb{R}$  (real numbers). The set  $\mathbb{N}$  can be defined in two ways, depending on the situation: either  $\mathbb{N} = \{0, 1, 2, \dots\}$  or  $\mathbb{N} = \{1, 2, 3, \dots\}$ .

**Table A.1** Logical operators

$A$	$B$	$\neg A$	$\neg B$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0
1	0	0	1	0	1	0	0
1	1	0	0	1	1	1	1

There are several notations for defining sets. For example,

$$A = \{2n : n \in \mathbb{Z}\}$$

consists of all even integers, and

$$B = \{x \in \mathbb{R} : x > 2\}$$

consists of all real numbers that are greater than two.

---

## Logic

The value of a logical expression is either *true* (1) or *false* (0). The most important logical operators are  $\neg$  (*negation*),  $\wedge$  (*conjunction*),  $\vee$  (*disjunction*),  $\Rightarrow$  (*implication*), and  $\Leftrightarrow$  (*equivalence*). Table A.1 shows the meanings of these operators.

The expression  $\neg A$  has the opposite value of  $A$ . The expression  $A \wedge B$  is true if both  $A$  and  $B$  are true, and the expression  $A \vee B$  is true if  $A$  or  $B$  or both are true. The expression  $A \Rightarrow B$  is true if whenever  $A$  is true, also  $B$  is true. The expression  $A \Leftrightarrow B$  is true if  $A$  and  $B$  are both true or both false.

A *predicate* is an expression that is true or false depending on its parameters. Predicates are usually denoted by capital letters. For example, we can define a predicate  $P(x)$  that is true exactly when  $x$  is a prime number. Using this definition,  $P(7)$  is true but  $P(8)$  is false.

A *quantifier* connects a logical expression to the elements of a set. The most important quantifiers are  $\forall$  (*for all*) and  $\exists$  (*there is*). For example,

$$\forall x(\exists y(y < x))$$

means that for each element  $x$  in the set, there is an element  $y$  in the set such that  $y$  is smaller than  $x$ . This is true in the set of integers, but false in the set of natural numbers.

Using the notation described above, we can express many kinds of logical propositions. For example,

$$\forall x((x > 1 \wedge \neg P(x)) \Rightarrow (\exists a(\exists b(a > 1 \wedge b > 1 \wedge x = ab))))$$

means that if a number  $x$  is larger than 1 and not a prime number, then there are numbers  $a$  and  $b$  that are larger than 1 and whose product is  $x$ . This proposition is true in the set of integers.

---

## Functions

The function  $\lfloor x \rfloor$  rounds the number  $x$  down to an integer, and the function  $\lceil x \rceil$  rounds the number  $x$  up to an integer. For example,

$$\lfloor 3/2 \rfloor = 1 \text{ and } \lceil 3/2 \rceil = 2.$$

The functions  $\min(x_1, x_2, \dots, x_n)$  and  $\max(x_1, x_2, \dots, x_n)$  give the smallest and largest of values  $x_1, x_2, \dots, x_n$ . For example,

$$\min(1, 2, 3) = 1 \text{ and } \max(1, 2, 3) = 3.$$

The *factorial*  $n!$  can be defined by

$$\prod_{x=1}^n x = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$$

or recursively

$$\begin{aligned} 0! &= 1 \\ n! &= n \cdot (n - 1)! \end{aligned}$$

The *Fibonacci numbers* arise in many situations. They can be defined recursively as follows:

$$\begin{aligned} f(0) &= 0 \\ f(1) &= 1 \\ f(n) &= f(n - 1) + f(n - 2) \end{aligned}$$

The first Fibonacci numbers are

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots$$

There is also a closed-form formula for calculating Fibonacci numbers, which is sometimes called *Binet's formula*:

$$f(n) = \frac{(1 + \sqrt{5})^n - (1 - \sqrt{5})^n}{2^n \sqrt{5}}.$$

## Logarithms

The *logarithm* of a number  $x$  is denoted  $\log_b(x)$ , where  $b$  is the base of the logarithm. It is defined so that  $\log_b(x) = a$  exactly when  $b^a = x$ . The *natural logarithm*  $\ln(x)$  of a number  $x$  is a logarithm whose base is  $e \approx 2.71828$ .

A useful property of logarithms is that  $\log_b(x)$  equals the number of times we have to divide  $x$  by  $b$  before we reach the number 1. For example,  $\log_2(32) = 5$  because 5 divisions by 2 are needed:

$$32 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

The logarithm of a product is

$$\log_b(xy) = \log_b(x) + \log_b(y),$$

and consequently,

$$\log_b(x^n) = n \cdot \log_b(x).$$

In addition, the logarithm of a quotient is

$$\log_b\left(\frac{x}{y}\right) = \log_b(x) - \log_b(y).$$

Another useful formula is

$$\log_u(x) = \frac{\log_b(x)}{\log_b(u)},$$

using which it is possible to calculate logarithms to any base if there is a way to calculate logarithms to some fixed base.

---

## Number Systems

Usually, numbers are written in base 10, which means that the digits 0, 1, ..., 9 are used. However, there are also other number systems, like the base 2 binary system that has only two digits 0 and 1. In general, in a base  $b$  system, the integers 0, 1, ...,  $b - 1$  are used as digits.

We can convert a base 10 number to base  $b$  by dividing the number by  $b$  until it becomes zero. The remainders in reverse order correspond to the digits in base  $b$ . For example, let us convert the number 17 to base 3:

- $17/3 = 5$  (remainder 2)
- $5/3 = 1$  (remainder 2)
- $1/3 = 0$  (remainder 1)

Thus, the number 17 in base 3 is 122. Then, to convert a base  $b$  number to base 10, it suffices to multiply each digit by  $b^k$ , where  $k$  is the zero-based position of the digit starting from the right, and sum the results together. For example, we can convert the base 3 number 122 back to base 10 as follows:

$$1 \cdot 3^2 + 2 \cdot 3^1 + 2 \cdot 3^0 = 17$$

The number of digits of an integer  $x$  in base  $b$  can be calculated using the formula  $\lfloor \log_b(x) + 1 \rfloor$ . For example,  $\lfloor \log_3(17) + 1 \rfloor = 3$ .

---

## References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*, Pearson, 1993.
2. A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.
3. M. A. Bender and M. Farach-Colton. The LCA problem revisited. *Latin American Symposium on Theoretical Informatics*, 88–94, 2000.
4. J. Bentley and D. Wood. An optimal worst case algorithm for reporting intersections of rectangles. *IEEE Transactions on Computers*, C-29(7):571–577, 1980.
5. Codeforces: On “Mo’s algorithm”, <http://codeforces.com/blog/entry/20032>
6. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*, MIT Press, 2009 (3rd edition).
7. K. Diks et al. *Looking for a Challenge? The Ultimate Problem Set from the University of Warsaw Programming Competitions*, University of Warsaw, 2012.
8. D. Fanding. A faster algorithm for shortest-path – SPFA. *Journal of Southwest Jiaotong University*, 2, 1994.
9. P. M. Fenwick. A new data structure for cumulative frequency tables. *Software: Practice and Experience*, 24(3):327–336, 1994.
10. J. Fischer and V. Heun. Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE. *Annual Symposium on Combinatorial Pattern Matching*, 36–48, 2006.
11. F. Le Gall. Powers of tensors and fast matrix multiplication. *International Symposium on Symbolic and Algebraic Computation*, 296–303, 2014.
12. A. Grønlund and S. Pettie. Threesomes, degenerates, and love triangles. *Annual Symposium on Foundations of Computer Science*, 621–630, 2014.
13. D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.
14. S. Halim and F. Halim. *Competitive Programming 3: The New Lower Bound of Programming Contests*, 2013.
15. The International Olympiad in Informatics Syllabus, <https://people.ksp.sk/~misof/loi-syllabus/>

16. J. Kärkkäinen and P. Sanders. Simple linear work suffix array construction. *International Colloquium on Automata, Languages, and Programming*, 943–955, 2003.
17. R. M. Karp, R. E. Miller, and A. L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. *Annual ACM Symposium on Theory of Computing*, 125–135, 1972.
18. T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park. Linear-time longest-common-prefix computation in suffix arrays and its applications. *Annual Symposium on Combinatorial Pattern Matching*, 181–192, 2001.
19. J. Kleinberg and É. Tardos. *Algorithm Design*, Pearson, 2005.
20. D. E. Knuth. Optimum binary search trees. *Acta Informatica* 1(1):14–25, 1971.
21. S. Kopeliovich. Offline solution of connectivity and 2-edge-connectivity problems for fully dynamic graphs. MSc thesis, Saint Petersburg State University, 2012.
22. M. G. Main and R. J. Lorentz. An  $O(n \log n)$  algorithm for finding all repetitions in a string. *Journal of Algorithms*, 5(3):422–432, 1984.
23. J. Pachocki and J. Radoszewski. Where to use and how not to use polynomial string hashing. *Olympiads in Informatics*, 7(1):90–100, 2013.
24. D. Pearson. A polynomial-time algorithm for the change-making problem. *Operations Research Letters*, 33(3):231–234, 2005.
25. 27-Queens Puzzle: Massively Parallel Enumeration and Solution Counting. <https://github.com/preusser/q27>
26. M. I. Shamos and D. Hoey. Closest-point problems. *Annual Symposium on Foundations of Computer Science*, 151–162, 1975.
27. S. S. Skiena. *The Algorithm Design Manual*, Springer, 2008 (2nd edition).
28. S. S. Skiena and M. A. Revilla. *Programming Challenges: The Programming Contest Training Manual*, Springer, 2003.
29. D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, 1983.
30. P. Stańczyk. Algorytmika praktyczna w konkursach Informatycznych. MSc thesis, University of Warsaw, 2006.
31. V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.
32. F. F. Yao. Efficient dynamic programming using quadrangle inequalities. *Annual ACM Symposium on Theory of Computing*, 429–435, 1980.

# Index

## Symbols

2SAT problem, [192](#)  
2SUM problem, [112](#)  
3SAT problem, [193](#)

## A

Adjacency list, [80](#)  
Adjacency matrix, [81](#)  
Amortized analysis, [111](#)  
Ancestor, [137](#)  
And operation, [21](#)  
Andrew's algorithm, [224](#)  
Antichain, [207](#)  
Arithmetic progression, [1](#), [269](#)  
Articulation point, [208](#)

## B

Backtracking, [18](#)  
Bellman–Ford algorithm, [88](#)  
Biconnected graph, [207](#)  
Biconnectivity, [207](#)  
Binary indexed tree, [122](#)  
Binary search, [46](#)  
Binary search tree, [55](#)  
Binet's formula, [5](#), [273](#)  
Binomial coefficient, [157](#)  
Binomial distribution, [177](#)  
Bipartite graph, [80](#)  
Bipartiteness check, [87](#)  
Birthday paradox, [231](#)  
Bit mask, [22](#)  
Bit representation, [20](#)  
Bit shift, [22](#)  
Bit-parallel algorithm, [107](#)  
Bitset, [24](#)  
Border, [234](#)

Breadth-First Search (BFS), [85](#)  
Bridge, [208](#)  
Bubble sort, [38](#)  
Burnside's lemma, [163](#)

## C

Catalan number, [159](#)  
Cayley's formula, [164](#)  
Centroid, [144](#)  
Centroid decomposition, [144](#)  
Child, [131](#)  
Chinese remainder theorem, [155](#)  
Closest pair, [221](#)  
Coin change problem, [63](#)  
Collatz conjecture, [5](#)  
Collision, [230](#)  
Combinatorics, [156](#)  
Comparison function, [43](#)  
Comparison operator, [42](#)  
Complement, [3](#), [271](#)  
Complete graph, [79](#)  
Complex number, [211](#)  
Component, [78](#)  
Component graph, [189](#)  
Conditional probability, [175](#)  
Conjunction, [4](#), [272](#)  
Connected graph, [78](#)  
Connectivity check, [86](#)  
Constant factor, [31](#)  
Constant-time algorithm, [30](#)  
Convex function, [116](#)  
Convex hull, [224](#)  
Convex hull trick, [258](#)  
Coprime, [153](#)  
Counting sort, [41](#)  
Cross product, [213](#)

Cubic algorithm, 30  
Cut, 198  
Cycle, 78  
Cycle detection, 86, 94, 99

## D

Data structure, 51  
De Bruijn sequence, 196  
Degree, 78  
Depth-First Search (DFS), 83  
Depth-first search tree, 207  
Deque, 54  
Derangement, 162  
Diameter, 134  
Difference, 3, 271  
Difference array, 129  
Dijkstra's algorithm, 89  
Dilworth's theorem, 207  
Diophantine equation, 155  
Directed Acyclic Graph (DAG) 94  
Directed graph, 78  
Disjoint paths, 202  
Disjunction, 4, 272  
Distance function, 218  
Distribution, 177  
Divide and conquer optimization, 260  
Divisibility, 148  
Divisor, 148  
Dynamic array, 51  
Dynamic connectivity, 266  
Dynamic programming, 63  
Dynamic programming optimization, 258  
Dynamic segment tree, 249

## E

Edge, 78  
Edge list, 82  
Edit distance, 227  
Edmonds–Karp algorithm, 200  
Equivalence, 4, 272  
Euclid's algorithm, 151  
Euclidean distance, 218  
Euler tour tree, 141  
Euler's theorem, 154  
Euler's totient function, 153  
Eulerian circuit, 194  
Eulerian path, 194  
Eulerian subgraph, 209  
Expected value, 176  
Extended Euclid's algorithm, 152

## F

Factor, 148  
Factorial, 5, 273  
Faulhaber's formula, 1, 269  
Fenwick tree, 122  
Fermat's little theorem, 154  
Fibonacci number, 5, 167, 273  
Floating point number, 13  
Flow, 198  
Floyd's algorithm, 99  
Floyd–Warshall algorithm, 92  
Ford–Fulkerson algorithm, 199  
Functional graph, 97

## G

Game state, 181  
Game theory, 181  
Geometric distribution, 177  
Geometric progression, 2, 270  
Geometry, 211  
Graph, 78  
Graph coloring, 180  
Greatest common divisor, 151  
Greedy algorithm, 45  
Grundy number, 184  
Grundy's game, 186

## H

Hall's theorem, 203  
Hamiltonian circuit, 195  
Hamiltonian path, 195  
Hamming distance, 107  
Harmonic sum, 2, 150, 270  
Hash table, 55  
Hash value, 228  
Hashing, 228  
Heap, 58  
Heavy-Light decomposition, 145  
Hierholzer's algorithm, 195

## I

Identity matrix, 166  
Implication, 4, 272  
In-order, 133  
Inclusion-exclusion, 161  
Indegree, 79  
Independence, 175  
Independent set, 205  
Index compression, 128  
Input and output, 10  
Integer, 12

Integer partition, 243  
Intersection, 3, 271  
Intersection point, 220  
Inversion, 38  
Iterator, 53

**K**

König's theorem, 204  
Knapsack, 71, 243  
Knight's tour, 197  
Knuth's optimization, 261  
Kosaraju's algorithm, 190  
Kruskal's algorithm, 101

**L**

Las Vegas algorithm, 179  
Lazy propagation, 246  
Lazy segment tree, 246  
LCP array, 236  
Leaf, 131  
Levenshtein distance, 227  
Line segment intersection, 214  
Linear algorithm, 30  
Linear recurrence, 167  
Logarithm, 6, 274  
Logarithmic algorithm, 30  
Logic, 4, 272  
Longest border, 234  
Longest common subsequence, 227  
Longest increasing subsequence, 69  
Losing state, 181  
Lowest common ancestor, 140  
Lowest common multiple, 151

**M**

Macro, 14  
Manhattan distance, 218  
Map, 57  
Markov chain, 178  
Matching, 203  
Matrix, 164  
Matrix exponentiation, 167  
Matrix multiplication, 165, 180  
Matrix sum, 165  
Maximum flow, 198  
Maximum independent set, 205  
Maximum matching, 203  
Maximum spanning tree, 100  
Maximum subarray sum, 32  
Meet in the middle, 263  
Memoization, 66

Merge sort, 39  
Mex function, 184  
Minimal rotation, 230  
Minimum cut, 198, 201  
Minimum node cover, 204  
Minimum spanning tree, 100  
Misère game, 183  
Mo's algorithm, 244  
Modular arithmetic, 12  
Modular exponentiation, 153  
Modular multiplicative inverse, 154  
Monte Carlo algorithm, 179  
Multinomial coefficient, 158  
Multiset, 57

**N**

Natural logarithm, 6, 274  
Nearest smaller elements, 113  
Negation, 4, 272  
Negative cycle, 89  
Neighbor, 78  
Nim game, 182  
Nim sum, 183  
Nim theory, 181  
Node, 78  
Node cover, 204  
Not operation, 21  
NP-hard problem, 31  
Number theory, 147

**O**

Or operation, 21  
Order statistic, 179  
Outdegree, 79

**P**

Parallel binary search, 265  
Parent, 131  
Parenthesis expression, 159  
Pascal's triangle, 157  
Path, 78  
Path compression, 105  
Path cover, 205  
Pattern matching, 229, 233  
Perfect matching, 203  
Permutation, 16  
Persistent segment tree, 250  
Pick's theorem, 218  
Point, 211  
Point in a polygon, 216  
Point location, 214

Point-line distance, 215  
Policy-based set, 59  
Polygon area, 216  
Polynomial algorithm, 31  
Polynomial hashing, 228  
Post-order, 133  
Prüfer code, 164  
Pre-order, 133  
Predicate, 4, 272  
Prefix, 225  
Prefix doubling method, 235  
Prefix sum array, 120  
Prim's algorithm, 106  
Primality test, 148  
Prime, 148  
Prime decomposition, 148  
Priority queue, 58  
Probability, 173  
Probability event, 174

## Q

Quadrangle inequality, 260  
Quadratic algorithm, 30  
Quantifier, 4, 272  
Queen problem, 18, 35  
Queue, 54

## R

Random variable, 175  
Randomized algorithm, 179  
Range, 53  
Range query, 119  
Range update, 129  
Reachability, 110  
Recursion, 15  
Regular graph, 79  
Remainder, 12  
Root, 131  
Rooted tree, 131  
Rotating coordinates, 219  
Rotation, 230

## S

Scaling algorithm, 201  
Segment tree, 125, 245  
Set, 3, 55, 271  
Shoelace formula, 216  
Shortest path, 87  
Sieve of Eratosthenes, 150

Signed number, 20  
Sliding window, 114  
Sliding window minimum, 114  
Sorting, 37  
Sorting algorithm, 37  
Spanning tree, 100  
Sparse segment tree, 250  
Sparse table algorithm, 121  
SPFA algorithm, 89  
Sprague–Grundy theorem, 184  
Square matrix, 165  
Square root algorithm, 239  
Stack, 54  
Strassen's algorithm, 166  
String hashing, 228  
Strongly connected component, 189  
Strongly connected graph, 189  
Subalgorithm, 241  
Subsequence, 225  
Subset, 3, 15, 271  
Substring, 225  
Subtree, 131  
Successor, 97  
Successor graph, 97  
Suffix, 225  
Suffix array, 234  
Sweep line algorithm, 44, 220

## T

Ternary search, 115  
Tiling, 74  
Time complexity, 27  
Topological sorting, 94  
Transpose, 165  
Treap, 253  
Tree, 78, 131  
Tree query, 137  
Tree traversal array, 138  
Trie, 226  
Two pointers method, 111  
Two-dimensional segment tree, 253

## U

Uniform distribution, 177  
Union, 3, 271  
Union-find structure, 103  
Universal set, 3, 271  
Unsigned number, 20

**V**

Vector, [52](#), [165](#), [211](#)

**W**

Warnsdorf's rule, [197](#)

Weighted graph, [78](#)

Winning state, [181](#)

**X**

Xor operation, [21](#)

**Z**

Z-algorithm, [231](#)

Z-array, [231](#)