Enumeration of Enumeration Algorithms

Article · May 2016

1 author:

Kunihiro Wasa
National Institute of Informatics

Enumeration of Enumeration Algorithms

Kunihiro Wasa*

May 18, 2016

Abstract

In this paper, we enumerate enumeration problems and algorithms[1]. Other useful catalogues for enumeration algorithms are provided by Komei Fukuda[2] and Yasuko Matsui[3]. This survey is under construction. If you know some results not in this survey or there is anything wrong, please let me know.

Contents

*kunihiro.wasa@gmail.com

[1]See also `http://www-ikn.ist.hokudai.ac.jp/~wasa/enumeration_complexity.html`

[2]`http://www-oldurls.inf.ethz.ch/personal/fukudak/publ/Enumeration/enumalgo95_update.pdf`

[3]`http://www-oldurls.inf.ethz.ch/personal/fukudak/publ/Enumeration/enumalgo93.pdf`

1 Geometry

1.1 Arrangement

1.1.1 Enumeration of all cells in arrangements

Input An arrangement of distinct hyperplanes.

Output All cells in arrangements.

Complexity $O(nm\ell(n,m)N)$ total time and $O(nm)$ space.

Comment n is the dimension, m is the number of hyperplanes, and $\ell(n,m)$ is the time for solving an LP with n variables and $m-1$ inequalities. N is the number of solutions.

Reference [7]

1.2 Face

1.2.1 Enumeration of all arrangements

Input An integer n and $\mathcal{J} \subseteq \{+,-\}^n$.

Output The set $\mathcal{F}$ of faces if $\mathcal{J}$ is the set of maximal faces of an oriented matroid

Complexity $O(\max(n^2|\mathcal{J}|^3, n^3|\mathcal{J}|^2))$ total time.

Comment Their paper treats a reconstruction of the location vectors of all faces from the graph.

Reference [85]

1.3 Matching

1.3.1 Enumeration of all non-crossing perfect matchings (and convex partitions) in a given points

Input A point sets P in a plane

Output All non-crossing perfect matchings (and convex partitions) in P.

Complexity After $O(2^n n^4)$ preprocessing time, polynomial delay.

Comment n is the number of points in P

Reference [259]

1.4 Nearest neighbor

1.4.1 Enumeration of the k smallest distances between pairs of points

Input n points in a plane.

Output The k smallest distances between pairs of points in non-decreasing order.

Complexity $O(n \log n + k \log k)$ total time and $O(n + k)$ space.

Comment I'm looking for this paper.

Reference [57]

1.5 Spanning tree

1.5.1 Enumeration of the Euclidean k best spanning trees in a plane with n points

Input n points in a plane

Output The Euclidean k best spanning trees of the given points.

Complexity $O(k^2 n + n \log n)$ total time

Comment An Euclidean spanning tree in a plane is a spanning tree of the complete graph whose vertices are all given points and the weight of an edge equal to the Euclidean distance between the corresponding vertices.

Reference [64]

1.5.2 Enumeration of the orthogonal k best spanning trees in a plane with n points

Input n points in a plane

Output The orthogonal k best spanning trees of the given points.

Complexity $O(k^2 + kn \log\log(n/k) + n \log n)$ total time.

Comment An orthogonal spanning tree in a plane is a spanning tree of the complete graph whose vertices are all given points and the weight of an edge equal to the L_1 distance between the corresponding vertices.

Reference [64]

1.5.3 Enumeration of all spanning trees of a plane

Input A point set P.

Output All spanning trees of P.

Complexity $O(|P|^3 N)$ total time and $O(|P|)$ space.

Comment N is the number of solutions.

Reference [7]

1.5.4 Enumeration of the Euclidean k best spanning trees in a plane with n points

Input n points in a plane

Output The Euclidean k best spanning trees of the given points.

Complexity $O(n \log n \log k + k \min(k, n)^{1/2})$ total time

Comment An Euclidean spanning tree in a plane is a spanning tree of the complete graph whose vertices are all given points and the weight of an edge equal to the Euclidean distance between the corresponding vertices.

Reference [68]

1.6 Triangulation

1.6.1 Enumeration of all regular triangulations

Input n points.

Output All regular triangulations of given points in $(d-1)$-dimensional space.

Complexity $O(dsLP(r, ds)T)$ time and $O(ds)$ space.

Comment s is the upper bound of the number of simplcies contained in one regular triangulation, $LP(r, ds)$ denotes the time complexity of solving linear programming problem with ds strict inequality constraints in r variables, and T is the number of regular triangulations.

Reference [152]

1.6.2 Enumeration of all spanning regular triangulations

Input n points.

Output All spanning regular triangulations of given points in $(d-1)$-dimensional space.

Complexity $O(dsLP(r, ds)T')$ time and $O(ds)$ space.

Comment s is the upper bound of the number of simplcies contained in one regular triangulation, $LP(r, ds)$ denotes the time complexity of solving linear programming problem with ds strict inequality constraints in r variables, and T' is the number of regular triangulations.

Reference [152]

1.6.3 Enumeration of all triangulations of a point set

Input A point set P.

Output All triangulations of P.

Complexity $O(|P|N)$ total time and $O(|P|)$ space.

Comment N is the number of solutions.

Reference [7]

1.6.4 Enumeration of all triangulations in general dimensions

Input Points.

Output All triangulations.

Complexity See the paper.

Comment This algorithm uses the enumeration algorithm for maximal independent sets.

Reference [233]

1.6.5 Enumeration of all regular triangulations in general dimensions

Input Points.

Output All regular triangulations.

Complexity See the paper.

Comment This algorithm uses the enumeration algorithm for maximal independent sets.

Reference [233]

1.6.6 Enumeration of all triangulation paths of a point set

Input A point set P.

Output All triangulation paths of P.

Complexity $O(N|P|^3 \log |P|)$ time and $O(|P|)$ space.

Comment N is the number of solutions.

Reference [60]

1.6.7 Enumeration of all pseudotriangulations of a finite point set

Input A point set S of size n.

Output All pointed pseudotriangulations of S.

Complexity $O(\log n)$ time per solution with linear space.

Reference [18]

1.6.8 Enumeration of all pseudotriangulations of a point set

Input A point set P.

Output All pseudotriangulations of P.

Complexity $O(n)$ time per pseudotriangulation.

Comment n is the number of points in P.

Reference [36]

1.6.9 Enumeration of all triangulations of a convex polygon of n vertices with numbered

Input A convex polygon P with n vertices that are numbered.

Output All triangulations of P.

Complexity $O(1)$ time per triangulation and $O(n)$ space.

Reference [185]

1.6.10 Enumeration of all triangulations of a convex polygon of n vertices

Input A convex polygon P with n vertices that are not numbered.

Output All triangulations of P.

Complexity $O(n^2)$ time per triangulation and $O(n)$ space.

Reference [185]

2 Graph

2.1 k-degenerate graph

2.1.1 Enumerate well-ordered (strongly) k-generate with n vertices

Input n: the number of vertices.

Output All well-ordered (strongly) k-degenerate graphs with n vertices.

Complexity $O(nm + m^2)$ time per enumerated and printed graph.

Comment m is the number of edges of printed graphs.

Reference [16]

2.1.2 Enumerate well-ordered (strongly) k-generate with n vertices and m edges

Input n: the number of vertices, m: the number of edges.

Output All well-ordered (strongly) k-degenerate graphs with n vertices and m edges.

Complexity $O(n^{3/2}m^2)$ time per enumerated and printed graph.

Reference [16]

2.2 Bounded union

2.2.1 Enumeration of all bounded union

Input A graph with h multiedges each with at most d vertices.

Output All minimal subset U of at most k vertices that entirely includes at least one set from each multiedge.

Complexity $O(dc^{k+1}h + \min(kc^{2k}, hkc^k))$ time.

Reference [52]

2.3 Bridge

2.3.1 Enumeration of all bridge-avoiding extensions of a graph

Input A graph $G = (V, E)$ and an edge subset $B \subseteq E$.

Output All bridge-avoiding extensions of G.

Complexity $O(K^2 \log(K)|E|^2 + K^2(|V| + |E|)|E|^2)$ total time.

Comment K is the number of bridge-avoiding extensions. X is a bridge-avoiding extensions of G if X is a minimal edge set $X \subseteq E \setminus B$ such that no edge $b \in B$ is a bridge in $(V, B \cup X)$.

Reference [127]

2.4 Bubble

2.4.1 Enumeration of all bubbles in a directed graph

Input A directed graph $G = (V, E)$ and a source s.

Output All bubbles with a given source s.

Complexity $O(|V| + |E|)$ delay with $O(|V|(|V| + |E|))$ preprocessing time.

Comment An (s, t)-bubble is two disjoint (s, t)-paths that only share s and t.

Reference [22]

2.4.2 Enumeration of all $(s, t, \alpha_1, \alpha_2)$-bubble in a directed graph

Input A directed graph $G = (V, E)$, source vertex s, and two upper bounds α_1 and α_2.

Output All $(s, t, \alpha_1, \alpha_2)$-bubble in G.

Complexity $O(|V|(|E| + |V| \log |V|))$ delay.

Comment A pair of two vertex-disjoint (s, t)-paths p_1 and p_2 is $(s, t, \alpha_1, \alpha_2)$-bubble in G, if $|p_1| \leq \alpha_1$ and $|p_2| \leq \alpha_2$.

Reference [207]

2.5 Chord diagram

2.5.1 Enumeration of all non-isomorphic chord diagrams

Input An integer $2n$.

Output All non-isomorphic chord diagrams with $2n$ points.

Comment A *chord diagram* is a set of $2n$ points on an oriented circle (counterclockwise) joined pairwise by n chords. In an experiment, their algorithm runs in almost CAT, but there is no Mathematical proof.

Reference [213]

2.6 Chordal graph

2.6.1 Enumeration of all chordal graphs in a graph

Input A graph G.

Output All chordal graphs in G.

Complexity $O(1)$ delay with $O(n^3)$ space.

Comment Using reverse search.

Reference [130]

2.6.2 Enumeration of all chordal supergraph that contains a given graph

Input A graph $G = (V, E)$.

Output All chordal supergraphs each of which contain G.

Complexity $O(|V|^3)$ time for each and $O(|V|^2)$ space.

Reference [129]

2.6.3 Enumeration of all chordal sandwiches in a graph

Input A graph $G = (V, E)$.

Output All chordal sandwiches in G.

Complexity Polynomial delay with polynomial space.

Comment I'm looking for this paper.

Reference [128]

2.7 Clique

2.7.1 Enumeration of all cliques in a graph

Input A graph G.

Output All cliques in G.

Comment They pointed out Augustson and Minker's algorithm has two errors.

Reference [167]

2.7.2 Enumeration of all maximal clique in a graph

Input An undirected graph $G = (V, E)$.

Output All maximal clique.

Reference [3]

2.7.3 Enumeration of all cliques in a graph

Input A graph G.

Output All cliques in G.

Reference [35]

2.7.4 Enumeration of all cliques in an undirected graph

Input An undirected graph G.

Output All cliques in G.

Reference [113]

2.7.5 Enumeration of all maximal cliques in a given graph

Input A graph G.

Output All maximal cliques in G.

Reference [92]

2.7.6 Enumeration of all maximum cliques in a circle graph

Input A circle graph $G = (V, E)$.

Output All maximum cliques in G.

Complexity $O(1)$ time per maximum clique.

Comment A graph is a circle graph if it is the intersection graph of chords in a circle. The definition of a circle graph can be found in ¡a href=”http://www.graphclasses.org/c System on Graph Classes and their Inclusions¡/a¿. I'm looking for this paper.

Reference [195]

2.7.7 Enumeration of all cliques in a connected graph

Input A connected graph $G = (V, E)$ and an order $l \geq 2$ of cliques.

Output All cliques in G.

Complexity $O(l\alpha(G)^{l-2}|E|)$ total time and linear space.

Comment $\alpha(G)$ is the minimum number of edge-disjoint spanning forests into which G can be decomposed. I'm looking for this paper.

Reference [47]

2.7.8 Enumeration of all maximal cliques in a connected graph

Input A connected graph $G = (V, E)$.

Output All maximal cliques in G.

Complexity $O(\alpha(G)|E|)$ total time and linear space.

Comment $\alpha(G)$ is the minimum number of edge-disjoint spanning forests into which G can be decomposed. I'm looking for this paper.

Reference [47]

2.7.9 Enumeration of all maximum cliques of a circular-arc graph

Input A circular-arc graph $G = (V, E)$.

Output All maximum cliques of G.

Complexity $O(|V|^{3.5} + N)$.

Comment N is the number of maximum cliques of G. For a family A of arcs on a circle, a graph $G = (V, E)$ is called the *circular-arc graph* for A if there exists a one-to-one correspondence between V and A such that two vertices in V are adjacent if and only if their corresponding arcs in A intersect.

Reference [120]

2.7.10 Enumeration of all maximal cliques in a graph

Input A graph $G = (V, E)$.

Output All maximal cliques included in G.

Complexity $O(M(n))$ time delay and $O(n^2)$ space, or $O(\delta^4)$ time delay and $O(n + m)$ space, after $O(nm)$ preprocessing.

Comment $M(n)$: the time needed to multiply two $n \times n$ matrices, δ: maximum degree of $G = (V, E)$, n: total number of vertices, and m: total number of edges.

Reference [149]

2.7.11 Enumeration of all maximal cliques in a bipartite graph

Input A bipartite graph $G = (V, E)$.

Output All maximal bicliques included in G.

Complexity $O(M(n))$ time delay and $O(n^2)$ space, or $O(\delta^4)$ time delay and $O(n + m)$ space, after $O(nm)$ preprocessing.

Comment $M(n)$: the time needed to multiply two $n \times n$ matrices, δ: maximum degree of $G = (V, E)$, n: total number of vertices, and m: total number of edges.

Reference [149]

2.7.12 Enumeration of all maximal cliques in a dynamic graph

Input A graph $G_t = (V, E_t)$.

Output All maximal cliques in G_t.

Reference [228]

2.7.13 Enumeration of all maximal cliques in a dynamic graph

Input A dynamic graph G.

Output All maximal cliques in G.

Reference [228]

2.7.14 Enumeration of all bicliques in a graph in lexicographical order

Input A graph $G = (V, E)$.

Output All bicliques in G.

Complexity $O(|V|^3)$ delay and $O(2^{|V|})$ space.

Comment There is no polynomial-delay enumeration algorithm for all bicliques in reverse lexicographical order unless $P = NP$.

Reference [56]

2.7.15 Enumeration of all c-isolated cliques in a graph

Input A graph $G = (V, E)$ and a positive integer c.

Output Enumeration of all c-isolated cliques in G.

Complexity $O(c^5 2^{2c}|E|)$ total time.

Comment A clique S is said to be c-isolated if $|E(S, G \setminus S)| < ck$, where k is the number of vertices in S and $E(G_A, G_B)$ denotes the set of edges connecting the two subgraphs G_A and G_B directly, i.e., the set of edges (v_1, v_2) such that v_1 is in G_A and v_2 in G_B.

Reference [108]

2.7.16 Enumeration of all maximal cliques in a graph

Input A graph $G = (V, E)$.

Output All maximal Cliques in G.

Complexity $O(\frac{\Delta M_c Tri^2}{P})$ delay and $O(|V| + |E|)$ space.

Comment Δ is the maximum degree in G, M_c is the size of the maximum clique in G, and Tri is the number of triangles in G. This algorithm runs on the computer with P processors.

Reference [59]

2.7.17 Enumeration of all cliques in a chordal graph

Input A chordal graph G.

Output All cliques in G.

Complexity $O(1)$ time per solution on average.

Reference [130]

2.7.18 Enumeration of all maximal cliques in a graph

Input A graph $G = (V, E)$.

Output All maximal cliques in G.

Complexity $O(3^{n/3})$ total time.

Reference [237]

2.7.19 Enumeration of K-largest maximal cliques in a graph

Input A graph G.

Output K-largest maximal cliques in G.

Reference [37]

2.7.20 Enumeration of all maximal cliques in a graph

Input A graph G.

Output All maximal cliques in G.

Reference [183]

2.7.21 Enumeration of all maximal bicliques in a bipartite graph

Input A bipartite graph $G = (U, V, E)$.

Output All maximal bicliques in G in lexicographical order on U.

Complexity $O((|U| + |V|)^2)$ delay with exponential space.

Reference [90]

2.7.22 Enumeration of all maximal cliques in a comparability graph

Input A comparability graph $G = (V, E)$.

Output All maximal cliques in G in lexicographical order.

Complexity $O(|V|)$ delay and $O(|V| + |E|)$ space.

Reference [90]

2.7.23 Enumeration of all maximal cliques in a graph

Input A graph $G = (V, E)$.

Output All maximal cliques in G in lexicographical order.

Complexity $O(|V||E|)$ delay with exponential space.

Reference [90]

2.7.24 Enumeration of all maximal cliques in a graph

Input A graph G.

Output All maximal cliques in G.

Comment This algorithm can be paralleled.

Reference [218]

2.7.25 Enumeration of all c-isolated maximal clique in a graph

Input A graph $G = (V, E)$ and a positive real number c.

Output All c-isolated maximal clique in G.

Complexity $O(c^4 2^{2c}|E|)$ total time.

Reference [107]

2.7.26 Enumeration of all c-isolated pseudo-clique in a graph

Input A graph $G = (V, E)$ and a positive real number c.

Output All c-isolated $PC(k - \log k, \frac{k}{\log k})$ in G.

Complexity $O(c^4 2^{2c}|E|)$ total time.

Comment $PC(\alpha, \beta)$ is a induced subgraph of G with an average degree at least α and the minimum degree at least β.

Reference [107]

2.7.27 Enumeration of all avg-c-isolated maximal cliques in a graph

Input A graph $G = (V, E)$ and an integer c.

Output All avg-c-isolated maximal cliques in G.

Complexity $O(2^c c^5 |E|)$ total time.

Comment A vertex set $S \subseteq V$ with k vertices is ave-c-isolated if it has less than ck outgoing edges, where an outgoing edge is an edge between a vertex in S and a vertex in $V \setminus S$.

Reference [105]

2.7.28 Enumeration of all max-c-isolated maximal cliques in a graph

Input A graph $G = (V, E)$ and an integer c.

Output All max-c-isolated maximal cliques in G.

Complexity $O(2^c c^5 |E|)$ total time.

Comment A vertex set $S \subseteq V$ with k vertices is max-c-isolated if every vertex in S has less than c neighbors in $V \setminus S$.

Reference [105]

2.7.29 Enumeration of all cliques in a graph with degeneracy d

Input A graph $G = (V, E)$ with degeneracy d.

Output All cliques in G.

Complexity $O(d|V|3^{d/3})$ time.

Comment They also show the largest possible number of maximal cliques in G. The number is $(|V| - d)3^{d/3}$.

Reference [67]

2.7.30 Enumeration of all pseudo-cliques in a graph

Input A graph $G = (V, E)$ and a threshold θ.

Output All pseudo-cliques in G, whose densities are no less than θ.

Complexity $O(\Delta|V| + \min\{\Delta^2, |V| + |E|\})$ delay with $O(|V| + |E|)$ space.

Comment Δ is the maximum degree of G. The density of a subgraph $G[S]$ induced by $S \subseteq V$ is given by the number of edges over the number of all its vertex pairs.

Reference [245]

2.7.31 Enumeration of all maximal cliques in a graph with limited memory

Input A graph G.

Output All maximal graphs in G.

Complexity See the paper.

Reference [46]

2.7.32 Enumeration of all maximal cliques in a graph

Input An undirected graph $G = (V, E)$.

Output All maximal clique.

Complexity $O(\delta \cdot H^3)$ time delay and $O(n + m)$ space.

Comment H satisfies $|\{v \in V | \sigma(v) \geq H\}| \leq H$.

Reference [42]

2.7.33 Enumeration of all maximal cliques in a graph

Input A graph G.

Output All maximal cliques in G.

Reference [102]

2.8 Coloring

2.8.1 Enumeration of all the edge colorings in a bipartite graph

Input A bipartite graph $G = (V, E)$.

Output All edge colorings in G.

Complexity $O(\Delta|E|)$ time per solution and space.

Comment Δ is the maximum degree in G. I'm looking for this paper.

Reference [271]

2.8.2 Enumeration of all the edge colorings of a bipartite graph

Input A bipartite graph $B = (V, E)$.

Output All the edge colorings pf B.

Complexity $O(T(|V| + |E| + \Delta) + K \min\{|V|^2 + |E|, T(|V|, |E|, \Delta)\})$ total time and $O(|E|\Delta)$ space.

Comment Δ is the number of maximum degree and $T(|V|, |E|, \Delta)$ is the time complexity of an edge coloring algorithm.

Reference [156]

2.8.3 Enumeration of all the edge colorings of a bipartite graph

Input A bipartite graph $B = (V, E)$.

Output All the edge colorings pf B.

Complexity $O(|V|)$ time per solution and $O(|E|)$ space.

Comment I'm looking for this paper.

Reference [159]

2.9 Connected

2.9.1 Enumeration of all minimal strongly connected subgraphs in a strongly connected subgraph

Input A strongly connected subgraph G.

Output All minimal strongly connected subgraph.

Complexity Incremental polynomial time.

Reference [27]

2.9.2 Enumeration of all minimal 2-vertex connected spanning subgraphs in a graph

Input A graph G.

Output All minimal 2-vertex connected spanning subgraphs of G.

Complexity Incremental polynomial time.

Reference [125]

2.10 Cut

2.10.1 Enumeration of all cuts between all pair of vertices in a given graph

Input A graph G.

Output All cuts bewteen all pair of vertices in G

Reference [150]

2.10.2 Enumeration of all cutsets in a graph

Input A graph $G = (V, E)$.

Output All cutsets in G.

Complexity $O((|V| + |E|)(\mu + 1))$ total time and $O(|V| + |E|)$ or $O(|V|^2)$ space.

Comment μ is the number of solutions.

Reference [239]

2.10.3 Enumeration of k best cuts in a directed graph

Input A directed graph $G = (V, E)$.

Output k best cuts in G.

Complexity $O(k|V|^4)$ total time.

Reference [100]

2.10.4 Enumeration of K-best cuts in a network

Input A graph $G = (V, E)$.

Output K-best cuts in G.

Complexity $O(K \cdot |V|^4)$ time.

Reference [100]

2.10.5 Enumeration of all articulation pairs in a planar graph

Input An undirected graph $G = (V, E)$.

Output All articulation pairs in G.

Complexity $O(|V|^2)$ total time.

Reference [141]

2.10.6 Enumeration of all minimum-size separating vertex sets in a graph

Input A graph $G = (V, E)$.

Output All minimum-size separating vertex sets.

Complexity $\Theta(M|V| + C) = O(2^k n^3)$ total time.

Comment M is the number of solutions, k is the connectivity of G, and $C = k|V| \min(k(|V| + |E|), A)$, where A is the time complexity of the best maximum network flow algorithm for unit network. I'm looking for this paper.

Reference [115]

2.10.7 Enumeration of all minimal separators in a graph

Input A graph $G = (V, E)$.

Output All minimal separators in G.

Complexity $O(|V|^6 R)$ total time.

Comment R is the number of solutions.

Reference [131]

2.10.8 Enumeration of all (s, t)-cuts in a graph

Input A graph $G = (V, E)$ and two vertices s, t in G.

Output All (s, t)-cuts in G.

Complexity $O(|E|)$ time per cut.

Reference [189]

2.10.9 Enumeration of all (s,t)-cuts in a directed graph

Input A directed graph $G = (V, E)$ and two vertices s, t in G.

Output All (s,t)-cuts in G.

Complexity $O(|E|)$ time per cut.

Reference [189]

2.10.10 Enumeration of all (s,t)-uniformly directed cuts in a directed graph

Input A directed graph $G = (V, E)$ and two vertices s, t in G.

Output All (s,t)-uniformly directed cuts in G.

Complexity $O(|V|)$ time per cut.

Comment An *undirected directed cut* is also called a UDC. An (s,t)-DUC is an (s,t)-cut (X, Y) such that $(X, Y) = \emptyset$, where $(X, Y) = \{(u, v) \in E : u \in X, v \in Y\}$.

Reference [189]

2.10.11 Enumeration of all minimum weighted (s,t) cuts in an weighted graph

Input An weighted graph $G = (V, E)$ and two vertices s, t in G.

Output All minimum weighted (s,t) cuts in G.

Complexity $O(|V|)$ time per cut.

Reference [189]

2.10.12 Enumeration of all semidirected (s,t) cuts in a directed graph

Input A directed graph $G = (V, E)$ and two vertices s, t in G.

Output All semidirected (s,t) cuts in G.

Complexity $O(|V||E|)$ time per cut.

Comment For a subset D of directed edges, a *semidirected* (s,t) *cut* with respect to D is an (s,t) cut (X,Y) such that $(X,Y) \cup (Y,X)$ defnes an undirected (s,t) cutset and such that $(X,Y) \cap D = \emptyset$, where a *cutset* is an minimal cut set.

Reference [189]

2.10.13 Enumeration of all strong (s,K) cutsets in a graph

Input A graph $G = (V,E)$, $s \in V$ and $K \subseteq V \setminus \{s\}$.

Output All strong (s,K) cutsets in G.

Complexity $O(|E|)$ time per cut.

Comment An (s,K)-*cut* is defined to be any cut (X,Y) for which $s \in X$ and $K \cap Y \neq \emptyset$. A *strong* (s,K) *cutset* is minimal cuts of the form (X,Y) where $s \in X$ and $K \subseteq Y$.

Reference [189]

2.10.14 Enumeration of all strong (s,K) cutsets in a directed graph

Input A directed graph $G = (V,E)$, $s \in V$ and $K \subseteq V \setminus \{s\}$.

Output All strong (s,K) cutsets in G.

Complexity $O(|E|)$ time per cut.

Comment An (s,K)-*cut* is defined to be any cut (X,Y) for which $s \in X$ and $K \cap Y \neq \emptyset$. A *strong* (s,K) *cutset* is minimal cuts of the form (X,Y) where $s \in X$ and $K \subseteq Y$.

Reference [189]

2.10.15 Enumeration of all quasi (s,K) cuts in a graph

Input A graph $G = (V,E)$, $s \in V$ and $K \subseteq V \setminus \{s\}$.

Output All quasi (s,K) cutsets in G.

Complexity $O(|E|)$ time per cut.

Comment An (s, K)-*cut* is defined to be any cut (X, Y) for which $s \in X$ and $K \cap Y \neq \emptyset$. A *Quasi* (s, K) *cut* is an edge set that is strong (s, A)-cutsets for some $A \subseteq K$ and $A \neq \emptyset$.

Reference [189]

2.10.16 Enumeration of all quasi (s, K) cuts in a directed graph

Input A directed graph $G = (V, E)$, $s \in V$ and $K \subseteq V \setminus \{s\}$.

Output All quasi (s, K) cutsets in G.

Complexity $O(|E|)$ time per cut.

Comment An (s, K)-*cut* is defined to be any cut (X, Y) for which $s \in X$ and $K \cap Y \neq \emptyset$. A *Quasi* (s, K) *cut* is an edge set that is strong (s, A)-cutsets for some $A \subseteq K$ and $A \neq \emptyset$.

Reference [189]

2.10.17 Enumeration of all (s, K) cutsets in a graph

Input A graph $G = (V, E)$, $s \in V$ and $K \subseteq V \setminus \{s\}$.

Output All (s, K) cutsets in G.

Complexity $O(|E|)$ time per cut.

Comment An (s, K)-*cut* is defined to be any cut (X, Y) for which $s \in X$ and $K \cap Y \neq \emptyset$. A (s, K) *cutset* is minimal (s, K) cuts.

Reference [189]

2.10.18 Enumeration of all minimal a-b separators in a graph

Input An undirected connected simple graph $G = (V, E)$, non-adjacent vertices a and b in G.

Output All minimal a-b separator of G.

Complexity $O(R_{ab}|V|^3)$ total time.

Comment R_{ab} is the number of minimal a-b separators.

Reference [223]

2.10.19 Enumeration of all minimal a-b separators in a graph

Input An undirected connected simple graph $G = (V, E)$, non-adjacent vertices a and b in G.

Output All minimal a-b separator of G.

Complexity $O(R_{ab}|V|/\log|V|)$ total time.

Comment R_{ab} is the number of minimal a-b separators. This algorithm needs $O(|V|^3)$ processors on a CREW PRAM.

Reference [223]

2.10.20 Enumeration of all minimal separators of a graph

Input A graph $G = (V, E)$.

Output All minimal separators of G.

Complexity $O(|V|^5 R)$ total time.

Comment R is the number of solutions.

Reference [132]

2.10.21 Enumeration of all minimal separator of a graph

Input A graph $G = (V, E)$.

Output All minimal separator of G.

Complexity $O(|V|^3)$ time per solution.

Reference [19]

2.10.22 Enumeration of all minimal separator of a chordal graph

Input A chordal graph $G = (V, E)$.

Output All minimal separator of G.

Complexity $O(|V| + |E|)$ total time.

Reference [40]

2.10.23 Enumeration of all cut conjunctions for a given set of vertex pairs in a graph

Input A graph $G = (V, E)$, and a collection $B = \{(s_1, t_1), ..., (s_k, t_k)\}$ of k vertex pairs $s_i, t_i \in V$.

Output All cut conjunctions for B in G.

Complexity Incremental polynomial time.

Comment A minimal edge sets $X \subseteq E$ is a *cut conjunction* if, for all $i = 1, \ldots, k$, vertices s_i and t_i is disconnected in $G' = (V, E \setminus X)$. A cut conjunction is a generalization of an $s - t$ cut.

Reference [126]

2.10.24 Enumeration of all minimal separators of a 3-connected planar graph

Input A 3-connected planar graph $G = (V, E)$.

Output All minimal separators of G.

Complexity $O(|V|)$ time per solution.

Reference [162]

2.10.25 Enumeration of all cut conjunctions of a graph

Input A graph $G = (V, E)$ and a collection $B = \{(s_1, t_1), \ldots, (s_k, t_k)\}$.

Output All cut conjunctions of G.

Complexity $O(K^2 \log(K)|V||E|^2 + K^2|B|(|V| + |E|)|E|^2)$ total time.

Comment K is the number of cut conjunctions. X is a cut conjunctions of G if X is a minimal edge set such that for all $i = 1, \ldots, k$, a pair of vertices s_i and t_i is disconnected in $(V, E \setminus X)$.

Reference [127]

2.10.26 Enumeration of all (s,t)-cuts in an weighted directed graph

Input An weighted directed graph $G = (V, E)$.

Output All cuts in G by non-decreasing weights ordering.

Complexity $O(|V||E|\log(|V|^2/|E|))$ delay.

Reference [273]

2.10.27 Enumeration of all (s,t)-cuts in an weighted undirected graph

Input An weighted undirected graph $G = (V, E)$.

Output All cuts in G by non-decreasing weights ordering.

Complexity $O(|V||E|\log(|V|^2/|E|))$ delay.

Reference [273]

2.11 Cycle

2.11.1 Enumeration of all cycles in an n-cube, where $n \leq 4$

Input An integer n.

Output All cycles (closed paths) in an n-cube.

Reference [93]

2.11.2 Enumeration of all cycles in a graph

Input A graph G.

Output All cycles in G.

Reference [256]

2.11.3 Enumeration of all simple cycles in a graph

Input A graph G.

Output All simple cycles in G.

Complexity

Reference [186]

2.11.4 Enumeration of all circuits in a graph

Input A graph G.

Output All circuits in G.

Reference [257]

2.11.5 Enumeration of all Hamiltonian circuits in a graph

Input A graph G.

Output All Hamiltonian circuits in G.

Reference [272]

2.11.6 Enumeration of all directed circuits in a directed graph

Input A directed graph G.

Output All directed circuits in G.

Reference [114]

2.11.7 Enumeration of all cycles in a graph

Input A graph G.

Output All cycles in G.

Reference [53]

2.11.8 Enumeration of all cycles in a graph

Input A graph G.

Output All cycles in G.

Reference [10]

2.11.9 Enumeration of all elementary circuit in a graph

Input A graph G.

Output All elementary circuit in G.

Complexity

Reference [236]

2.11.10 Enumeration of all cycles in a graph

Input A graph G.

Output All cycles in G.

Reference [44]

2.11.11 Enumeration of all cycles in an undirected graph

Input An undirected graph G.

Output All cycles in G.

Reference [255]

2.11.12 Enumeration of all cycles in a finite undirected graph

Input A finite undirected graph G.

Output All cycles in G.

Comment He claimed that J. T. Welch, Jr.'s algorithm is wrong.

Reference [255]

2.11.13 Enumeration of all cycles in a directed graph

Input A directed graph $G = (V, E)$.

Output All cycles in G.

Complexity $O((|V| \cdot |E|)(C + 1))$ total time and $O(|V| + |E|)$ space.

Comment C is the number of cycles included in G.

Reference [235]

2.11.14 Enumeration of all cycles in a directed graph

Input A graph $G = (V, E)$.

Output All cycles in G.

Complexity $O(|E| + c(|V| \times |E|))$ total, where c is the number of circuits in G.

Reference [61]

2.11.15 Enumeration of all cycles in a directed graph

Input A directed graph $G = (V, E)$.

Output All cycles in G.

Complexity $O((|V| + |E|)(C + 1))$ total time and $O(|V| + |E|)$ space.

Comment C is the number of cycles included in G.

Reference [111]

2.11.16 Enumeration of all cycles in a graph

Input A graph $G = (V, E)$.

Output All cycles in G.

Complexity $O(|E|)$ time per cycle with $O(|E|)$ space.

Reference [193]

2.11.17 Enumeration of all cycles in a directed graph

Input A directed graph $G = (V, E)$.

Output All cycles in G.

Complexity $O(|E|)$ time per cycle with $O(|E|)$ space.

Reference [193]

2.11.18 Enumeration of all cycles in a planar graph

Input A planar graph $G = (V, E)$.

Output All cycles in G.

Complexity $O(|V|(C + 1))$ total time and $O(|V|)$ space.

Comment C is the number of cycles included in G.

Reference [230]

2.11.19 Enumeration of all cycles of a given length in a graph

Input A graph G and an integer k.

Output All cycles of length k in G.

Complexity See paper.

Reference [4]

2.11.20 Enumeration of all cycles in a graph

Input A graph G.

Output All cycles in G.

Reference [261]

2.11.21 Enumeration of all small cycles in a graph

Input A graph G.

Output All cycles with length at most 5 in G.

Reference [261]

2.11.22 Enumeration of all chordless cycles in a graph

Input A graph G.

Output All chordless cycles in G.

Reference [261]

2.11.23 Enumeration of all Hamiltonian cycles in a graph

Input A graph G.

Output All Hamiltonian cycles in G.

Reference [261]

2.11.24 Enumeration of all cycles in a graph

Input A graph $G = (V, E)$.

Output All cycles in G.

Complexity $O(|E| + \sum_{c \in \mathcal{C}(G)} |c|)$ total time.

Comment $\mathcal{C}(G)$ is the set of all cycles in G.

Reference [23]

2.11.25 Enumeration of all chordless cycles in a graph

Input A graph $G = (V, E)$.

Output All chordless cycles in G.

Complexity $\tilde{O}(|E| + |V| \cdot C)$ total time.

Comment C is the number of chordless cycles in G. I'm looking for this paper.

Reference [76]

2.11.26 Enumeration of all chordless cycles in a graph

Input A graph $G = (V, E)$.

Output All chordless cycles in G.

Complexity $O(|V| + |E|)$ time per chordless cycle.

Reference [247]

2.12 Dominating set

2.12.1 Enumeration of all minimal dominating sets in a graph

Input A graph $G = (V, E)$.

Output All minimal dominating sets in G.

Complexity $O(1.7159^{|V|})$ total time.

Reference [78]

2.12.2 Enumeration of z smallest weighted edge dominating sets in a graph

Input An weighted graph $G = (V, E)$, and positive integers k and z. Each edge of G has a positive weight.

Output z smallest weighted edge dominating sets in G.

Complexity $O(5.6^{2k}k^4z^2 + 4^{2k}k^3z|V|)$ total time.

Reference [250]

2.12.3 Enumeration of all minimal dominating sets in a line graph

Input A line graph G.

Output All minimal dominating sets in G.

Complexity $O(||G||^5 N^6)$ total time.

Comment $||G||$ is the size of G and N is the number of minimal dominating sets in G.

Reference [118]

2.12.4 Enumeration of all minimal dominating sets in a path graph or $(C_4, C_5, craw)$-free graph

Input A line graph or $(C_4, C_5, craw)$-free graph G.

Output All minimal dominating sets in G.

Complexity $O(||G||^2 N^3)$ total time.

Comment $||G||$ is the size of G and N is the number of minimal dominating sets in G.

Reference [118]

2.12.5 Enumeration of all minimal edge-dominating sets in a graph

Input A graph G.

Output All minimal edge-dominating sets in G.

Complexity $O(||L(G)||^5 N^6)$ total time.

Comment $L(G)$ is the line graph of G, $||L(G)||$ is the size of $L(G)$, and N is the number of minimal edge-dominating sets in G.

Reference [118]

2.12.6 Enumeration of all minimal dominating sets in an undirected permutation graph

Input An undirected permutation graph $G = (V, E)$.

Output All minimal dominating sets.

Complexity $O(|V|)$ delay with $O(|V|^8)$ pre-processing.

Reference [117]

2.12.7 Enumeration of all minimal dominating sets in an undirected interval graph

Input An undirected interval graph $G = (V, E)$.

Output All minimal dominating sets.

Complexity $O(|V|)$ delay with $O(|V|^3)$ pre-processing.

Reference [117]

2.12.8 Enumeration of all minimal edge dominating sets in a graph

Input A graph $G = (V, E)$.

Output All minimal edge dominating sets in G.

Complexity $O(|V|^6|\mathcal{L}|)$ delay.

Comment $\mathcal{L}$ is the set of already generated solutions.

Reference [95]

2.12.9 Enumeration of all minimal dominating sets in a line graph

Input A line graph $G = (V, E)$.

Output All minimal dominating sets in G.

Complexity $O(|V|^2|E|^2|\mathcal{L}|)$ delay.

Comment $\mathcal{L}$ is the set of already generated solutions.

Reference [95]

2.12.10 Enumeration of all minimal edge dominating sets in a bipartite graph

Input A bipartite graph $G = (V, E)$.

Output All minimal edge dominating sets in G.

Complexity $O(|V|^4|\mathcal{L}|)$ delay.

Comment $\mathcal{L}$ is the set of already generated solutions.

Reference [95]

2.12.11 Enumeration of all minimal dominating sets in the line graph of a bipartite graph

Input A graph $G = (V, E)$.

Output All minimal dominating sets in G.

Complexity $O(|V|^2|E||\mathcal{L}|)$ delay.

Comment $\mathcal{L}$ is the set of already generated solutions.

Reference [95]

2.12.12 Enumeration of all minimal dominating sets in a graph of girth at least 7

Input A graph $G = (V, E)$ of girth at least 7.

Output All minimal dominating sets in G.

Complexity $O(|V|^2|E||\mathcal{L}|^2)$ delay.

Comment $\mathcal{L}$ is the set of already generated solutions.

Reference [95]

2.12.13 Enumeration of all 2-dominating sets in a tree

Input A tree $T = (V, E)$.

Output All 2-dominating sets of T.

Complexity $O(1.3248^n)$ total time.

Comment If a subset $U \subseteq V$ is a 2-dominating set if every vertex $v \in V \setminus U$ has at least two neighbors in U.

Reference [140]

2.12.14 Enumeration of all minimal dominating sets in a P_6-free chordal graph

Input A P_6-free chordal graph $G = (V, E)$.

Output All minimal dominating sets in G.

Complexity Linear delay with $O(|V|^2)$ space.

Reference [116]

2.12.15 Enumeration of all minimal dominating sets in a chordal bipartite graph

Input A chordal bipartite graph G.

Output All minimal dominating sets in G.

Complexity $O(n^3m|\mathcal{L}|^2)$ delay and the total running time is $O(n^3m|\mathcal{L}^*|^2)$.

Comment n is the number vertices in G, m is the number of edges in G, $\mathcal{L}$ is the family of already generated minimal dominating sets, and $\mathcal{L}^*$ is the family of all minimal dominating sets.

Reference [96]

2.13 Drawing

2.13.1 Enumeration of all rectangle drawings with n faces

Input An integer n.

Output All rectangle drawings with n faces.

Complexity $O(n)$ time per drawing and space.

Comment I'm looking for this paper.

Reference [232]

2.14 Feedback arc set

2.14.1 Enumeration of all minimal feedback arc sets in a graph

Input A graph G.

Output All minimal feedback arc sets in G.

Reference [272]

2.14.2 Enumeration of all minimal feedback arc sets in a directed graph

Input A directed graph $G = (V, E)$.

Output All minimal feedback arc sets in G.

Complexity $O(|V||E|(|V|+|E|))$ time delay.

Reference [219]

2.15 Feedback vertex set

2.15.1 Enumeration of all minimal feedback vertex sets in a graph

Input A graph G.

Output All minimal feedback vertex sets in G.

Reference [272]

2.15.2 Enumeration of all feedback vertex sets in a strongly connected directed graph

Input A strongly connected directed graph $G = (V, E)$ and an integer k.

Output All feedback vertex sets of size k in G.

Complexity $O(|V|^{k-1}|E|)$ total time and $O(|E|)$ space.

Reference [89]

2.15.3 Enumeration of all minimal feedback vertex sets in a graph

Input A graph $G = (V, E)$.

Output All minimal feedback vertex sets in G.

Complexity $O(|V||E|(|V|+|E|))$ time delay.

Reference [219]

2.15.4 Enumeration of all minimal feedback vertex sets in a directed graph

Input A directed graph $G = (V, E)$.

Output All minimal feedback vertex sets in G.

Complexity $O(|V|^2(|V|+|E|))$ time delay.

Reference [219]

2.16 General

2.16.1 Enumeration of all graphs in an almost sure first order fimiles

Input A first order language θ.

Output All graphs in $\mathcal{G}_\theta$.

Complexity Polynomial space and delay.

Reference [94]

2.17 Independent set

2.17.1 Enumeration of all maximal independent sets in a chordal graph

Input A chordal graph $G = (V, E)$.

Output All maximal independent sets in G.

Complexity $O(|V||E|\mu)$ total time.

Comment μ is the number of maximal independent sets of G. This algorithm can also enumerate all the maximal cliques.

Reference [240]

2.17.2 Enumeration of all maximal independent sets in a claw-free graph

Input A claw-free graph G.

Output All maximal independent sets in G.

Reference [166]

2.17.3 Enumeration of all maximal independent sets in an undirected graph

Input A graph G.

Output All maximal independent sets in G in lexicographically.

Reference [146]

2.17.4 Enumeration of all maximal independent sets in an interval graph

Input An interval graph $G = (V, E)$.

Output All maximal independent sets in G.

Complexity $O(|V|^2 + \beta)$ total time.

Comment β is the sum of the vertices in all maximal independent sets of G.

Reference [143]

2.17.5 Enumeration of all maximal independent sets in a circular-arc graph

Input A circular-arc graph $G = (V, E)$.

Output All maximal independent sets in G.

Complexity $O(|V|^2 + \beta)$ total time.

Comment β is the sum of the vertices in all maximal independent sets of G.

Reference [143]

2.17.6 Enumeration of all maximal independent sets in a chordal graph

Input A chordal graph $G = (V, E)$.

Output All maximal independent sets in G.

Complexity $O((|V| + |E|)N)$ total time.

Comment N is the number of solutions.

Reference [143]

2.17.7 Enumeration of all maximal independent sets in a graph

Input A graph $G = (V, E)$.

Output All maximal independent sets included in G.

Complexity $O(n(m + n \log C)) = O(n^3)$ delay and exponential space.

Comment C: total number of maximal independent sets, n: total number of vertices, and m: total number of edges. Is there no polynomial space and delay algorithm?

Reference [110]

2.17.8 Enumeration of all maximum independent sets of a bipartite graph

Input A bipartite graph $B = (V, E)$.

Output All maximum independent sets of B.

Complexity $O(|V|^{2.5} + N)$.

Comment N is the number of maximum independent sets of B.

Reference [120]

2.17.9 Enumeration of all maximal independent sets on a tree in lexicographic order

Input Tree $T = (V, E)$.

Output All maximal independent sets on T in lexicographic order.

Complexity $O(|V|^2)$ delay with $O(|V|)$ space.

Reference [43]

2.17.10 Enumeration of all maximum independent set of a graph

Input A graph $G = (V, E)$.

Output All maximum independent set of G.

Complexity $O(2^{0.114|E|})$ total time and polynomial space.

Reference [17]

2.17.11 Enumeration of all maximum independent set of a graph

Input A graph $G = (V, E)$ with maximum degree 3.

Output All maximum independent set of G.

Complexity $O(2^{0.171|V|})$ total time and polynomial space.

Reference [17]

2.17.12 Enumeration of all maximum independent set of a graph

Input A graph $G = (V, E)$ with maximum degree 4.

Output All maximum independent set of G.

Complexity $O(2^{0.228|V|})$ total time and polynomial space.

Reference [17]

2.17.13 Enumeration of all maximum independent set of a graph

Input A graph $G = (V, E)$.

Output All maximum independent set of G.

Complexity $O(2^{0.290|V|})$ total time and polynomial space.

Reference [17]

2.17.14 Enumeration of all maximal independent sets of a graph

Input A graph $G = (V, E)$ and a position integer k.

Output Enumeration of all maximal independent sets with at most size k of G.

Complexity $O(3^{4k-|V|}4^{|V|-3k})$ total time.

Reference [66]

2.17.15 Enumeration of all independent sets in a chordal graph

Input A chordal graph $G = (V, E)$.

Output All independent sets in G.

Complexity Constant time per solution on average after $O(|V|+|E|)$ time for preprocessing.

Reference [179]

2.17.16 Enumeration of all independent sets of size k in a chordal graph

Input A chordal graph $G = (V, E)$ and a positive integer k.

Output All independent sets of size k in G.

Complexity Constant time per solution on average after $O((|V|+|E|)|V|^2)$ time for preprocessing.

Reference [179]

2.17.17 Enumeration of all maximum independent sets in a chordal graph

Input A chordal graph $G = (V, E)$.

Output All maximum independent sets in G.

Complexity Constant time per solution on average after $O((|V|+|E|)|V|^2)$ time for preprocessing.

Reference [179]

2.17.18 Enumeration of all independent sets in an input chordal graph

Input A chordal graph $G = (V, E)$.

Output All independent sets in G.

Complexity $O(1)$ delay and $O(|V|(|V| + |E|))$ time and space for preprocessing.

Comment Counting all independent sets in an input chordal graph needs $O(n+m)$ time.

Reference [180]

2.17.19 Enumeration of all maximum independent sets in an input chordal graph

Input A chordal graph $G = (V, E)$.

Output All maximum independent sets in G.

Complexity $O(1)$ delay and $O(|V|(|V| + |E|))$ time and space for preprocessing.

Comment Counting all maximum independent sets in an input chordal graph needs $O(n+m)$ time.

Reference [180]

2.17.20 Enumeration of all independent sets with k vertices in an input chordal graph

Input A chordal graph $G = (V, E)$ and an integer k.

Output All maximum independent sets with k vertices in G.

Complexity $O(1)$ delay and $O(k|V|(|V|+|E|))$ time and space for preprocessing.

Comment The number of independent sets with k vertices in an input chordal graph needs $O(k^2(|V| + |E|))$ time.

Reference [180]

2.18 Interval graph

2.18.1 Enumeration of all interval supergraph that contains a given graph

Input A graph $G = (V, E)$.

Output All interval supergraphs each of which contain G.

Complexity $O(|V|^3)$ time for each and $O(|V|^2)$ space.

Reference [129]

2.18.2 Enumeration of all interval graph of a given graph

Input A graph $G = (V, E)$.

Output All interval graphs of G.

Complexity $O((|V| + |E|)^2)$ time for each.

Reference [129]

2.18.3 Enumeration of Proper Interval Graphs

Input An integer n.

Output All proper interval graphs with n vertices.

Complexity $O(1)$ time per proper interval graph and $O(n)$ space, after $O(n^2)$ preprocessing time.

Comment Preprocessing: generating the complete graph with n vertices.

Reference [209]

2.19 Matching

2.19.1 Enumeration of the k best perfect matchings of a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k best perfect matchings of G in order.

Complexity $O(k|V|^3)$ total time.

Reference [45]

2.19.2 Enumeration of all stable marriage

Input A graph $G = (V, E)$.

Output All stable marriage of G.

Complexity $O(|V|)$ time per solution and $O(|V|^2)$ space.

Comment I'm looking for this paper.

Reference [97]

2.19.3 Enumeration of all minimum cost perfect matchings in an weighted bipartite graph

Input An weighted bipartite graph $B = (V, E)$.

Output All minimum cost perfect matchings in B.

Complexity $O(|V|(|V| + |E|))$ time per solution and $O(|V| + |E|)$ space

Comment I'm looking for this paper.

Reference [83]

2.19.4 Enumeration of all perfect matchings in a bipartite graph

Input A bipartite graph $B = (U, V, E)$, where $|U| = |V|$.

Output All perfect matchings in B.

Complexity $O(c(|V|+|E|))$ total time and $O(|V|+|E|)$ space, after $O(n^{2.5})$ preprocessing time.

Comment c is the number of solutions.

Reference [82]

2.19.5 Enumeration of the k best perfect matchings of a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k best perfect matchings of G in decreasing order.

Complexity $O(k|V|^3)$ total time with $O(k|V|^2)$ space.

Reference [155]

2.19.6 Enumeration of all perfect, maximum, and maximal matchings in bipartite graphs

Input A bipartite graph $B = (V, E)$.

Output All perfect, maximum, and maximal matching in B.

Complexity $O(|V|)$ time per matching.

Reference [243]

2.19.7 Enumeration of all maximal matchings in a graph

Input A graph $G = (V, E)$.

Output All maximal matchings in G.

Complexity $O(|V| + |E| + \Delta N)$ total time and $O(|V| + |E|)$ space.

Comment N is the number of solutions and Δ is the maximum degree in G. I'm looking for this paper.

Reference [241]

2.19.8 Enumeration of all minimal blocker in a bipartite graph

Input A bipartite graph $G = (U, V, E)$.

Output All minimal blocker in G.

Complexity Polynomial delay and space.

Comment A *blocker* of G is an edge subset X of E such that $G' = (U, V, E \setminus X)$ has no perfect matching. I'm looking for this paper.

Reference [30]

2.19.9 Enumeration of all basic perfect 2-matchings in a graph

Input A graph $G = (V, E)$.

Output All basic perfect 2-matchings in G.

Complexity Incremental polynomial delay.

Comment A *basic 2-matching* of G is a subset of edges that cover the vertices with vertex-disjoint edges and vertex-disjoint odd cycles. I'm looking for this paper.

Reference [30]

2.19.10 Enumeration of all d-factor in a bipartite graph

Input A bipartite graph $G = (U, V, E)$ and any non negative function $d : A \cup B \to \{0, 1, \cdots, |U| + |V|\}$.

Output All d-factor in G.

Complexity $O(|E|)$ delay.

Comment A d-factor in G is a subgraph $G' = (U, V, X)$ covering all vertices of G, whose each vertex v has degree $d(v)$. If for any $v \in U \cup V$, $d(v) = 1$, G' is a perfect matching. I'm looking for this paper.

Reference [30]

2.19.11 Enumeration of all maximal induced matchings in a triangle-free graph

Input A triangle-free graph G.

Output All maximal induced matchings in G.

Complexity $O(1.4423^n)$ total time with polynomial delay.

Comment n is the number of vertices in G.

Reference [15]

2.20 Matroid

2.20.1 Enumeration of all bases of a graphic matroid in a graph

Input A graph $G = (V, E)$.

Output All bases of a graphic matroid in G.

Complexity $O(|V| + |E| + N)$ total time and $O(|V| + |E|)$ space.

Comment N is the number of solutions. If G is connected, any base is a spanning tree.

Reference [242]

2.20.2 Enumeration of all bases of a linear matroid in a graph

Input A graph $G = (V, E)$.

Output All bases of a linear matroid in G.

Complexity $O(|V|)$ time per solution and $O(|V|^2|E|)$ preprocessing after time.

Reference [242]

2.20.3 Enumeration of all bases of a matching matroid in a graph

Input A graph $G = (V, E)$.

Output All bases of a matching matroid in G.

Complexity $O(|V| + |E|)$ time per solution.

Reference [242]

2.21 Ordering

2.21.1 Enumeration of all topological sortings of a directed graph

Input A directed graph $G = (V, E)$.

Output All topological sortings of G.

Complexity $O(|V| + |E|)$ time per sorting and $O(|V| + |E|)$ space.

Reference [134]

2.21.2 Enumeration of all topological sortings of a given set in lexicographically

Input An n-element set S.

Output All topological sortings of S in lexicographically.

Complexity $O(m)$ time per solution(?).

Reference [133]

2.21.3 Enumeration of all topological sortings of a po set

Input A partial order set P.

Output All topological sortings of P.

Complexity $O(|P|)$ time per solution.

Comment $|P|$ is the number of objects in P.

Reference [248]

2.21.4 Enumeration of all topological sortings in a directed acyclic graph

Input A directed graph G.

Output All topological sortings in G.

Complexity $O(1)$ amortized time per solution with $O(|G|)$.

Comment Linear extensions correspond to topological sortings.

Reference [191]

2.21.5 Enumeration of all topological sortings

Input A graph G.

Output All topological sortings in G.

Complexity $O(1)$ amortized time per solution.

Comment A topological sorting is also known as a linear extension.

Reference [199]

2.21.6 Enumeration of all topological sortings

Input A directed acyclic graph D.

Output All topological sortings D.

Complexity $O(1)$ amortized time per topological sorting and $O(|V|)$ space in addition to the space used for D.

Comment Linear sortings correspond to topological sortings.

Reference [190]

2.21.7 Enumeration of all linear extensions of a given poset

Input A poset P.

Output All linear extensions of P.

Complexity $O(1)$ time per solution.

Comment Their algorithm is a loop-free algorithm.

Reference [38]

2.21.8 Enumeration of all topological sortings of an acyclic directed graph

Input An acyclic directed graph $G = (V, E)$.

Output All topological sortings of G.

Complexity $O(|V|N)$ total time and $O(|V||E|)$ space.

Comment N is the number of solutions.

Reference [7]

2.21.9 Enumeration of all perfect elimination orderings

Input A chordal graph G.

Output All perfect elimination orderings of G.

Complexity Constant amortized time per solution.

Reference [41]

2.21.10 Enumeration of all forest extensions of a partially ordered set

Input A partially ordered set P.

Output All forest extensions of P.

Complexity $O(|E|^2)$ delay and $O(|E||R|)$ space.

Comment E is the set of elements. R is the binary relation on E.

Reference [231]

2.21.11 Enumeration of all topological sortings of a directed acyclic graph

Input A directed acyclic graph D.

Output All topological sortings D.

Complexity $O(1)$ delay per topological sorting.

Comment Linear extensions correspond to topological sortings.

Reference [181]

2.21.12 Enumeration of all realizer of a triangulated planar graph

Input A triangulated planar graph $G = (V, E)$.

Output All realizer of G.

Complexity $O(|V|)$ time per realizer.

Comment I'm looking for this paper.

Reference [267]

2.21.13 Enumeration of all perfect elimination orderings of a chordal graph

Input A chordal graph $G = (V, E)$.

Output All perfect elimination orderings of G.

Complexity $O(1)$ time per solution on average with $O(|V|^2)$ space and $O(|V|^3)$ with $O(|V|^2)$ space pre-computation.

Reference [157]

2.21.14 Enumeration of all perfect sequences in a chordal graph

Input A chordal graph $G = (V, E)$.

Output All perfect sequences of G.

Complexity $O(1)$ time per graph with $O(|V|^2)$ space with $O(|V|^3)$ time and $O(|V|^2)$ space pre-computation.

Reference [158]

2.22 Orientation

2.22.1 Enumeration of all acyclic orientation of a graph

Input A graph $G = (V, E)$.

Output All acyclic orientation of G.

Complexity $O(N(|V|+|E|))$ total time ($O(|V|(|V|+E|))$ delay) and $O(|V|+|E|)$ space.

Comment A acyclic orientation of G is an assignment of directions of each edge such that G is acyclic.

Reference [12]

2.22.2 Enumeration of all (s,t)-orientations of a biconnected planar graph

Input A biconnected planar graph $G = (V, E)$ and an edge (s, t) in G.

Output All (s,t)-orientations of G.

Complexity $O(|V|)$ time per solution.

Comment I'm looking for this paper.

Reference [222]

2.23 Other

2.23.1 Enumeration of all Hamiltonian centers in a graph

Input A graph G.

Output All Hamiltonian centers in G.

Reference [272]

2.23.2 Enumeration of all CA-sets of a directed graph

Input Directed graph $G = (V, E)$.

Output All CA-sets of G.

Complexity $O(|V|^{2.49+} + \gamma)$.

Comment γ is the output size. $S \subset V$ is a *CA-set* if, for each $v \in S$, all ancestor of v belongs to S.

Reference [120]

2.23.3 Enumeration of all maximal induced subgraphs for (connected) hereditary graph properties

Input A graph G.

Output All maximal induced subgraphs in $\mathcal{P}(G)$.

Complexity See the paper.

Comment $\mathcal{P}$ is a set of subgraphs of G with (connected) hereditary graph properties.

Reference [48]

2.24 Path

2.24.1 Enumeration of all simple paths in a graph

Input A graph G.

Output All simple paths in G.

Complexity

Reference [186]

2.24.2 Enumeration of all Hamiltonian paths in a graph

Input A graph G.

Output All Hamiltonian paths in G.

Reference [272]

2.24.3 Enumeration of all directed paths in a directed graph

Input A directed graph G.

Output All directed paths in G.

Reference [114]

2.24.4 Enumeration of all paths in a graph

Input A graph G.

Output All paths in G.

Reference [139]

2.24.5 Enumeration of all paths in a graph

Input A graph G.

Output All paths in G.

Reference [53]

2.24.6 Enumeration of k shortest paths in a graph

Input A graph $G = (V, E)$.

Output K shortest paths in G.

Complexity $O(|V|^3)$ total time.

Comment I'm looking for this paper.

Reference [274]

2.24.7 Enumeration of k shortest paths in a graph

Input A graph $G = (V, E)$.

Output k shortest paths in G.

Complexity $O(k|V|c(|V|))$ total time.

Comment (?) $c(n)$ is the time complexity to find an optimal solution to a problem with n (0, 1) variables. I'm looking for this paper.

Reference [142]

2.24.8 Enumeration of all paths in a graph

Input A graph $G = (V, E)$.

Output All paths in G.

Complexity $O(|E|)$ time per path with $O(|E|)$ space.

Reference [193]

2.24.9 Enumeration of all paths in a directed graph

Input A directed graph $G = (V, E)$.

Output All paths in G.

Complexity $O(|E|)$ time per path with $O(|E|)$ space.

Reference [193]

2.24.10 Enumeration of k shortest paths in a graph

Input A graph $G = (V, E)$.

Output K shortest paths in G.

Complexity $O(k|V|^3)$ total time.

Comment I'm looking for this paper.

Reference [224]

2.24.11 Generation of the k-th longest path in a tree

Input A tree $T = (V, E)$ and an integer k.

Output The k-th longest path in T.

Complexity $O(n \log^2 n)$ time.

Comment I'm looking for this paper.

Reference [164]

2.24.12 Enumeration of all shortest paths in a graph

Input A graph G.

Output All shortest paths in G.

Comment I'm looking for this paper.

Reference [77]

2.24.13 Enumeration of k shortest paths of a directed graph

Input A graph $G = (V, E)$.

Output k shortest paths that may contains cycles in G.

Reference [151]

2.24.14 Enumeration of all quickest paths in a network

Input A network $N = (V, E, c, \ell)$.

Output All quickest paths in N.

Complexity $O(rS|V||E| + rS|V|^2 \log |V|)$ total time.

Comment c is a positive edge weight function and ℓ is a nonnegative edge weight function. r is the number of distinct capacity value of N. S is the number of solutions.

Reference [194]

2.24.15 Counting all acyclic walks in a graph

Input A graph $G = (V, E)$.

Output The number of acyclic walks in G.

Reference [9]

2.24.16 Enumeration of the k shortest paths in a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k smallest shortest paths in G.

Complexity $O(k|E|)$ total time.

Reference [8]

2.24.17 Enumeration of all constrained quickest paths in a network

Input Network $N = (V, E)$ and constraints L and C.

Output All quickest paths in N.

Complexity $O(k|V|^2|E|)$ total time.

Comment k is the number of solutions. A quickest path is a variant of a shortest path.

Reference [91]

2.24.18 Enumeration of the k shortest paths in a directed graph

Input A directed graph $G = (V, E)$ and an integer k.

Output The k smallest shortest paths in G.

Complexity $O(k|E|)$ total time

Reference [63]

2.24.19 Enumeration of all minimal path conjunctions in a graph

Input A directed graph $G = (V, E)$, $s_1, s_2, t_1 \in V$, $T_2 \subseteq V$, and $\mathcal{P} = \{(s_1, t_1)\} \cup \{(s_2, t) : t \in T_2\}$.

Output All minimal path conjunctions in G.

Complexity Polynomial delay.

Comment A path conjunction is a edge subset $E' \subseteq E$ such that for all $(s, t) \in \mathcal{P}$, s is connected to t in the graph $G' = (V, E')$.

Reference [29]

2.24.20 Enumeration of all st-paths in a graph

Input A graph $G = (V, E)$ and $s, v \in V$.

Output All st-paths in G.

Complexity $O(|E| + \sum_{\pi \in \mathcal{P}_{st}(G)} |\pi|)$ total time.

Comment $\mathcal{P}_{st}(G)$ is the set of all st-paths in G.

Reference [23]

2.24.21 Enumeration of all P_3's in a graph

Input A graph G.

Output All of all P_3's in G.

Complexity $O(|E|^{1.5} + p_3(G))$ total time.

Comment P_3 is a induced path of G with three vertices and $p_3(G)$ is the number of P_3 in G.

Reference [104]

2.24.22 Enumeration of all P_k's in a graph

Input A graph G and an integer $k \geq 4$.

Output All of all P_k's in G.

Complexity $O(|V|^{k-1} + p_k(G) + k \cdot c_k(G))$ total time.

Comment P_k and C_k are a induced path and cycle of G with k vertices, respectively. $p_k(G)$ and $c_k(G)$ are the number of P_k and C_k in G, respectively.

Reference [104]

2.24.23 Enumeration of all C_k's in a graph

Input A graph G and an integer $k \geq 4$.

Output All of all C_k's in G.

Complexity $O(|V|^{k-1} + p_k(G) + c_k(G))$ total time.

Comment P_k and C_k are a induced path and cycle of G with k vertices, respectively. $p_k(G)$ and $c_k(G)$ are the number of P_k and C_k in G, respectively.

Reference [104]

2.24.24 Enumeration of all chordless st-paths in a graph

Input A graph $G = (V, E)$ and two vertices $s, t \in V$.

Output All chordless st-paths in G.

Complexity $\tilde{O}(|E| + |V| \cdot P)$ total time.

Comment P is the number of chordless st-paths in G. I'm looking for this paper.

Reference [76]

2.24.25 Enumeration of all chordless st-paths in a graph

Input A graph $G = (V, E)$ and $s, t \in V$.

Output All chordless st-paths (from s to t) in G.

Complexity $O(|V| + |E|)$ time per chordless st-path.

Reference [247]

2.25 Permutation graph

2.25.1 Enumeration of all connected bipartite permutation graphs with n vertices

Input A graph size n.

Output All connected bipartite permutation graphs.

Complexity $O(1)$ time per graph with $O(n)$ space.

Reference [208]

2.26 Pitch

2.26.1 Enumeration of all stories in a graph

Input A directed graph $G = (V, E, S, T)$ that has the set of source vertices S and the set of target vertices of T.

Output All stories in G.

Comment A *pitch* P of G is a set of arcs $E' \subseteq E$, such that the subgraph $G' = (V', E')$ of G, where $V' \subseteq V$ is the set of vertices of G having at least one out-going or in-coming arc in E', is acyclic and for each vertex $w \in V' \backslash S$, w is not a source in G', and for each vertex $w \in V' \backslash T$,w is not a target in G'. P is a *story* if P is maximal.

Reference [1]

2.26.2 Enumeration of all pitches

Input A directed graph $G = (V, E, S, T)$ that has the set of source vertices S and the set of target vertices of T.

Output All pitches in G.

Complexity $O(|V| + |E|)$ delay with $O(|V| + |E|)$ space.

Comment A pitch P of G is a set of arcs $E' \subseteq E$, such that the subgraph $G' = (V', E')$ of G, where $V' \subseteq V$ is the set of vertices of G having at least one out-going or in-coming arc in E', is acyclic and for each vertex $w \in V' \backslash S$, w is not a source in G', and for each vertex $w \in V' \backslash T$,w is not a target in G'. Enumeration of all stories (maximal pitches) is still open.

Reference [25]

2.27 Planar graph

2.27.1 Enumeration of all maximal planar graphs with n vertices

Input An integer n.

Output All maximal planar graph with n vertices.

Complexity $O(n^3)$ time per graph with $O(n)$ space.

Comment A planar graph with n vertices is *maximal* if it has exactly $3n-6$ edges.

Reference [145]

2.27.2 Enumeration of all based floorplans with at most n faces

Input An integer n.

Output All based floorplans with at most n faces.

Complexity $O(1)$ time per solution with $O(n)$ space.

Comment A planar graph is called a *floorplan* if every face is a rectangle. A *based floorplan* is a floorplan with one designated base line segment on the outer face.

Reference [171]

2.27.3 Enumeration of all based floorplans with exactly n faces

Input An integer n.

Output All based floorplans with exactly n faces.

Complexity $O(1)$ time per solution with $O(n)$ space.

Comment A planar graph is called a *floorplan* if every face is a rectangle. A *based floorplan* is a floorplan with one designated base line segment on the outer face.

Reference [171]

2.27.4 Enumeration of all floorplans with exactly n faces

Input An integer n.

Output All floorplans with exactly n faces.

Complexity $O(1)$ time per solution with $O(n)$ space.

Comment A planar graph is called a *floorplan* if every face is a rectangle.

Reference [171]

2.27.5 Enumeration of all internally triconnected planar graphs

Input Integers n and g.

Output All internally triconnected planar graphs with exactly n vertices such that $\kappa(G) = 2$ and the size of each inner face is at most g.

Complexity $O(n^3)$ time per solution on average with $O(n)$ space.

Reference [280]

2.28 Plane graph

2.28.1 Enumeration of all plane straight-line graphs on a given point set in the plane

Input A point set P in the plane.

Output All plane straight-line graphs on P.

Complexity $O(|P| \log |P|)$ time per solution.

Comment Use gray code.

Reference [2]

2.28.2 Enumeration of all plane and connected straight-line graphs on a given point set in the plane

Input A point set P in the plane.

Output All plane and connected straight-line graphs on P.

Complexity $O(|P| \log |P|)$ time per solution.

Comment Use gray code.

Reference [2]

2.28.3 Enumeration of all plane spanning trees on a given point set in the plane

Input A point set P in the plane.

Output All plane spanning trees on P.

Complexity $O(|P| \log |P|)$ time per solution.

Comment Use gray code.

Reference [2]

2.28.4 Enumeration of all plane graphs

Input m: the maximum number of edges.

Output All connected rooted plane graphs with at most m edges.

Complexity amortized $O(1)$ time per graph with $O(m)$ space.

Comment This algorithm does not outputs the entire graph but the difference from previous one.

Reference [268]

2.28.5 Enumeration of all plane graphs

Input m: the maximum number of edges.

Output All connected non-rooted plane graphs with at most m edges.

Complexity $O(m^3)$ time per graph with $O(m)$ space.

Comment This algorithm does not outputs the entire graph but the difference from previous one.

Reference [268]

2.28.6 Enumeration of all plane graphs on a given point set in the plane

Input A fixed point set P.

Output All plane graphs on P.

Complexity $O(N)$ total time.

Comment N is the number of solutions.

Reference [122]

2.28.7 Enumeration of all non-crossing spanning connected graphs on a given point set in the plane

Input A fixed point set P.

Output All non-crossing spanning connected graphs on P.

Complexity $O(N)$ total time.

Comment N is the number of solutions.

Reference [122]

2.28.8 Enumeration of all non-crossing spanning trees on a given point set in the plane

Input A fixed point set P.

Output All non-crossing spanning trees on P.

Complexity $O(N + |P| tri(P))$ total time.

Comment N is the number of solutions and $tri(P)$ is the number of triangulations of P.

Reference [122]

2.28.9 Enumeration of all non-crossing minimally rigid frameworks on a given point set in the plane

Input A fixed point set P.

Output All non-crossing minimally rigid frameworks on P.

Complexity $O(|P|^2 N)$ total time.

Comment N is the number of solutions.

Reference [122]

2.28.10 Enumeration of all non-crossing perfect matchings on a given point set in the plane

Input A fixed point set P.

Output All non-crossing perfect matchings on P.

Complexity $O(|P|^{3/2}tri(P) + |P|^{5/2}N)$ total time.

Comment N is the number of solutions and $tri(P)$ is the number of triangulations of P.

Reference [122]

2.28.11 Enumeration of all triconnected rooted plane graphs

Input Integers n and g.

Output All triconnected rooted plane graphs with n vertices, whose each inner face has the length at most g.

Complexity $O(1)$ delay with $O(n)$ space after $O(n)$ time preprocessing.

Reference [281]

2.28.12 Enumeration of all triconnected rooted plane graphs

Input An integer n.

Output All triconnected rooted plane graphs with n vertices.

Complexity $O(n^3)$ delay with $O(n)$ space after $O(n)$ time preprocessing.

Reference [281]

2.28.13 Enumeration of all biconnected rooted plane graphs

Input Integers n and g.

Output All biconnected rooted plane graphs with exactly n vertices such that each inner face is of length at most g.

Complexity $O(1)$ delay with $O(n)$ space, after an $O(n)$ time preprocessing.

Reference [279]

2.28.14 Enumeration of all biconnected plane graphs

Input Integers n and g.

Output All biconnected plane graphs with at most n vertices such that each inner face is of length at most g.

Complexity $O(n^3)$ time per solution on average with $O(n)$ space.

Reference [279]

2.28.15 Enumeration of all biconnected rooted plane graphs

Input Integers n and g.

Output All biconnected rooted plane graphs with at most n vertices such that each inner face is of length at most g.

Complexity $O(1)$ delay with $O(n)$ space.

Reference [279]

2.28.16 Enumeration of all rooted plane graphs in $\mathcal{G}_{\mathbf{int}}(n, g) - \mathcal{G}_3(n, g)$

Input Integers n and g.

Output All rooted plane graphs in $\mathcal{G}_{\text{int}}(n, g) - \mathcal{G}_3(n, g)$

Complexity $O(1)$ delay with $O(n)$ space and time preprocessing.

Reference [280]

2.29 Polytope

2.29.1 Enumeration of all 3-polytopes of a graph

Input A graph $G = (V, E)$.

Output All 3-polytopes of G.

Reference [55]

2.30 Quadrangle

2.30.1 Enumeration of all quadrangles in a graph

Input A connected graph $G = (V, E)$.

Output All quadrangles in G.

Complexity $O(\alpha(G)|E|)$ total time and $O(|E|)$ space.

Comment $\alpha(G)$ is the minimum number of edge-disjoint spanning forests into which G can be decomposed. I'm looking for this paper.

Reference [47]

2.31 Quadrangulation

2.31.1 Enumeration of all based biconnected plane quadrangulations with at most f faces

Input An integer f.

Output All based biconnected plane quadrangulations with at most f faces.

Complexity $O(1)$ time per quadrangulation and $O(f)$ space.

Comment A *plane quadrangulation* is a plane graph such that each inner face has exactly four edges on its contour. A *based plane quadrangulation* is a plane quadrangulation with one designated edge on the outer face. I'm looking for this paper.

Reference [144]

2.32 Regular graph

2.32.1 Enumeration of all cubic graphs with less than or equal to n vertices

Input An integer n.

Output All cubic graphs with less than or equal to n vertices.

Reference [34]

2.33 Series-parallel

2.33.1 Enumeration of all series-parallel graphs with at most m edges

Input An integer m.

Output All series-parallel graphs with at most m edges.

Complexity $O(m)$ time per graph.

Comment I'm looking for this paper.

Reference [124]

2.34 Spanning subgraph

2.34.1 Enumeration of all minimal k-vertex connected spanning subgraphs in a k-connected graph.

Input A k-connected graph G.

Output All minimal k-vertex connected spanning subgraphs in G.

Complexity $O(K^3|E|^3|n| + K^2|E|^5|V|^4 + K|V|^k|E|^2)$ total time.

Comment K is the number of solutions. A graph G is k-connected if a subgraph of G obtained by removing at most $k-1$ vertices is still connected.

Reference [31]

2.35 Spanning tree

2.35.1 Enumeration of all spanning trees in a graph

Input A graph G.

Output All spanning trees in G.

Complexity

Reference [99]

2.35.2 Enumeration of all spanning trees in a graph

Input A graph $G = (V, E)$.

Output All spanning trees of G.

Comment In this paper, 'trees' indicate 'spanning trees'.

Reference [165]

2.35.3 Enumeration of all spanning trees of a graph

Input A graph G.

Output All spanning trees of G.

Reference [160]

2.35.4 Enumeration of all spanning trees in a graph

Input A graph $G = (V, E)$.

Output All spanning trees in G.

Complexity $O(|V||E|^2)$ time per spanning tree with $O(|V||E|)$ space.

Reference [193]

2.35.5 Enumeration of the k smallest weight spanning trees in a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k smallest weight spanning trees in G.

Complexity $O(k|E|\alpha(|E|, |V|) + |E| \log |E|)$ total time and $O(k + |E|)$ space, $\alpha(\cdot)$ is Tarjan's inverse of Ackermann's function.

Reference [86]

2.35.6 Enumeration of all spanning trees in a graph

Input A graph $G = (V, E)$ and an integer k.

Output The all spanning trees in G in order.

Complexity $O(N|V|)$ total time and $O(N + |E|)$ space, N is the number of spanning trees in G.

Reference [86]

2.35.7 Enumeration of all spanning trees in an undirected graph

Input An undirected graph $G = (V, E)$.

Output All spanning trees in G.

Complexity $O(|V| + |E| + |V|N)$ total time and $O(|V| + |E|)$ space.

Comment N is the number of spanning trees in G.

Reference [87]

2.35.8 Enumeration of all spanning trees in a directed graph

Input A directed graph $G = (V, E)$.

Output All spanning trees in G.

Complexity $O(|V| + |E| + |E|N)$ total time and $O(|V| + |E|)$ space.

Comment N is the number of spanning trees in G.

Reference [87]

2.35.9 Enumeration of the k smallest weight spanning trees in a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k smallest weight spanning trees in G.

Complexity $O(k|E| + \min(|V|^2, |E| \log \log |V|))$ total time and $O(k + |E|)$ space

Reference [121]

2.35.10 Enumeration of all spanning trees in an undirected graph

Input An undirected graph G.

Output All spanning trees in G.

Comment They analized Char's enumeration algorithm.

Reference [109]

2.35.11 Enumeration of the k smallest weight spanning trees in a graph in increasing order

Input A graph $G = (V, E)$ and an integer k.

Output The k smallest weight spanning trees in G.

Complexity $O(|E| \log \log_{(2+|E|/|V|)} n + k^2\sqrt{|E|})$ total time and $O(|E| + k\sqrt{|E|})$ space.

Reference [79]

2.35.12 Enumeration of the k smallest weight spanning trees in a planar graph in increasing order

Input A planar graph $G = (V, E)$ and an integer k.

Output The k smallest weight spanning trees in G.

Complexity $O(|V| + k^2(\log |V|)^2)$ total time and $O(|V| + k(\log |V|)^2)$ space.

Reference [79]

2.35.13 Enumeration of all undirected minimum spanning trees in an undirected graph

Input An undirected graph $G = (V, E)$.

Output All undirected minimum spanning trees in G.

Complexity $O(|E| \log \beta(|E|, |V|))$ total time.

Comment $\beta(|E|, |V|) = \min\{i | \log^{(i)} |V| \leq |E|/|N|\}$.

Reference [88]

2.35.14 Enumeration of all directed minimum spanning trees in an directed graph

Input A directed graph $G = (V, E)$.

Output All directed minimum spanning trees in G.

Complexity $O(|V| \log \beta(|E|, |V|))$ total time.

Comment $\beta(|E|, |V|) = \min\{i | \log^{(i)} |V| \leq |E|/|N|\}$.

Reference [88]

2.35.15 Enumeration of all spanning tree in a graph

Input A graph $G = (V, E)$.

Output All spanning tree of G.

Complexity $O(|V| + |E| + N)$ total time and $O(|V||E|)$ space.

Comment N is the number of spanning trees in G.

Reference [119]

2.35.16 Enumeration of all spanning tree in an weighted graph

Input An weighted graph $G = (V, E)$.

Output All spanning tree of G in increasing order of weight.

Complexity $O(N \log |V| + |V||E|)$ total time and $O(N + |V|^2|E|)$ space.

Comment N is the number of spanning trees in G.

Reference [119]

2.35.17 Enumeration of all spanning tree in a directed graph

Input A directed graph $G = (V, E)$.

Output All spanning tree of G.

Complexity $O(N|V| + |V|^3)$ total time and $O(|V|^2)$ space.

Comment N is the number of spanning trees in G.

Reference [119]

2.35.18 Enumeration of the k best spanning trees in a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k best spanning trees of G.

Complexity $O(m \log \beta(|E|, |V|) + k^2)$ total time.

Comment $\beta(|E|, |V|) = \min\{\log^{(i)} |V| \leq |E|/|V|\}$.

Reference [64]

2.35.19 Enumeration of the k best spanning trees in a planar graph

Input A planar graph $G = (V, E)$ and an integer k.

Output The k best spanning trees of G.

Complexity $O(n + k^2)$ total time

Reference [64]

2.35.20 Generation of the k-th minimum spanning tree in a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k-th minimum spanning tree of G.

Complexity $O((|V||E|)^{k-1})$ time.

Reference [161]

2.35.21 Enumeration of all spanning trees

Input A graph $G = (V, E)$.

Output All spanning trees in G.

Complexity $O(N + |V| + |E|)$ total time and $O(|V||E|)$ space.

Comment N is the number of spanning trees in G.

Reference [225]

2.35.22 Enumeration of all spanning trees in a directed graph

Input A directed graph $G = (V, E)$.

Output All spanning trees in G.

Complexity INCORRECT: $O(N \log |V| + |V|^2 \alpha(V, V) + |V||E|)$

Comment α: the inverse Ackermann's function. [KR2000] gives this result is wrong.

Reference [101]

2.35.23 Enumeration of all spanning trees in a directed graph

Input A directed graph $G = (V, E)$.

Output All spanning trees in G.

Complexity $O(E + ND(V, E))$ total time and $O(E + DS(V, E))$ space.

Comment $D(V, E)$ and $DS(V, E)$ are the time and space complexities of the data structure for updating the minimum spanning tree in an undirected graph with V vertices and E edges. Here N denotes the number of directed spanning trees in G.

Reference [244]

2.35.24 Enumeration of all spanning trees in a graph

Input A graph $G = (V, E)$.

Output All spanning trees included in G.

Complexity $O(N + |V| + |E|)$ total time and $O(|V| + |E|)$ space.

Comment N = number of spanning trees in G.

Reference [226]

2.35.25 Enumeration of the k smallest weight spanning trees in a graph

Input A graph $G = (V, E)$ and an integer k.

Output The k smallest weight spanning trees in G.

Complexity $O(m \log \log *n + k \min(n, k)^{1/2})$ total time, or a randomized version taking $O(m + k \min(n, k)^{1/2})$ total time.

Reference [68]

2.35.26 Enumeration of all spanning trees in a graph

Input A graph $G = (V, E)$.

Output All spanning trees in G.

Complexity $O(|V| + |E| + \tau)$ time and $O(|V| + |E|)$ space (depth first manner) or $O(\tau|V| + |E|)$ space (breadth first manner).

Comment By using breadth first manner, the proposed algorithm can be used in a parallel computer.

Reference [154]

2.35.27 Enumeration of all spanning trees in a graph in non-decreasing order

Input A graph $G = (V, E)$.

Output All spanning trees in G in non-decreasing order.

Complexity $O(|V| + |E| + \tau)$ time and $O(\tau|V| + |E|)$ space.

Comment Using breadth first manner.

Reference [154]

2.35.28 Enumeration of all directed spanning trees in a directed graph

Input A directed graph $G = (V, E)$.

Output All directed spanning trees in G.

Complexity $O(|E| \log |V| + |V| + N \log^2 |V|)$ total time and $O(|E| + |V|)$ space.

Comment N is the number of directed spanning trees.

Reference [246]

2.35.29 Enumeration of all spanning trees of an weighted graph in order of increasing cost

Input An weighted graph $G = (V, E)$.

Output All spanning trees of G in order of increasing cost.

Complexity $O(N|E| \log |E| + N^2)$ total time and $O(N|E|)$ space.

Comment N is the number of spanning trees of G.

Reference [227]

2.35.30 Enumeration of all the minimum spanning trees in a graph

Input An weighted graph $G = (V, E)$.

Output All the minimum spanning trees in G.

Complexity $O(N|E| \log |V|)$ total time and $O(|E|)$ space.

Comment N is the number of the minimum spanning trees in G.

Reference [265]

2.36 Steiner tree

2.36.1 Enumeration of all Steiner W-trees in a connected graph

Input A connected graph $G = (V, E)$, a vertex set $W \subseteq V$ such that $|W| = k$, for a fixed integer k.

Output Enumeration of all Steiner W-trees in G.

Complexity $O(|V|^2(|V|+|E|)+|V|^{k-2}+N|V|)$ total time with $O(|V|^{k-2}+|V|^2(|V| + |E|))$ space.

Comment A connected subgraph T of G is a Steiner W-tree if $W \subseteq V(T)$ and $|E(T)$ is minimum.

Reference [58]

2.37 Subforest

2.37.1 Enumeration of all k-trees in a graph

Input A graph $G = (V, E)$.

Output All k-trees in G.

Comment A k-tree is a forest with k connected components.

Reference [98]

2.38 Subgraph

2.38.1 Enumeration of all connected induced subgraph of a graph

Input A graph $G = (V, E)$.

Output All connected induced subgraph of G.

Complexity $O(|V||E|N)$ total time and $O(|V| + |E|)$ space.

Comment N is the number of solutions.

Reference [7]

2.38.2 Enumeration of all connected common maximal subgraphs in two graphs

Input Two graphs G and G'.

Output All connected common maximal subgraphs in G and G'.

Reference [135]

2.38.3 Enumeration of all minimal spanning graph

Input A graph $G = (V, E)$, $S \subseteq V$, and requirements $r(u, v)$ for all $(u, v) \in V \times V$.

Output All minimal spanning graph H of G satisfying $\lambda_H^S \geq r(u, v)\ \forall (u, v) \in V \times V$.

Complexity Incremental polynomial time.

Comment The S-connectivity $\lambda_G^S(u, v)$ of (u, v) in G is the maximum number of uv-paths such that no two of them have an edge or a node in $S \setminus \{u, v\}$ in common. This complexity holds for edge-connectivity.

Reference [178]

2.38.4 Enumeration of all k-outconnected minimal spanning graph

Input A graph $G = (V, E)$, a vertex $s \in V$, and an integer k.

Output All minimal k-outconneted from s spanning subgraph of G.

Complexity Incremental polynomial time.

Comment A graph is k-outconnected from s if it contains k internally-disjoint st-paths for every $t \in V$. This complexity holds for both vertex and edge-connectivity.

Reference [178]

2.38.5 Enumeration of all k-outconnected minimal spanning graph

Input A directed graph $G = (V, E)$, a vertex $s \in V$, and an integer k.

Output All minimal k-outconneted from s spanning subgraph of G.

Complexity Incremental polynomial time.

Comment A graph is k-outconnected from s if it contains k internally-disjoint st-paths for every $t \in V$. This complexity holds for both vertex and edge-connectivity.

Reference [178]

2.38.6 Enumeration of all k-connected minimal spanning graph

Input A directed graph $G = (V, E)$ and an integer k.

Output All minimal k-connected spanning subgraph of G.

Complexity Incremental polynomial time.

Comment This complexity holds for both vertex and edge-connectivity.

Reference [178]

2.39 Subtree

2.39.1 Enumeration of all subtrees in an input tree

Input A tree $T = (V, E)$.

Output All subtrees included in T.

Complexity $O(|V|)$ delay and $O(|V|)$ space.

Reference [201]

2.39.2 Enumeration of all k-noded subtrees in a tree

Input A tree T and an integer k.

Output All k-noded subtrees in T.

Reference [103]

2.39.3 Enumeration of all k-subtrees in a graph

Input A graph $G = (V, E)$ and a positive integer k.

Output All k-subtrees included in G.

Complexity $O(sk)$ total time, $O(k)$ amortized time per solution, and $O(|E|)$ space.

Comment s = number of k-subtrees in G, a k-subtree means a connected, acyclic, and edge induced subgraph with k vertices.

Reference [75]

2.39.4 Enumeration of all k-subtrees in an input tree

Input A tree $T = (V, E)$ and an integer k.

Output All k-subtrees included in T.

Complexity $O(1)$ delay and $O(|V|)$ space after $O(|V|)$ time preprocessing.

Comment A k-subtree is a connected, acyclic, and edge induced subgraph with k vertices. I'm looking for this paper.

Reference [253]

2.39.5 Enumeration of all k-cardinarity subtrees of a tree with w vertices

Input An integer k and a tree T with w element, where $k \leq w$.

Output All subtrees with k vertices of T.

Complexity $O(Nw^5)$ total time.

Comment N is the number of ideals.

Reference [262]

2.39.6 Enumeration of all induced subtrees in a k-degenerate graph

Input A k-degenerate graph $G = (V, E)$.

Output All induced subtrees in G.

Complexity $O(k)$ amortized time per solution with $O(|V|+|E|)$ space and preprocessing time.

Reference [252]

2.40 Tour

2.40.1 Enumeration of k best solutions to the Chinese postman problem solutions

Input A graph $G = (V, E)$.

Output K best solutions to the Chinese postman problem.

Complexity $O(S(n,m)+K(n+m+\log k+nT(n+m,m)))$ where $S(s,t)$ denotes the time complexity of an algorithm for ordinary Chinese postman problems and $T(s,t)$ denotes the time complexity of a post-optimal algorithm for non-bipartite matching problems defined on a graph with s vertices and t edges.

Reference [210]

2.41 Tree

2.41.1 Enumeration of all binary trees with fixed number leaves in lexicographically

Input An integer n.

Output All binary trees with n leaves in lexicographically.

Complexity $O(1)$ time per binary tree.

Reference [197]

2.41.2 Enumeration of all t-ary trees with fixed number leaves in lexicographically

Input An integer n.

Output All t-ary trees with n leaves in lexicographically.

Complexity $O(t)$ time per binary tree.

Reference [200]

2.41.3 Enumeration of all k-ary trees with n vertices

Input Integers k and n.

Output All k-ary trees with n vertices

Complexity $O(1)$ time per solution

Reference [238]

2.41.4 Enumeration of all binary trees with n vertices

Input An integer n.

Output All binary trees with n vertices.

Reference [196]

2.41.5 Enumeration of all k-ary trees with n vertices

Input Integers k and n.

Output All k-ary trees with n vertices

Reference [277]

2.41.6 Enumeration of all rooted trees with n vertices

Input An integer n.

Output All rooted trees with n vertices.

Complexity $O(1)$ amortized time per solution.

Reference [21]

2.41.7 Enumeration of all binary trees with n vertices

Input An integer n.

Output All binary trees with n vertices.

Reference [187]

2.41.8 Enumeration of all k-ary trees in lexicographically

Input An integer n.

Output All k-ary trees with n internal vertices in lexicographically

Complexity $O(1-(k-1)^{k-1}/k^k)^{-1}$ time per solution. This limit is 4/3 for the binary case.

Reference [276]

2.41.9 Enumeration of all regular k-ary trees with n ndoes

Input Integers k and n.

Output All regular k-ary trees with n ndoes.

Reference [275]

2.41.10 Enumeration of all binary trees with n vertices in the lexicographic ordering

Input An integer n.

Output All binary trees with n vertices in the lexicographic ordering

Complexity $O(1)$ time per solution on average.

Reference [278]

2.41.11 Enumeration of all binary trees with n vertices

Input An integer n.

Output All binary trees with n vertices.

Complexity $O(n)$ time per solution.

Reference [188]

2.41.12 Enumeration of all ordered trees with n internal vertices

Input An integer n.

Output All ordered trees with n internal vertices.

Reference [71]

2.41.13 Enumeration of all free trees with n vertices

Input An integer n.

Output All free trees with n vertices.

Complexity $O(1)$ time per solution.

Reference [263]

2.41.14 Enumeration of all t-ary trees with n vertices

Input Integers t and n.

Output All t-ary trees with n vertices.

Reference [72]

2.41.15 Enumeration of all binary trees with n vertices

Input An integer n.

Output All binary trees with n vertices.

Complexity $O(1)$ time per solution.

Reference [147]

2.41.16 Enumeration of all trees with n vertices and m leaves

Input Integers n and m.

Output All trees with n vertices and m leaves

Comment I'm looking for this paper.

Reference [182]

2.41.17 Enumeration of all t-ary trees in A-order

Input Integers t and n.

Output All t-ary trees with n vertices in A-order.

Complexity $O(1)$ amortized time per solution.

Reference [14]

2.41.18 Enumeration of all binary trees with n leaves

Input An integer n.

Output All binary trees with n leaves.

Complexity $O(1)$ amortized time per solution.

Comment A strong Gray code can be listed in constant average time per solution.

Reference [203]

2.41.19 Enumeration of all binary trees with n vertices

Input An integer n.

Output All binary trees with n vertices.

Complexity $O(1)$ time per solution.

Comment A loopless generation algorithm is an algorithm where the amount of computation to go from one object to the next is $O(1)$.

Reference [13]

2.41.20 Enumeration of all k-ary trees in natural order

Input Two integers k and n.

Output All k-ary trees with n vertices.

Reference [70]

2.41.21 Enumeration of all binary trees with n vertices

Input An integer n.

Output All binary trees with n vertices.

Complexity $O(1)$ delay.

Reference [148]

2.41.22 Enumeration of all binary trees with n vertices

Input An integer n.

Output All binary trees with n vertices.

Reference [11]

2.41.23 Enumeration of all k-ary tree

Input An integer k.

Output All k-ary trees.

Complexity $O(1)$ delay.

Reference [137]

2.41.24 Enumeration of all binary trees

Output All binary trees.

Complexity $O(1)$ time per tree.

Reference [264]

2.41.25 Enumeration of all k-ary trees with n vertices

Input Two integers k and n.

Output All k-ary trees with n vertices.

Complexity $O(1)$ delay.

Comment Shifts and loopless algorithm.

Reference [138]

2.41.26 Enumeration of all rooted plane trees with at most n vertices

Input An integer n.

Output All rooted plane trees with at most n vertices.

Complexity $O(1)$ time per tree with $O(n)$ space.

Comment A *rooted plane tree* is a rooted tree with a left-to-right ordering specified for the children of each vertex.

Reference [169]

2.41.27 Enumeration of all rooted plane trees with exactly n vertices

Input An integer n.

Output All rooted plane trees with exactly n vertices.

Complexity $O(1)$ time per tree with $O(n)$ space.

Comment A *rooted plane tree* is a rooted tree with a left-to-right ordering specified for the children of each vertex.

Reference [169]

2.41.28 Enumeration of all rooted plane trees with at most n vertices and the maximum degree D

Input Integers n and D.

Output All rooted plane trees with at most n vertices and the maximum degree D.

Complexity $O(1)$ time per tree with $O(n)$ space.

Comment A *rooted plane tree* is a rooted tree with a left-to-right ordering specified for the children of each vertex.

Reference [169]

2.41.29 Enumeration of all rooted plane trees with exactly n vertices and exactly c leaves

Input An integer n.

Output All rooted plane trees with exactly n vertices and exactly c leaves .

Complexity $O(n - c)$ time per tree with $O(n)$ space.

Comment A *rooted plane tree* is a rooted tree with a left-to-right ordering specified for the children of each vertex.

Reference [169]

2.41.30 Enumeration of all plane trees with exactly n vertices

Input An integer n.

Output All plane trees with exactly n vertices.

Complexity $O(n^3)$ time per tree with $O(n)$ space.

Comment A *plane tree* is a tree with a left-to-right ordering specified for the children of each vertex.

Reference [169]

2.41.31 Enumeration of all rooted trees with at most n vertices

Input An integer n.

Output All rooted tree with at most n vertices.

Complexity $O(1)$ time per tree and $O(n)$ space.

Comment I'm looking for this paper.

Reference [174]

2.41.32 Enumeration of all n-trees

Input An integer n.

Output All n-trees.

Complexity $O(n^4 N)$ total time.

Comment Reverse search. N is the number of solutions.

Reference [5]

2.41.33 Enumeration of all trees with n vertices and d diameter

Input Integers n and d.

Output All trees with n vertices and d diameter.

Complexity $O(1)$ time per tree with $O(n)$ space.

Comment By using the algorithm for each $d = 2, \ldots, n-1$, all trees can be enumerated.

Reference [173]

2.41.34 Enumeration of all c-tree with at most v vertices and diameter d

Input Integers n and d

Output All c-tree with at most v vertices and diameter d.

Complexity $O(1)$ time per tree.

Comment A tree is a c-tree if each vertex has a color $c \in \{c_1, \ldots, c_m\}$.

Reference [175]

2.41.35 Enumeration of all nonisomorphic rooted plane trees with n vertices

Input An integer n.

Output All nonisomorphic rooted plane trees with n vertices.

Complexity Constant amortized time per solution.

Reference [216]

2.41.36 Enumeration of all nonisomorphic free plane trees with n vertices

Input An integer n.

Output All nonisomorphic free plane trees with n vertices.

Complexity Constant amortized time per solution.

Reference [216]

2.41.37 Enumeration of all multitrees satisfying given constraints

Input A set Σ of labels, a function $val : \Sigma \to \mathbb{Z}^+$, and a feature vector g of level K.

Output All (Σ, val)-labeled multitrees T such that $f_K(T) = g$ and $deg(v; T) = val(\ell(v))$ for all vertices $v \in T$.

Comment This algorithm is for chemical graphs.

Reference [106]

2.41.38 Enumeration of all ordered trees with n vertices and k leaves

Input Integers n and k.

Output All ordered trees with n vertices and k leaves.

Complexity $O(1)$ delay and $O(n)$ space.

Reference [269]

2.41.39 Enumeration of all trees with specified degree sequence

Input A degree sequence D.

Output All trees with D.

Complexity $O(1)$ time per tree.

Comment I'm looking for this paper.

Reference [172]

2.42 Triangle

2.42.1 Enumeration of all minimal triangle graphs with a fixed number vertices

Input An integer n.

Output All minimal triangle graphs with n vertices.

Complexity

Reference [32]

2.42.2 Enumeration of all triangles in a graph

Input A graph $G = (V, E)$.

Output All triangles in G.

Complexity $O(\alpha(G)|E|)$ total time and linear space.

Comment $\alpha(G)$ is the minimum number of edge-disjoint spanning forests into which G can be decomposed. If G is planar, then the time complexity becomes $O(|V|)$. I'm looking for this paper.

Reference [47]

2.43 Triangulation

2.43.1 Enumeration of all triangulations of 2-sphere

Input A 2-sphere G.

Output All triangulations of G.

Complexity

Reference [33]

2.43.2 Enumeration of all r-rooted 2-connected triangulations of a planar graph

Input A planar graph $G = (V, E)$ and an integer r.

Output All r-rooted 2-connected triangulations of G.

Complexity $O(|V|^2 N)$ total time and $O(|V|)$ space.

Comment N is the number of solutions.

Reference [6]

2.43.3 Enumeration of all r-rooted 3-connected triangulations of a planar graph

Input A planar graph $G = (V, E)$ and an integer r.

Output All r-rooted 3-connected triangulations of G.

Complexity $O(|V|^2 N)$ total time and $O(|V|)$ space.

Comment N is the number of solutions.

Reference [6]

2.43.4 Enumeration of all based plane triangulations with n vertices

Input An integer n.

Output All based plane triangulations.

Complexity $O(1)$ time per based plane triangulation with $O(n)$ space.

Comment A *based plane triangulation* is a plane triangulation with one designated edge on the outer face. The algorithm does not output entire solution but output the difference from the previous solution.

Reference [145]

2.43.5 Enumeration of all biconnected based plane triangulations with n vertices and r vertices on the outer face

Input Integers n and r.

Output All biconnected based plane triangulations with n vertices and r vertices on the outer face.

Complexity $O(1)$ time per based plane triangulation with $O(n)$ space.

Comment A *based plane triangulation* is a plane triangulation with one designated edge on the outer face. The algorithm does not output entire solution but output the difference from the previous solution.

Reference [145]

2.43.6 Enumeration of all biconnected plane triangulations with n vertices and r vertices on the outer face

Input Integers n and r.

Output All biconnected based plane triangulations with n vertices and r vertices on the outer face.

Complexity $O(r^2 n)$ time per based plane triangulation with $O(n)$ space.

Reference [145]

2.43.7 Enumeration of all rooted triconnected plane triangulations with at most n vertices

Input An integer n.

Output All triconnected rooted plane triangulations with at most n vertices.

Complexity $O(1)$ time per tree and $O(n)$ space.

Comment A *rooted* plane triangulation is a plane triangulation with one designated vertex on the outer face.

Reference [170]

2.43.8 Enumeration of all rooted triconnected plane triangulations with exactly n vertices and r leaves

Input An integer n.

Output All triconnected rooted plane triangulations with exactly n vertices and r leaves.

Complexity $O(r)$ time per tree and $O(n)$ space.

Comment A *rooted* plane triangulation is a plane triangulation with one designated vertex on the outer face.

Reference [170]

2.43.9 Enumeration of all triconnected plane triangulations with exactly n vertices and r leaves

Input An integer n.

Output All triconnected plane triangulations with exactly n vertices and r leaves.

Complexity $O(r^n)$ time per triangulation and $O(n)$ space.

Reference [170]

2.43.10 Enumeration of all triangulations

Input A set S of n points in the general position in the plane.

Output all the triangulations whose vertex set is S and edge set includes the convex hull of S.

Complexity $O(\log \log n)$ time per triangulation and linear space.

Comment Whether there is the algorithm that outputs all triangulations in constant time delay?

Reference [20]

2.43.11 Enumeration of all biconnected plane triangulations with n vertices and r vertices on the outer faces

Input Integers n and r.

Output All biconnected plane triangulations with n vertices and r vertices on the outer faces.

Complexity $O(rn)$ time per triangulation and $O(n)$ space.

Reference [176]

2.43.12 Enumeration of all triangulations of a triconnected plane graph of n vertices

Input A triconnected planar graph G with n vertices.

Output All triangulations of G.

Complexity $O(1)$ time per triangulation and $O(n)$ space.

Reference [185]

2.44 Vertex cover

2.44.1 Enumeration of all minimal vertex covers in a graph

Input A graph $G = (V, E)$.

Output All minimal vertex covers of size up to k in G.

Complexity $O^*(1.6181^k)$ total time.

Comment This algorithm also lists some non-minimal vertex covers. This algorithm uses compact representation technique.

Reference [74]

2.44.2 Enumeration of all minimal vertex covers of size at most k in a graph

Input A graph $G = (V, E)$.

Output All minimal vertex covers of size at most k in G.

Complexity $O(|E| + k^2 2^k)$ total time.

Reference [52]

3 Hypergraph

3.1 Acyclic subhypergraph

3.1.1 Enumeration of all maximal α-acyclic subhypergraphs in a hypergraph

Input A hypergraph $H = (V, \mathcal{E})$.

Output All maximal α-acyclic subhypergraphs in H.

Complexity $O(|\mathcal{E}|^2(|V| + |\mathcal{E}|))$ delay and $O(|\mathcal{E}|)$space.

Comment The name of their algorithm is `GenMAS`. This algorithm uses the algorithm `FindMAS` that outputs a maximal α-acyclic subhypergraph.

Reference [51]

3.1.2 Enumeration of all Berge acyclic subhypergraphs in a hypergraph

Input A hypergraph $\mathcal{H}$.

Output All Berge acyclic subhypergraphs in $\mathcal{H}$.

Complexity $O(rd\tau(m))$ time per subhypergraph.

Comment r and d are the rank and the degree of $\mathcal{H}$ and $\tau(m) = O((\log \log m)^2 / \log \log \log(m))$.

Reference [254]

3.2 Independent set

3.2.1 Enumeration of all maximal independent set of a hypergraph of bounded dimension

Input A hypergraph H of bounded dimension.

Output All maximal independent set of a hypergraph of bounded dimension.

Comment The proposed algorithm runs in parallel.

Reference [26]

3.2.2 Enumeration of all maximal independent sets in a c-conformal hypergraph

Input A c-conformal hypergraph $\mathcal{H} \in A(k, r)$, where $c \leq$ constant and $k + r \leq c$.

Output All maximal independent sets in $\mathcal{H}$.

Complexity Incremental polynomial time.

Comment $A(k, r)$ is the class of hyperedges with (k, r)-bounded intersections, i.e. in which the intersection of any k distinct hyperedges has size at most r.

Reference [28]

3.2.3 Enumeration of all maximal independent sets in a hypergraph of bounded intersections

Input A hypergraph $\mathcal{H} \in A(k, r)$, where $k + r \leq$ constant.

Output All maximal independent sets in $\mathcal{H}$.

Complexity Incremental polynomial time with polynomial space.

Comment $A(k, r)$ is the class of hyperedges with (k, r)-bounded intersections, i.e. in which the intersection of any k distinct hyperedges has size at most r.

Reference [28]

3.3 Transversal

3.3.1 Enumeration of all minimal transversal of a hypergraph

Input A hypergraph $\mathcal{H}$.

Output All minimal transversal of $\mathcal{H}$.

Complexity $(n+N)^{O(\log n)}$ total time with $O(n \log n)$ words.

Comment $n = \sum_{X \in \mathcal{H}} |X|$ and N is the number of solutions.

Reference [234]

3.3.2 Enumeration of all minimal transversal in a hypergraph

Input A hypergraph $\mathcal{H} \in A(k, r)$.

Output All minimal transversal in $\mathcal{H}$.

Complexity $O(n^{k+r+1} |\mathcal{H}^d|^{r+1})$ total time and $O(N^{r+1})$ total space.

Comment $A(k, r)$ is the class of hyperedges with (k, r)-bounded intersections, i.e. in which the intersection of any k distinct hyperedges has size at most r. Minimal transversals of hypergraphs in some restricted classes can be enumerating in polynomial delay and space.

Reference [28]

4 Matroid

4.1 Basis

4.1.1 Enumeration of all common bases in two matroids

Input Two matroids $M_1 = (E, \beta_1)$ and $M_2 = (E, \beta_2)$.

Output All $B \in \beta_1 \cap \beta_2$.

Complexity $O(|E|(|E|^2 + t)\lambda)$ total time and $O(|E|^2)$ space.

Comment λ is the number of common bases and t is time complexity of one pivot operation.

Reference [84]

4.1.2 Enumeration of all basis of a matroid

Input A matroid M on the ground set P with rank m.

Output All basis of M.

Complexity $O((m|P| + t(Piv))N)$ total time and space complexity independent of N.

Comment N is the number of solutions. $t(Piv)$ is the time necessary to do one pivot operation.

Reference [7]

4.2 Spanning

4.2.1 Enumeration of all minimal spanning and connected subsets in a matroid

Input A matroid M.

Output All minimal spanning and connected subsets in M.

Complexity Incremental quasi-polynomial time.

Comment $f(x)$ is a quasi-polynomial if $f(x) \in O(2^{polylog(n)})$.

Reference [125]

4.3 Subset

4.3.1 Enumeration of all maximal subset

Input A binary matroid M on ground set S and $B = \{b_1, b_2\} \subseteq S$.

Output All maximal subsets X of $A := S \setminus B$ that span neither b_1 nor b_2.

Complexity Incremental polynomial time.

Reference [127]

5 Order

5.1 Ideal

5.1.1 Enumeration of all k-cardinarity ideals of a w-element poset

Input An integer k and a poset P with w element, where $k \leq w$.

Output All k-cardinarity ideals of P.

Complexity $O(Nw^3)$ total time.

Comment N is the number of ideals.

Reference [262]

6 Other

6.1 Assignment

6.1.1 Enumeration of all assignment

Input An integer n and $n \times n$ cost matrix $C = (c_{ij})$.

Output All assignments that minimizes $\sum_{i=1}^{i=n} \sum_{j=1}^{j=n} c_{ij}x_{ij}$ subject to $\sum_{i=1}^{i=n} x_{ij} = 1$ $(j = 1, \ldots, n)$, $\sum_{j=1}^{j=n} x_{ij} = 1$ $(i = 1, \ldots, n)$, and $x_{ij} \geq 0$.

Reference [168]

6.2 Full disjunction

6.2.1 Enumeration of all full disjunction in an acyclic set

Input An acyclic set of relation R with N tuples.

Output All full disjunctions of R.

Complexity $O(N)$ delay.

Reference [49]

6.3 Matrix

6.3.1 Enumeration of all minimal sets of at most k rows the deletion of which leaves a PP matrix

Input A binary $n \times m$ matrix B and a positive integer k, where $n > 4k$.

Output All minimal sets of at most k rows the deletion of which leaves a PP matrix.

Complexity $O(3^k nm)$ time.

Comment A PP matrix is a perfect phylogeny matrix.

Reference [52]

6.4 Round-robin tournament score

6.4.1 Enumeration of all round-robin tournament scores of n players

Input An integer n.

Output All round-robin tournament scores of n players.

Reference [177]

7 Permutation

7.1 Arrangements

7.1.1 Enumeration of all arrangements with n marks

Input An integer n.

Output All arrangements with n marks.

Complexity $O(n!)$ total time.

Reference [258]

7.1.2 Enumeration of all arrangements with n marks

Input An integer n.

Output All arrangements with n marks.

Complexity $O(n!)$ total time.

Reference [112]

7.2 Ladder lottery

7.2.1 Enumeration of all optimal ladder lotteries

Input A permutation.

Output All optimal ladder lotteries with satisfying the permutation.

Complexity $O(1)$ time per solution on average, $O(n^2)$ space, and $O(n^2)$ time preprocessing.

Comment n is the length of the input permutation. We call a ladder lottery is an optimal when the number of horizontal lines in the ladder lottery is minimum. Ladder lotteries are also known as arrangements of pseudolines.

Reference [270]

7.2.2 Enumeration of all ladder lotteries with k bars

Input A permutation π and integer k.

Output All ladder lotteries of π with k bars.

Complexity $O(1)$ time per ladder lottery.

Comment Ladder lotteries are also known as *Amida kuji* in Japan. I'm looking for this paper.

Reference [266]

7.3 Set

7.3.1 Enumeration of all permutations of a set of elements

Input A set S.

Output All permutations of S

Comment He proposed the general algorithm for some combinatorial problems.

Reference [62]

7.3.2 Enumeration of all permutations of a set of elements

Input A set S.

Output All permutations of S

Reference [54]

8 SAT

8.1 Boolean CSP

8.1.1 Enumeration of all models of ϕ by non-decreasing weight

Input A Γ-formula ϕ.

Output All models of ϕ by non-decreasing weight.

Complexity If Γ is Horn or width-2 affine, there exists a polynomial delay algorithm.

Comment Otherwise, such an algorithm does not exist unless $P \neq NP$.

Reference [50]

9 Set

9.1 Bitstring

9.1.1 Enumeration of all bitstrings of length n that contains exactly k 1's

Input An even integer n and odd integer k.

Output All bitstrings of length n that contains exactly k 1's.

Complexity $O(1)$ amortized time per solution.

Reference [198]

9.2 Ideals

9.2.1 Enumeration of all ideals in a poset

Input A poset $\mathcal{P}$.

Output All ideals in $\surd$.

Complexity $O(1)$ delay.

Reference [136]

9.3 Partition

9.3.1 Enumeration of all paritions in natural order

Input An integer n.

Output All partitions of n in natural order

Reference [163]

9.3.2 Enumeration of all paritions with restriction

Input Integers k and n.

Output All partitions of n whose the smallest part is greater than or equal to k.

Reference [260]

9.3.3 Enumeration of all k-partitions of n

Input Integers k and n.

Output All k-partitions of n.

Reference [177]

9.3.4 Enumeration of all paritions of an integer

Input An integer n.

Output All paritions of n.

Complexity

Reference [73]

9.3.5 Enumeration of all partitions of a set

Input A set S.

Output All partitions of S.

Complexity $O(1)$ amortized time per solution.

Reference [221]

9.3.6 Enumeration of all partions of n into integers of size at most k

Input Integers n and k.

Output All partions of n into integers of size at most k.

Reference [212]

9.3.7 Enumeration of all partitions of a set into a fixed number of blocks

Input An integer k.

Output All partitions of a set into k blocks

Complexity $O(1)$ amortized time per solution.

Reference [202]

9.3.8 Enumeration of all partitions of an integer n

Input Three integers n, k, and σ.

Output All partitions $P_\sigma(n,k)$ of n into parts of size at most k in which parts are congruent to 1 modulo σ.

Complexity $O(N)$ total time. E.g., $P_3(11,8) = P_3(11,7) = \{\{7,4\},\{7,1,1,1,1\},\{4,4,1,1,1\},\{4,1,1,$

Comment N is the number of partitions.

Reference [192]

9.3.9 Enumeration of all partitions of an integer n

Input Two integers n and k.

Output All partitions $D(n,k)$ of n into distinct parts of size at most k.

Complexity $O(N)$ total time. E.g., $D(10,5) = \{\{5,4,1\},\{5,3,2\},\{4,3,2,1\}\}$ and $D(11,4) = \emptyset$.

Comment N is the number of partitions.

Reference [192]

9.3.10 Enumeration of all partition of $\{1,\ldots,n\}$ into k non-empty subsets

Input An integer k.

Output All partition of $\{1,\ldots,n\}$ into k non-empty subsets.

Complexity $O(1)$ time per solution.

Comment The number of such partitions is known as the Stirling number of the second kind.

Reference [123]

9.3.11 Enumeration of all integer partitions in (anti-)lexicographical order

Input An integer n.

Output All integer partitions of n.

Complexity $O(1)$ time per solution on average.

Reference [229]

10 String

10.1 Binary string

10.1.1 Enumeration of all binary string with fixed number ones

Input Two integers n and k, where $n \geq k$.

Output All binary string with length n and k ones.

Complexity $O(1)$ time per binary string.

Reference [24]

10.2 Bracelet

10.2.1 Enumeration of all k-ary bracelets

Input n: a length of a bracelet, k: a number of alphabet size.

Output All k-ary bracelets.

Complexity $O(1)$ amortized per output and $O(n)$ space.

Comment A bracelet is the lexicographically smallest element of an equivalence class of k-ary strings under string rotation and reversal.

Reference [215]

10.3 Lyndon word

10.3.1 Enumeration of all k-ary Lyndon brackets of length n

Input Two integers k and n.

Output All k-ary Lyndon brackets of length n.

Complexity $O(n)$ time per solution.

Reference [217]

10.4 Necklace

10.4.1 Enumeration of all k-color necklaces with n beads

Input Integers k and n.

Output All k-color necklaces with n beads

Reference [81]

10.4.2 Enumeration of all necklaces of length n with two colors

Input An integer n.

Output All necklaces of length n with two colors.

Reference [80]

10.4.3 Enumeration of all k-ary necklaces

Input n: a length of a necklace, k: a number of alphabet size.

Output All k-ary necklaces with length n.

Complexity $O(1)$ amortized per output and $O(n)$ space.

Comment A k-ary necklace is an equivalence class of k-ary strings under rotation.

Reference [204]

10.4.4 Enumeration of all n-bit necklaces with fixed density d

Input Two integer n and d.

Output All n-bit necklaces with fixed density d.

Complexity $O(nN)$ total time and $O(n)$ space.

Comment N is the number of solutions. A density of a n-bit necklace T is d if T has d ones.

Reference [251]

10.4.5 Enumeration of all k-ary necklaces with fixed density

Input n: a length of a necklace, k: a number of alphabet size, d: a number of nonzero characters.

Output All k-ary necklaces with fixed density d.

Complexity $O(1)$ amortized per output and $O(n)$ space.

Comment The set of 3-ary necklace with 2-density and 4-length is $N_3(4,2) = \{0011, 0012, 0021, 0022, 0101, 0102, 0202\}$.

Reference [205]

10.4.6 Enumeration of all strings of some family

Complexity Constant amortized time per solution.

Comment This algorithm can list not only all necklaces but also all strings in other some family with CAT.

Reference [39]

10.4.7 Enumeration of all k-ary necklaces with fixed content of length n

Input Two integer k and n, and a content.

Output All k-ary necklaces with fixed content of length n.

Complexity Constant amortized time per solution.

Reference [214]

10.5 Parenthesis

10.5.1 Enumeration of all well-formed parenthesis with length $2n$

Input An integer n.

Output All well-formed parenthesis with length $2n$ in lexicographical ordering.

Reference [69]

10.5.2 Enumeration of all well-formed parenthesis strings of size n

Input An integer n.

Output All well-formed parenthesis strings of size n.

Complexity $O(1)$ delay with $O(n)$ space, or $O(n)$ delay with $O(1)$ space.

Reference [249]

10.6 Substring

10.6.1 Enumeration of all k-ary strings of length n that have no substring equal to f

Input A k-ary string f with length m, and a positive integer n.

Output All k-ary strings of length n that have no substring equal to f.

Complexity $O(1)$ time per string.

Reference [206]

10.6.2 Enumeration of all circular k-ary strings of length n that have no substring equal to f

Input A k-ary string f with length m, and a positive integer n.

Output All circular k-ary strings of length n that have no substring equal to f.

Complexity $O(1)$ time per string.

Reference [206]

10.6.3 Enumeration of all k-ary necklaces of length n that have no substring equal to f

Input A k-ary string f with length m, and a positive integer n.

Output All k-ary necklaces of length n that have no substring equal to f.

Complexity $O(1)$ time per string.

Comment f is an aperiodic necklace.

Reference [206]

11 Survey

11.1 Enumeration

11.1.1 Enumeration of k-best enumeration

Comment This is the full version of the Springer Encyclopedia of Algorithms, 2014.

Reference [65]

11.2 Graph

11.2.1 Alglrotihms for enumeration of all cycles in a given graph

Input A graph G.

Output All cycles belonging to G.

Comment 26 cycle enumeration algorithms are introduced.

Reference [153]

11.2.2 Enumeration of clique enumeration

Comment In Sec.3.

Reference [184]

11.2.3 Enumeration of gray code algorithms

Reference [211]

11.3 Logic

11.3.1

Comment The author of this paper investigate the class of databases with constant delay the answers to a query.

Reference [220]

Acknowledgements

I am deeply grateful to Komei Fukuda, Andrea Marino, Yasuko Matsui, Valia Mitsou, Shin-ichi Nakano, Ryuhei Uehara, and Katsuhisa Yamanaka for enumerating.

References

[1] Vicente Acuña et al. "Telling Stories: Enumerating Maximal Directed Acyclic Graphs With a Constrained Set of Sources and Targets". In: *Theoretical Computer Science* 457.0 (Oct. 2012), pp. 1–9. DOI: 10.1016/j.tcs.2012.07.023.

[2] O. Aichholzer et al. "Gray Code Enumeration of Plane Straight-Line Graphs". In: *Graphs and Combinatorics* 23.5 (Oct. 2007), pp. 467–479. DOI: 10.1007/s00373-007-0750-z.

[3] E. A. Akkoyunlu. "The Enumeration of Maximal Cliques of Large Graphs". In: *SIAM Journal on Computing* 2.1 (Mar. 1973), pp. 1–6. DOI: 10.1137/0202001.

[4] N Alon, R. Yuster, and U. Zwick. "Finding and counting given length cycles". In: *Algorithmica* 17.3 (Mar. 1997), pp. 209–223. DOI: 10.1007/BF02523189.

[5] Roberto Aringhieri, Pierre Hansen, and Federico Malucelli. "Chemical trees enumeration algorithms". In: *4OR* 1.1 (Mar. 2003), pp. 67–83. DOI: 10.1007/s10288-002-0008-9.

[6] David Avis. "Generating Rooted Triangulations Without Repetitions". In: *Algorithmica* 16.6 (Dec. 1996), pp. 618–632. DOI: 10.1007/BF01944353.

[7] David Avis and Komei Fukuda. "Reverse search for enumeration". In: *Discrete Applied Mathematics* 65.1-3 (Mar. 1996), pp. 21–46. DOI: 10.1016/0166-218X(95)00026-N.

[8] JoséAugusto Azevedo et al. "An algorithm for the ranking of shortest paths". In: *European Journal of Operational Research* 69.1 (Aug. 1993), pp. 97–106. DOI: 10.1016/0377-2217(93)90095-5.

[9] Darko Babić and Ante Graovac. "Enumeration of acyclic walks in a graph". In: *Discrete Applied Mathematics* 45.2 (Aug. 1993), pp. 117–123. DOI: 10.1016/0166-218X(93)90055-S.

[10] V.V. Bapeswara Rao and V.G.K. Murti. "Enumeration of all circuits of a graph". In: *Proceedings of the IEEE* 57.4 (1969), pp. 700–701. DOI: 10.1109/PROC.1969.7032.

[11] Vinnakota Bapiraju and V.V.Bapeswara Rao. "Enumeration of binary trees". In: *Information Processing Letters* 51.3 (Aug. 1994), pp. 125–127. DOI: 10.1016/0020-0190(94)00083-2.

[12] Valmir C. Barbosa and Jayme L. Szwarcfiter. "Generating all the acyclic orientations of an undirected graph". In: *Information Processing Letters* 72.1-2 (Oct. 1999), pp. 71–74. DOI: 10.1016/S0020-0190(99)00120-9.

[13] D.Roelants van Baronaigien. "A loopless algorithm for generating binary tree sequences". In: *Information Processing Letters* 39.4 (Aug. 1991), pp. 189–194. DOI: 10.1016/0020-0190(91)90178-K.

[14] D.Roelants van Baronaigien and Frank Ruskey. "Generating t-ary trees in A-order". In: *Information Processing Letters* 27.4 (Apr. 1988), pp. 205–213. DOI: 10.1016/0020-0190(88)90027-0.

[15] Manu Basavaraju et al. "Maximal Induced Matchings in Triangle-Free Graphs". In: *WG 2014: the 40th International Workshop on Graph-Theoretic Concepts in Computer Science*. Ed. by Dieter Kratsch and Ioan Todinca. Vol. 8747. Lecture Notes in Computer Science. Nouan-le-Fuzelier, France: Springer International Publishing, June 2014, pp. 93–104. DOI: 10.1007/978-3-319-12340-0_8.

[16] Reinhard Bauer, M Krug, and D Wagner. "Enumerating and Generating Labeled k-degenerate Graphs". In: *ANALCO 2010: the 7th Workshop on Analytic Algorithmics and Combinatorics*. Ed. by Robert Sedgewick and Mordecai Golin. Austin, Texas, USA: Society for Industrial and Applied Mathematics, Jan. 2010, pp. 90–98. DOI: 10.1137/1.9781611973006.12.

[17] Richard Beigel. "Finding maximum independent sets in sparse and general graphs". In: *SODA 1999: the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*. Ed. by Robert E. Tarjan and Tandy Warnow. Omni Inner Harbor Hotel, Baltimore, Maryland, USA: So-

ciety for Industrial and Applied Mathematics, Jan. 1999, pp. 856–857.

[18] Sergey Bereg. "Enumerating pseudo-triangulations in the plane". In: *Computational Geometry* 30.3 (Mar. 2005), pp. 207–222. DOI: 10.1016/j.comgeo.2004.09.002.

[19] Anne Berry, Jean-Paul Bordat, and Olivier Cogis. "Generating All the Minimal Separators of a Graph". en. In: *International Journal of Foundations of Computer Science* 11.03 (Sept. 2000), pp. 397–403. DOI: 10.1142/S0129054100000211.

[20] Sergei Bespamyatnikh. "An efficient algorithm for enumeration of triangulations". In: *Computational Geometry* 23.3 (Nov. 2002), pp. 271–279. DOI: 10.1016/S0925-7721(02)00111-6.

[21] Terry Beyer and Sandra Mitchell Hedetniemi. "Constant Time Generation of Rooted Trees". In: *SIAM Journal on Computing* 9.4 (Nov. 1980), pp. 706–712. DOI: 10.1137/0209055.

[22] E Birmelé, Pierluigi Crescenzi, and Rui Ferreira. "Efficient bubble enumeration in directed graphs". In: *SPIRE 2012: the 19th International Symposium String Processing and Information Retrieval*. Ed. by Liliana Calderón-Benavides et al. Vol. 7608. Lecture Notes in Computer Science. Cartagena de Indias, Colombia: Springer Berlin Heidelberg, Oct. 2012, pp. 118–129. DOI: 10.1007/978-3-642-34109-0_13.

[23] Etienne Birmelé et al. "Optimal Listing of Cycles and st-Paths in Undirected Graphs". In: *SODA 2012: the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*. Ed. by Sanjeev Khanna. New Orleans, LA, USA, Jan. 2012, pp. 1884–1896. DOI: 10.1137/1.9781611973105.134. eprint: 1205.2766v2.

[24] James R. Bitner, Gideon Ehrlich, and Edward M. Reingold. "Efficient generation of the binary reflected gray code and its applications". In: *Communications of the ACM* 19.9 (Sept. 1976), pp. 517–521. DOI: 10.1145/360336.360343.

[25] Michele Borassi et al. "Telling stories fast: Via linear-time delay pitch enumeration". In: *SEA 2013: the 12th International Symposium on Experimental Algorithms*. Ed. by Vincenzo Bonifaci, Camil Demetrescu, and Alberto Marchetti-Spaccamela. Vol. 7933. Lecture Notes in Computer Science. Rome, Italy: Springer Berlin Heidelberg, June 2013, pp. 200–211. DOI: 10.1007/978-3-642-38527-8_19.

[26] E. Boros et al. “An Efficient Incremental Algorithm for Generating All Maximal Independent Sets in Hypergraphs of Bounded Dimension”. In: *Parallel Processing Letters* 10.04 (Dec. 2000), pp. 253–266. DOI: 10.1142/S0129626400000251.

[27] E Boros et al. “Enumerating Minimal Dicuts and Strongly Connected Subgraphs and Related Geometric Problems”. In: *ICCO 2004: the 10th International Conference on Integer Programming and Combinatorial Optimization*. Ed. by Daniel Bienstock and George Nemhauser. Vol. 3064. Lecture Notes in Computer Science. New York, NY, USA: Springer Berlin Heidelberg, June 2004, pp. 152–162. DOI: 10.1007/978-3-540-25960-2_12.

[28] Endre Boros and Khaled Elbassioni. “Generating maximal independent sets for hypergraphs with bounded edge-intersections”. In: *LATIN 2004: the 6th Latin American Symposium on Theoretical Informatics*. Ed. by Martín Farach-Colton. Vol. 2976. Lecture Notes in Computer Science. Aires, Argentina: Springer Berlin Heidelberg, Apr. 2004, pp. 488–498. DOI: 10.1007/978-3-540-24698-5_52.

[29] Endre Boros, Khaled Elbassioni, and Vladimir Gurvich. “Generating paths and cuts in multi-pole (di) graphs”. In: *MFCS 2004: the 29th International Symposium on Mathematical Foundations of Computer Science*. Ed. by Jií Fiala, Václav Koubek, and Jan Kratochvíl. Vol. 3153. Lecture Notes in Computer Science Di. Prague, Czech Republic: Springer Berlin Heidelberg, Aug. 2004, pp. 298–309. DOI: 10.1007/978-3-540-28629-5_21.

[30] Endre Boros, Khaled Elbassioni, and Vladimir Gurvich. “Transversal hypergraphs to perfect matchings in bipartite graphs: Characterization and generation algorithms”. In: *Journal of Graph Theory* 53.3 (2006), pp. 209–232. DOI: 10.1002/jgt.20180.

[31] Endre Boros et al. “Generating Minimal k-Vertex Connected Spanning Subgraphs”. In: *COCOON 2007: the 13th Annual International Conference on Computing and Combinatorics*. Ed. by Guohui Lin. Vol. 4598. Lecture Notes in Computer Science. Banff, Canada: Springer Berlin Heidelberg, July 2007, pp. 222–231. DOI: 10.1007/978-3-540-73545-8_23.

[32] Robert Bowen. “The generation of minimal triangle graphs”. In: *Mathematics of Computation* 21.98 (May 1967), pp. 248–248. DOI: 10.1090/S0025-5718-1967-0223276-1.

[33] Robert Bowen and Stephen Fisk. “Generations of triangulations of the sphere”. In: *Mathematics of Computation* 21.98 (May 1967), pp. 250–250. DOI: 10.1090/S0025-5718-1967-0223277-3.

[34] Gunnar Brinkmann, Jan Goedgebeur, and BD McKay. "Generation of cubic graphs". In: *Discrete Mathematics and Theoretical Computer Science* 13.2 (2011), pp. 69–80.

[35] Coen Bron and Joep Kerbosch. "Algorithm 457: finding all cliques of an undirected graph". In: *Communications of the ACM* 16.9 (Sept. 1973), pp. 575–577. DOI: 10.1145/362342.362367. eprint: citation.cfm?doid=362342.362367

[36] Hervé Brönnimann et al. "Counting and Enumerating Pointed Pseudotriangulations With the Greedy Flip Algorithm". In: *SIAM Journal on Computing* 36.3 (Jan. 2006), pp. 721–739. DOI: 10.1137/050631008.

[37] Sr Bulo, Andrea Torsello, and Marcello Pelillo. "A continuous-based approach for partial clique enumeration". In: *GbRPR 2007: the 6th IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition*. Ed. by Francisco Escolano and Mario Vento. Vol. 4538. Lecture Notes in Computer Science. Alicante, Spain: Springer Berlin Heidelberg, June 2007, pp. 61–70. DOI: 10.1007/978-3-540-72903-7_6.

[38] E. Rodney Canfield and S. Gill Williamson. "A loop-free algorithm for generating the linear extensions of a poset". In: *Order* 12.1 (1995), pp. 57–75. DOI: 10.1007/BF01108590.

[39] Kevin Cattell et al. "Fast Algorithms to Generate Necklaces, Unlabeled Necklaces, and Irreducible Polynomials Over GF(2)". In: *Journal of Algorithms* 37.2 (Nov. 2000), pp. 267–282. DOI: 10.1006/jagm.2000.1108.

[40] LS Chandran. "A Linear Time Algorithm for Enumerating All the Minimum and Minimal Separators of a Chordal Graph". In: *COCOON 2001: the 7th Annual International Conference Computing and Combinatorics*. Ed. by Jie Wang. Vol. 2108. Lecture Notes in Computer Science. Guilin, China: Springer Berlin Heidelberg, Aug. 2001, pp. 308–317. DOI: 10.1007/3-540-44679-6_34.

[41] L.S. Chandran et al. "Generating and Characterizing the Perfect Elimination Orderings of a Chordal Graph". In: *Theoretical Computer Science* 307.2 (Oct. 2003), pp. 303–317. DOI: 10.1016/S0304-3975(03)00221-4.

[42] Lijun Chang, Jeffrey Xu Yu, and Lu Qin. "Fast Maximal Cliques Enumeration in Sparse Graphs". English. In: *Algorithmica* 66.1 (Mar. 2013), pp. 173–186. DOI: 10.1007/s00453-012-9632-8.

[43] Y.H. Chang, J.S. Wang, and R.C.T. Lee. "Generating all maximal independent sets on trees in lexicographic order". In: *Information Sciences* 76.3-4 (Jan. 1994), pp. 279–296. DOI: 10.1016/0020-0255(94)90013-2.

[44] J.P. Char. “Master circuit matrix”. In: *Proceedings of the Institution of Electrical Engineers* 117.8 (1970), p. 1655. DOI: 10.1049/piee.1970.0292.

[45] Chandra R. Chegireddy and Horst W. Hamacher. “Algorithms for finding K-best perfect matchings”. In: *Discrete Applied Mathematics* 18.2 (1987), pp. 155–165. DOI: 10.1016/0166-218X(87)90017-5.

[46] James Cheng et al. “Fast algorithms for maximal clique enumeration with limited memory”. In: *KDD 2012: the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. Ed. by Qiang Yang. Beijing, China: ACM Press, Aug. 2012, p. 1240. DOI: 10.1145/2339530.2339724.

[47] Norishige Chiba and Takao Nishizeki. “Arboricity and Subgraph Listing Algorithms”. en. In: *SIAM Journal on Computing* 14.1 (Feb. 1985), pp. 210–223. DOI: 10.1137/0214017.

[48] Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. “Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties”. In: *Journal of Computer and System Sciences* 74.7 (Nov. 2008), pp. 1147–1159. DOI: 10.1016/j.jcss.2008.04.003. eprint: 0410039.

[49] Sara Cohen et al. “Full disjunctions: polynomial-delay iterators in action”. In: *VLDB 2006: the 32nd international conference on Very Large Data Bases*. Ed. by Umeshwar Dayal et al. COEX, Seoul, Korea: VLDB Endowment, Sept. 2006, pp. 739–750.

[50] Nadia Creignou, Frédéric Olive, and Johannes Schmidt. “Enumerating all solutions of a Boolean CSP by non-decreasing weight”. In: *SAT 2011: the 14th International Conference Theory and Applications of Satisfiability Testing*. Ed. by Karem A. Sakallah and Laurent Simon. Vol. 6695. Lecture Notes in Computer Science. Ann Arbor, MI, USA, Springer Berlin Heidelberg, June 2011, pp. 120–133. DOI: 10.1007/978-3-642-21581-0_11.

[51] Taishin Daigo and Kouichi Hirata. “On Generating All Maximal Acyclic Subhypergraphs with Polynomial Delay”. In: *SOFSEM 2009: the 35th Conference on Current Trends in Theory and Practice of Computer Science*. Ed. by Mogens Nielsen et al. Vol. 5404. Lecture Notes in Computer Science. Špindlerv Mlýn, Czech Republic: Springer Berlin Heidelberg, Jan. 2009, pp. 181–192. DOI: 10.1007/978-3-540-95891-8_19.

[52] Peter Damaschke. “Parameterized Enumeration, Transversals, and Imperfect Phylogeny Reconstruction”. In: *Theoretical Computer Science* 351.3 (Feb. 2006), pp. 337–350. DOI: 10.1016/j.tcs.2005.10.004.

[53] G. Danielson. “On finding the simple paths and circuits in a graph”. In: *IEEE Transactions on Circuit Theory* 15.3 (Sept. 1968), pp. 294–295. DOI: 10.1109/TCT.1968.1082837.

[54] Nachum Dershowitz. “A simplified loop-free algorithm for generating permutations”. In: *BIT* 15.2 (June 1975), pp. 158–164. DOI: 10.1007/BF01932689.

[55] Antoine Deza, K Fukuda, and Vera Rosta. “Wagner’s theorem and combinatorial enumeration of 3-polytopes”. In: *Research reports on information sciences. Ser. B, Operations research* 271.- (1993), pp. 1–5.

[56] Vânia M.F. Dias, Celina M.H. de Figueiredo, and Jayme L. Szwarcfiter. “Generating Bicliques of a Graph in Lexicographic Order”. In: *Theoretical Computer Science* 337.1-3 (June 2005), pp. 240–248. DOI: 10.1016/j.tcs.2005.01.014.

[57] Matthew T. Dickerson, R.L. Scot Drysdale, and Jörg-Rüdiger Sack. “Simple Algorithms for Enumerating Interpoint Distances and Finding K Nearest Neighbors”. en. In: *International Journal of Computational Geometry & Applications* 02.03 (Sept. 1992), pp. 221–239. DOI: 10.1142/S0218195992000147.

[58] Mitre C. Dourado, Rodolfo A. Oliveira, and Fábio Protti. “Algorithmic Aspects of Steiner Convexity and Enumeration of Steiner Trees”. In: *Annals of Operations Research* 223.1 (May 2014), pp. 155–171. DOI: 10.1007/s10479-014-1607-5.

[59] Nan Du et al. “A Parallel Algorithm for Enumerating All Maximal Cliques in Complex Network”. In: *ICDMW 2006: the 6th IEEE International Conference on Data Mining - Workshops*. Hong Kong, China: IEEE, Dec. 2006, pp. 320–324. DOI: 10.1109/ICDMW.2006.17.

[60] Adrian Dumitrescu et al. “Enumerating triangulation paths”. In: *Computational Geometry* 20.1-2 (Oct. 2001), pp. 3–12. DOI: 10.1016/S0925-7721(01)00031-1.

[61] Andrzej Ehrenfeucht, Lloyd D. Fosdick, and Leon J. Osterweil. *An algorithm for finding the elementary circuits of a directed graph*. Tech. rep. CU-CS-024-23. Dept. of Computer Sci., Univ. of Colorado, Boulder, 1973, pp. 1–54.

[62] Gideon Ehrlich. “Loopless Algorithms for Generating Permutations, Combinations, and Other Combinatorial Configurations”. In: *Journal of the ACM* 20.3 (July 1973), pp. 500–513. DOI: 10.1145/321765.321781.

[63] David Eppstein. “Finding the K Shortest Paths”. In: *SIAM Journal on Computing* 28.2 (Jan. 1998), pp. 652–673. DOI: 10.1137/S0097539795290477.

[64] David Eppstein. “Finding the k smallest spanning trees”. In: *BIT* 32.2 (June 1992), pp. 237–248. DOI: 10.1007/BF01994879.

[65] David Eppstein. *K-Best Enumeration*. Tech. rep. arXiv ID: 1412.5075. USA: Cornell University, Dec. 2014. eprint: 1412.5075.

[66] David Eppstein. “Small Maximal Independent Sets and Faster Exact Graph Coloring”. In: *Journal of Graph Algorithms and Applications* 7.2 (2003), pp. 131–140. DOI: 10.7155/jgaa.00064.

[67] David Eppstein, M Löffler, and Darren Strash. “Listing all maximal cliques in sparse graphs in near-optimal time”. In: *ISAAC 2010: the 21st International Symposium on Algorithms and Computation*. Ed. by Otfried Cheong, Kyung-Yong Chwa, and Kunsoo Park. Vol. 6506. Lecture Notes in Computer Science. Jeju Island, Korea: Springer Berlin Heidelberg, Dec. 2010, pp. 403–414. DOI: 10.1007/978-3-642-17517-6_36.

[68] David Eppstein et al. “Sparsification—A Technique for Speeding Up Dynamic Graph Algorithms”. In: *Journal of the ACM* 44.5 (Sept. 1997), pp. 669–696. DOI: 10.1145/265910.265914.

[69] M. C. Er. “A Note on Generating Well-Formed Parenthesis Strings Lexicographically”. In: *The Computer Journal* 26.3 (Mar. 1983), pp. 205–207. DOI: 10.1093/comjnl/26.3.205.

[70] M. C. Er. “Efficient Generation of K-Ary Trees in Natural Order”. In: *The Computer Journal* 35.3 (June 1992), pp. 306–308. DOI: 10.1093/comjnl/35.3.306.

[71] M. C. Er. “Enumerating Ordered Trees Lexicographically”. In: *The Computer Journal* 28.5 (May 1985), pp. 538–542. DOI: 10.1093/comjnl/28.5.538.

[72] M. C. Er. “Lexicographic Listing and Ranking of T-Ary Trees”. In: *The Computer Journal* 30.6 (Dec. 1987), pp. 569–572. DOI: 10.1093/comjnl/30.6.569.

[73] T. I. Fenner. “A binary tree representation and related algorithms for generating integer partitions”. In: *The Computer Journal* 23.4 (Apr. 1980), pp. 332–337. DOI: 10.1093/comjnl/23.4.332.

[74] Henning Fernau. "Edge dominating set: Efficient enumeration-based exact algorithms". In: *IWPEC 2006: the 2nd International Workshop on Parameterized and Exact Computation*. Ed. by Hans L. Bodlaender and Michael A. Langston. Vol. 4169. Lecture Notes in Computer Science. Zürich, Switzerland: Springer Berlin Heidelberg, Sept. 2006, pp. 142–153. DOI: 10.1007/11847250_13.

[75] Rui Ferreira, Roberto Grossi, and Romeo Rizzi. "Output-sensitive listing of bounded-size trees in undirected graphs". In: *ESA 2011: the 19th Annual European Symposium on Algorithms*. Ed. by Camil Demetrescu and Magnús M. Halldórsson. Vol. 6942. Lecture Notes in Computer Science. Saarbrücken, Germany: Springer Berlin Heidelberg, Sept. 2011, pp. 275–286. DOI: 10.1007/978-3-642-23719-5_24.

[76] Rui Ferreira et al. "Amortized $\tilde{\{}O\}(|V|)$ -Delay Algorithm for Listing Chordless Cycles in Undirected Graphs". In: *ESA 2014: the 22th Annual European Symposium on Algorithms*. Ed. by Andreas S. Schulz and Dorothea Wagner. Vol. 8737. Lecture Notes in Computer Science. Wroclaw, Poland: Springer Berlin Heidelberg, Sept. 2014, pp. 418–429. DOI: 10.1007/978-3-662-44777-2_35.

[77] M. Florian, S. Nguyen, and S. Pallottino. "A dual simplex algorithm for finding all shortest paths". In: *Networks* 11.4 (1981), pp. 367–378. DOI: 10.1002/net.3230110406.

[78] Fedor V. Fomin et al. "Combinatorial bounds via measure and conquer". In: *ACM Transactions on Algorithms* 5.1 (Nov. 2008), pp. 1–17. DOI: 10.1145/1435375.1435384.

[79] Greg N. Frederickson. "Data Structures for On-Line Updating of Minimum Spanning Trees, With Applications". In: *SIAM Journal on Computing* 14.4 (1985), pp. 781–798. DOI: 10.1137/0214055.

[80] Harold Fredricksen and Irving J. Kessler. "An algorithm for generating necklaces of beads in two colors". In: *Discrete Mathematics* 61.2-3 (Sept. 1986), pp. 181–188. DOI: 10.1016/0012-365X(86)90089-0.

[81] Harold Fredricksen and James Maiorana. "Necklaces of beads in k colors and k-ary de Bruijn sequences". In: *Discrete Mathematics* 23.3 (Jan. 1978), pp. 207–210. DOI: 10.1016/0012-365X(78)90002-X.

[82] K. Fukuda and T. Matsui. "Finding All the Perfect Matchings in Bipartite Graphs". In: *Applied Mathematics Letters* 7.1 (Jan. 1994), pp. 15–18. DOI: 10.1016/0893-9659(94)90045-0.

[83] Komei Fukuda and Tomomi Matsui. "Finding all minimum-cost perfect matchings in Bipartite graphs". In: *Networks* 22.5 (Aug. 1992), pp. 461–468. DOI: 10.1002/net.3230220504.

[84] Komei Fukuda and Makoto Namiki. "Finding all common bases in two matroids". In: *Discrete Applied Mathematics* 56.2-3 (Jan. 1995), pp. 231–243. DOI: 10.1016/0166-218X(94)00088-U.

[85] Komei Fukuda, Shigemasa Saito, and Akihisa Tamura. "Combinatorial face enumeration in arrangements and oriented matroids". In: *Discrete Applied Mathematics* 31.2 (Apr. 1991), pp. 141–149. DOI: 10.1016/0166-218X(91)90066-6.

[86] Harold N. Gabow. "Two Algorithms for Generating Weighted Spanning Trees in Order". In: *SIAM Journal on Computing* 6.1 (1977), pp. 139–150. DOI: 10.1137/0206011.

[87] Harold N. Gabow and Eugene W. Myers. "Finding All Spanning Trees of Directed and Undirected Graphs". In: *SIAM Journal on Computing* 7.3 (Aug. 1978), pp. 280–287. DOI: 10.1137/0207024.

[88] Harold N. Gabow et al. "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs". In: *Combinatorica* 6.2 (June 1986), pp. 109–122. DOI: 10.1007/BF02579168.

[89] Michael R. Garey and Robert E. Tarjan. "A linear-time algorithm for finding all feedback vertices". In: *Information Processing Letters* 7.6 (1978), pp. 274–276. DOI: 10.1016/0020-0190(78)90015-7.

[90] Alain Gély, Lhouari Nourine, and Bachir Sadi. "Enumeration aspects of maximal cliques and bicliques". In: *Discrete Applied Mathematics* 157.7 (Apr. 2009), pp. 1447–1459. DOI: 10.1016/j.dam.2008.10.010.

[91] Chen Gen-Huey and Hung Yung-Chen. "Algorithms for the Constrained Quickest Path Problem and the Enumeration of Quickest Paths". In: *Computers & Operations Research* 21.2 (Feb. 1994), pp. 113–118. DOI: 10.1016/0305-0548(94)90045-0.

[92] L. Gerhards and W. Lindenberg. "Clique detection for nondirected graphs: Two new algorithms". In: *Computing* 21.4 (Dec. 1979), pp. 295–322. DOI: 10.1007/BF02248731.

[93] E. N. Gilbert. "Gray Codes and Paths on the n-Cube". In: *Bell System Technical Journal* 37.3 (May 1958), pp. 815–826. DOI: 10.1002/j.1538-7305.1958.tb0388

[94] Leslie Ann Goldberg. “Polynomial space polynomial delay algorithms for listing families of graphs”. In: *STOC 1993: the 25th annual ACM symposium on Theory of computing*. Ed. by Rao Kosaraju, David Johnson, and Alok Aggarwal. San Diego, CA, USA: ACM Press, June 1993, pp. 218–225. DOI: 10.1145/167088.167160.

[95] PA Golovach and Pinar Heggernes. “An incremental polynomial time algorithm to enumerate all minimal edge dominating sets”. In: *ICALP 2013: the 40th International Colloquium on Automata, Languages, and Programming*. Ed. by Fedor V. Fomin et al. Vol. 7965. Lecture Notes in Computer Science. Riga, Latvia: Springer Berlin Heidelberg, July 2013, pp. 485–496. DOI: 10.1007/978-3-642-39206-1_41.

[96] Petr A. Golovach et al. “Enumerating minimal dominating sets in chordal bipartite graphs”. In: *Discrete Applied Mathematics* 199 (Jan. 2015), pp. 30–36. DOI: 10.1016/j.dam.2014.12.010.

[97] Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Sept. 1989. DOI: 10.978.0262515/528.

[98] S. Hakimi and D. Green. “Generation and Realization of Trees and k-Trees”. In: *IEEE Transactions on Circuit Theory* 11.2 (1964), pp. 247–255. DOI: 10.1109/TCT.1964.1082276.

[99] S.L. Hakimi. “On trees of a graph and their generation”. In: *Journal of the Franklin Institute* 272.5 (Nov. 1961), pp. 347–359. DOI: 10.1016/0016-0032(61)90036-9.

[100] Horst W. Hamacher, Jean-Claude Picard, and Maurice Queyranne. “On Finding the K Best Cuts in a Network”. In: *Operations Research Letters* 2.6 (Mar. 1984), pp. 303–305. DOI: 10.1016/0167-6377(84)90083-X.

[101] Ramesh Hariharan, Sanjiv Kapoor, and Vijay Kumar. “Faster enumeration of all spanning trees of a directed graph”. In: *WADS 1995: the 4th International Workshop on Algorithms and Data Structures*. Ed. by Selim G. Akl et al. Vol. 955. Lecture Notes in Computer Science. Kingston, Canada: Springer Berlin Heidelberg, Aug. 1995, pp. 428–439. DOI: 10.1007/3-540-60220-8_82.

[102] CJ Henry and Sheela Ramanna. “Maximal clique enumeration in Finding Near Neighbourhoods”. In: *Transactions on Rough Sets XVI*. Ed. by James F. Peters et al. Vol. 7736. Lecture Notes in Computer Science. Banff, Canada: Springer Berlin Heidelberg, Oct. 2013, pp. 103–124. DOI: 10.1007/978-3-642-36505-8_7.

[103] Teruo Hikita. "Listing and counting subtrees of equal size of a binary tree". In: *Information Processing Letters* 17.4 (Nov. 1983), pp. 225–229. DOI: 10.1016/0020-0190(83)90046-7.

[104] Chính T. Hoàng et al. "Finding and listing induced paths and cycles". In: *Discrete Applied Mathematics* 161.4-5 (Mar. 2013), pp. 633–641. DOI: 10.1016/j.dam.2012.01.024.

[105] Falk Hüffner et al. "Isolation Concepts for Clique Enumeration: Comparison and Computational Experiments". In: *Theoretical Computer Science* 410.52 (Dec. 2009), pp. 5384–5397. DOI: 10.1016/j.tcs.2009.05.008.

[106] Yusuke Ishida et al. "Improved Algorithms for Enumerating Tree-like Chemical Graphs with Given Path Frequency". In: *the 19th International Conference on Genome Informatics 2008*. Ed. by Jonathan Arthur and See-Kiong Ng. Vol. 21. Genome Informatics Series. Gold Coast, Queensland, Australia: Imperial College Press, Dec. 2008, pp. 53–64. DOI: 10.1142/9781848163324_0005.

[107] Hiro Ito and Kazuo Iwama. "Enumeration of isolated cliques and pseudo-cliques". In: *ACM Transactions on Algorithms* 5.4 (Oct. 2009), pp. 1–21. DOI: 10.1145/1597036.1597044.

[108] Hiro Ito, Kazuo Iwama, and Tsuyoshi Osumi. "Linear-time enumeration of isolated cliques". In: *ESA 2005: the 13th Annual European Symposium on Algorithms*. Ed. by Gerth Stø lting Brodal and Stefano Leonardi. Vol. 3669. Lecture Notes in Computer Science. Palma de Mallorca, Spain: Springer Berlin Heidelberg, Oct. 2005, pp. 119–130. DOI: 10.1007/11561071_13.

[109] R. Jayakumar, K. Thulasiraman, and M. Swamy. "Complexity of computation of a spanning tree enumeration algorithm". In: *IEEE Transactions on Circuits and Systems* 31.10 (Oct. 1984), pp. 853–860. DOI: 10.1109/TCS.1984.1085435.

[110] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. "On generating all maximal independent sets". In: *Information Processing Letters* 27.3 (Mar. 1988), pp. 119–123. DOI: 10.1016/0020-0190(88)90065-8.

[111] Donald B. Johnson. "Finding All the Elementary Circuits of a Directed Graph". In: *SIAM Journal on Computing* 4.1 (Mar. 1975), pp. 77–84. DOI: 10.1137/0204007.

[112] Selmer M. Johnson. "Generation of permutations by adjacent transposition". In: *Mathematics of Computation* 17.83 (Sept. 1963), pp. 282–282. DOI: 10.1090/S0025-5718-1963-0159764-2.

[113] H. C. Johnston. "Cliques of a graph-variations on the Bron-Kerbosch algorithm". In: *International Journal of Computer & Information Sciences* 5.3 (Sept. 1976), pp. 209–238. DOI: 10.1007/BF00991836.

[114] T. Kamae. "A Systematic Method of Finding All Directed Circuits and Enumerating All DIrected Paths". In: *IEEE Transactions on Circuit Theory* 14.2 (June 1967), pp. 166–171. DOI: 10.1109/TCT.1967.1082699.

[115] Arkady Kanevsky. "Finding all minimum-size separating vertex sets in a graph". In: *Networks* 23.6 (Sept. 1993), pp. 533–541. DOI: 10.1002/net.3230230604.

[116] Mamadou Moustapha Kanté et al. "On the Enumeration of Minimal Dominating Sets and Related Notions". en. In: *SIAM Journal on Discrete Mathematics* 28.4 (Nov. 2014), pp. 1916–1929. DOI: 10.1137/120862612.

[117] MamadouMoustapha MM Kanté et al. "On the Enumeration and Counting of Minimal Dominating sets in Interval and Permutation Graphs". In: *ISAAC 2013: the 24th International Symposium on Algorithms and Computation*. Ed. by Leizhen Cai, Siu-Wing Cheng, and Tak-Wah Lam. Vol. 8283. Lecture Notes in Computer Science. Hong Kong, China: Springer Berlin Heidelberg, Dec. 2013, pp. 339–349. DOI: 10.1007/978-3-642-45030-3_32.

[118] MM Kanté et al. "On the neighbourhood Helly of some graph classes and applications to the enumeration of minimal dominating sets". In: *ISAAC 2012: the 23rd International Symposium on Algorithms and Computation*. Ed. by Kun-Mao Chao, Tsan-sheng Hsu, and Der-Tsai Lee. Vol. 7676. Lecture Notes in Computer Science. Taipei, Taiwan: Springer Berlin Heidelberg, Dec. 2012, pp. 289–298. DOI: 10.1007/978-3-642-35261-4_32.

[119] Sanjiv Kapoor and H Ramesh. "Algorithms for generating all spanning trees of undirected, directed and weighted graphs". In: *WADS 1991: the 2nd Workshop on Algorithms and Data Structures*. Ed. by Frank Dehne, Jörg-Rüdiger Sack, and Nicola Santoro. Vol. 519. Lecture Notes in Computer Science. Ottawa, Canada: Springer Berlin Heidelberg, Aug. 1991, pp. 461–472. DOI: 10.1007/BFb0028284.

[120] Toshinobu Kashiwabara et al. "Generation of Maximum Independent Sets of a Bipartite Graph and Maximum Cliques of a Circular-Arc Graph". In: *Journal of Algorithms* 13.1 (Mar. 1992), pp. 161–174. DOI: 10.1016/0196-6774(92)90012-2.

[121] N. Katoh, T. Ibaraki, and H. Mine. "Minimum Spanning Trees". In: *SIAM Journal on Computing* 10.2 (1981), pp. 247–255. DOI: 10.1137/0210017.

[122] Naoki Katoh and Shin-ichi Tanigawa. "Fast enumeration algorithms for non-crossing geometric graphs". In: *Discrete and Computational Geometry* 42.3 (Apr. 2009), pp. 443–468. DOI: 10.1007/s00454-009-9164-4.

[123] Shin-ichiro Kawano and Shin-ichi Nakano. "Constant time generation of set partitions". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E88-A.4 (Apr. 2005), pp. 930–934. DOI: 10.1093/ietfec/e88-a.4.930.

[124] Shin-ichiro Kawano and Shin-ichi Nakano. "Generating all series-parallel graphs". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E88-A.5 (2005), pp. 1129–1135. DOI: 10.1093/ietfec/e88-a.5.1129.

[125] L Khachiyan, E Boros, and K Borys. "Enumerating spanning and connected subsets in graphs and matroids". In: *ESA 2006: the 14th Annual European Symposium on Algorithms*. Ed. by Yossi Azar and Thomas Erlebach. Vol. 4168. Lecture Notes in Computer Science. Zurich, Switzerland: Springer Berlin Heidelberg, Sept. 2006, pp. 444–455. DOI: 10.1007/11841036_41.

[126] L Khachiyan, E Boros, and K Borys. "Generating Cut Conjunctions and Bridge Avoiding Extensions in Graphs". In: *ISAAC 2005: the 16th International Symposium on Algorithms and Computation*. Ed. by Xiaotie Deng and Ding-Zhu Du. Vol. 3827. Lecture Notes in Computer Science. Sanya, Hainan, China: Springer Berlin Heidelberg, Dec. 2005, pp. 156–165. DOI: 10.1007/11602613_17.

[127] Leonid Khachiyan et al. "Generating cut conjunctions in graphs and related problems". In: *Algorithmica* 51.3 (2008), pp. 239–263. DOI: 10.1007/s00453-007-9111-9.

[128] Shuji Kijima et al. "On Listing, Sampling, and Counting the Chordal Graphs With Edge Constraints". In: *Theoretical Computer Science* 411.26-28 (June 2010), pp. 2591–2601. DOI: 10.1016/j.tcs.2010.03.024.

[129] Masashi Kiyomi, Shuji Kijima, and Takeaki Uno. "Listing chordal graphs and interval graphs". In: *WG 2006: the 32nd International Workshop on Graph-Theoretic Concepts in Computer Science*. Ed. by Fedor V. Fomin. Vol. 4271. Lecture Notes in Computer Science. Bergen, Norway: Springer Berlin Heidelberg, June 2006, pp. 68–77. DOI: 10.1007/11917496_7.

[130] Masashi Kiyomi and Uno Takeaki. "Generating Chordal Graphs Included in Given Graphs". In: *IEICE Transactions on Information and Systems* E89-D.2 (Feb. 2006), pp. 763–770. DOI: 10.1093/ietisy/e89-d.2.763.

[131] T Kloks and D Kratsch. "Finding all minimal separators of a graph". In: *STACS 1994: the 11th Annual Symposium on Theoretical Aspects of Computer Science*. Ed. by Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner. Vol. 775. Lecture Notes in Computer Science. France: Springer Berlin Heidelberg, Feb. 1994, pp. 759–768. DOI: 10.1007/3-540-57785-8_188.

[132] T. Kloks and D. Kratsch. "Listing All Minimal Separators of a Graph". In: *SIAM Journal on Computing* 27.3 (June 1998), pp. 605–613. DOI: 10.1137/S009753979427087X.

[133] Donald E Knuth. "Lexicographic permutations with restrictions". In: *Discrete Applied Mathematics* 1.1-2 (Sept. 1979), pp. 117–125. DOI: 10.1016/0166-218X(79)90018-0.

[134] Donald E. Knuth and Jayme L. Szwarcfiter. "A structured program to generate all topological sorting arrangements". In: *Information Processing Letters* 2.6 (Apr. 1974), pp. 153–157. DOI: 10.1016/0020-0190(74)90001-5.

[135] Ina Koch. "Enumerating All Connected Maximal Common Subgraphs in Two Graphs". In: *Theoretical Computer Science* 250.1-2 (Jan. 2001), pp. 1–30. DOI: 10.1016/S0304-3975(00)00286-3.

[136] Y. Koda and F. Ruskey. "A Gray Code for the Ideals of a Forest Poset". In: *Journal of Algorithms* 15.2 (Sept. 1993), pp. 324–340. DOI: 10.1006/jagm.1993.1044.

[137] James F. Korsh. "Loopless generation of k-ary tree sequences". In: *Information Processing Letters* 52.5 (Dec. 1994), pp. 243–247. DOI: 10.1016/0020-0190(94)00149-9.

[138] James F. Korsh and Seymour Lipschutz. "Shifts and loopless generation of k-ary trees". In: *Information Processing Letters* 65.5 (Mar. 1998), pp. 235–240. DOI: 10.1016/S0020-0190(97)00215-9.

[139] D. Kroft. "All paths through a maze". In: *Proceedings of the IEEE* 55.1 (Jan. 1967), pp. 88–90. DOI: 10.1109/PROC.1967.5389.

[140] Marcin Krzywkowski. "An algorithm for listing all minimal 2-dominating sets of a tree". In: *FAW-AAIM 2013: the 3rd Joint International Conference on Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*. Ed. by Michael Fellows, Xuehou Tan, and Binhai Zhu. Vol. 7924. Lecture Notes in Computer Science.

Dalian, China: Springer Berlin Heidelberg, June 2013, pp. 12–16. DOI: 10.1007/978-3-642-38756-2_4.

[141] Jean-Paul Laumond. "Enumeration of articulation pairs of a planar graph". In: *Information Processing Letters* 21.4 (Oct. 1985), pp. 173–179. DOI: 10.1016/0020-0190(85)90055-9.

[142] E. L. Lawler. "A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem". In: *Management Science* 18.7 (1972), pp. 401–405. DOI: 10.1287/mnsc.18.7.401.

[143] Joseph Y.-T. Leung. "Fast Algorithms for Generating All Maximal Independent Sets of Interval, Circular-Arc and Chordal Graphs". In: *Journal of Algorithms* 5.1 (Mar. 1984), pp. 22–35. DOI: 10.1016/0196-6774(84)90037-3.

[144] Zhang-Jian Li and Shin-ichi Nakano. "Generating Biconnected Plane Quadrangulations". In: *IEICE transactions on information and systems* E86-D.4 (2003), pp. 698–703. DOI: MY/ietisy/e86-d.4.698.

[145] Zhangjian Li and Shin-ichi Nakano. "Efficient Generation of Plane Triangulations without Repetitions". In: *ICALP 2001: the 28th International Colloquium on Automata, Languages and Programming*. Ed. by Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen. Vol. 2076. Lecture Notes in Computer Science. Crete, Greece: Springer Berlin Heidelberg, July 2001, pp. 433–443. DOI: 10.1007/3-540-48224-5_36.

[146] E. Loukakis and C. Tsouros. "A Depth First Search Algorithm to Generate the Family of Maximal Independent Sets of a Graph Lexicographically". In: *Computing* 27.4 (Dec. 1981), pp. 349–366. DOI: 10.1007/BF02277184.

[147] Joan M. Lucas. "The Rotation Graph of Binary Trees Is Hamiltonian". In: *Journal of Algorithms* 8.4 (Dec. 1987), pp. 503–535. DOI: 10.1016/0196-6774(87)90048-4.

[148] Joan M. Lucas, D.R. Vanbaronaigien, and Frank Ruskey. "On Rotations and the Generation of Binary Trees". In: *Journal of Algorithms* 15.3 (Nov. 1993), pp. 343–366. DOI: 10.1006/jagm.1993.1045.

[149] Kazuhisa Makino and Takeaki Uno. "New algorithms for enumerating all maximal cliques". In: *SWAT 2004: the 9th Scandinavian Workshop on Algorithm Algorithm Theory*. Ed. by Torben Hagerup and Jyrki Katajainen. Vol. 3111. Lecture Notes in Computer Science. Humlebæk, Denmark: Springer Berlin Heidelberg, July 2004, pp. 260–272. DOI: 10.1007/978-3-540-27810-8_23.

[150] Alberto Martelli. "A Gaussian Elimination Algorithm for the Enumeration of Cut Sets in a Graph". In: *Journal of the ACM* 23.1 (Jan. 1976), pp. 58–73. DOI: 10.1145/321921.321928.

[151] Ernesto de Queirós Vieira Martins. "An algorithm for ranking paths that may contain cycles". In: *European Journal of Operational Research* 18.1 (1984), pp. 123–130. DOI: 10.1016/0377-2217(84)90269-8.

[152] Tomonari Masada, Hiroshi Imai, and Keiko Imai. "Enumeration of regular triangulations". In: *SCG 1996: the 12th annual symposium on Computational geometry*. Ed. by Sue Whitesides. Philadelphia, PA, USA: ACM Press, May 1996, pp. 224–233. DOI: 10.1145/237218.237373.

[153] Prabhaker Mateti and Narsingh Deo. "On Algorithms for Enumerating All Circuits of a Graph". en. In: *SIAM Journal on Computing* 5.1 (Mar. 1976), pp. 90–99. DOI: 10.1137/0205007.

[154] T. Matsui. "A Flexible Algorithm for Generating All the Spanning Trees in Undirected Graphs". In: *Algorithmica* 18.4 (Aug. 1997), pp. 530–543. DOI: 10.1007/PL00009171.

[155] Tomomi Matsui, Akihisa Tamura, and Yoshiko Ikebe. "Algorithms for finding a Kth best valued assignment". In: *Discrete Applied Mathematics* 50.3 (1994), pp. 283–296. DOI: 10.1016/0166-218X(92)00175-L.

[156] Yasuko Matsui and Tomomi Matsui. *Enumeration Algorithm for the Edge Coloring Problem on Bipartite Graphs*. Vol. 1120. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1996, pp. 18–26. DOI: 10.1007/3-540-61576-8_69.

[157] Yasuko Matsui, Ryuhei Uehara, and Takeaki Uno. "Enumeration of Perfect Sequences of Chordal Graph". In: *ISAAC 2008: the 19th International Symposium on Algorithms and Computation*. Ed. by Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga. Vol. 5369. Lecture Notes in Computer Science. Gold Coast, Australia: Springer Berlin Heidelberg, Dec. 2008, pp. 859–870. DOI: 10.1007/978-3-540-92182-0_75.

[158] Yasuko Matsui, Ryuhei Uehara, and Takeaki Uno. "Enumeration of the Perfect Sequences of a Chordal Graph". In: *Theoretical Computer Science* 411.40-42 (Sept. 2010), pp. 3635–3641. DOI: 10.1016/j.tcs.2010.06.007.

[159] Yasuko Matsui and Takeaki Uno. "A simple and fast algorithm for enumerating all edge colorings of a bipartite graph". In: *Korea-Japan Joint Workshop on Algorithms and Computation*. KAIST,Taejon,Korea, Aug. 1996, pp. 81–87.

[160] W. Mayeda and S. Seshu. "Generation of Trees without Duplications". English. In: *IEEE Transactions on Circuit Theory* 12.2 (June 1965), pp. 181–185. DOI: 10.1109/TCT.1965.1082432.

[161] Ernst W Mayr and C Greg Plaxton. "On the spanning trees of weighted graphs". In: *Combinatorica* 12.4 (1992), pp. 433–447. DOI: 10.1007/BF01305236.

[162] F. Mazoit. "Listing all the minimal separators of a 3-connected planar graph". In: *Discrete Mathematics* 306.3 (Feb. 2006), pp. 372–380. DOI: 10.1016/j.disc.2005.12.007.

[163] J. K. S. McKay. "Algorithm 371: Partitions in natural order [A1]". In: *Communications of the ACM* 13.1 (Jan. 1970), p. 52. DOI: 10.1145/361953.361980.

[164] N. Megiddo et al. "An $O(n \log^2 n)$ Algorithm for the kth Longest Path in a Tree with Applications to Location Problems". In: *SIAM Journal on Computing* 10.2 (1981), pp. 328–337. DOI: 10.1137/0210023.

[165] George J. Minty. "A Simple Algorithm for Listing All the Trees of a Graph". In: *IEEE Transactions on Circuit Theory* 12.1 (1965), p. 120. DOI: 10.1109/TCT.1965.1082385.

[166] George J. Minty. "On maximal independent sets of vertices in claw-free graphs". In: *Journal of Combinatorial Theory, Series B* 28.3 (June 1980), pp. 284–304. DOI: 10.1016/0095-8956(80)90074-X.

[167] Gordon D. Mulligan and D. G. Corneil. "Corrections to Bierstone's Algorithm for Generating Cliques". In: *Journal of the ACM* 19.2 (Apr. 1972), pp. 244–247. DOI: 10.1145/321694.321698.

[168] K. G. Murty. "Letter to the Editor–An Algorithm for Ranking All the Assignments in Order of Increasing Cost". In: *Operations Research* 16.3 (1968), pp. 682–687. DOI: 10.1287/opre.16.3.682.

[169] Shin-ichi Nakano. "Efficient generation of plane trees". In: *Information Processing Letters* 84.3 (Nov. 2002), pp. 167–172. DOI: 10.1016/S0020-0190(02)00240-5.

[170] Shin-ichi Nakano. "Efficient Generation of Triconnected Plane Triangulations". In: *COCOON 2001: the 7th Annual International Conference on Computing and Combinatorics*. Ed. by Jie Wang. Vol. 2108. Lecture Notes in Computer Science. Guilin, China: Springer Berlin Heidelberg, Aug. 2001, pp. 131–141. DOI: 10.1007/3-540-44679-6_15.

[171] Shin-ichi Nakano. “Enumerating Floorplans with n Rooms”. In: *ISAAC 2001: the 12th International Symposium on Algorithms and Computation*. Ed. by Peter Eades and Tadao Takaoka. Vol. 2223. Lecture Notes in Computer Science. Christchurch, New Zealand, Springer Berlin Heidelberg, Dec. 2001, pp. 107–115. DOI: 10.1007/3-540-45678-3_10.

[172] Shin-ichi Nakano. “Listing All Trees with Specified Degree Sequence (Acceleration and Visualization of Computation for Enumeration Problems)”. In: 1644.- (2009), pp. 55–62.

[173] Shin-ichi Nakano and Takeaki Uno. “Constant TIme Generation of Trees with Specified Diameter”. In: *WG 2004: the 30th International Workshop on Graph-Theoretic Concepts in Computer Science*. Ed. by Juraj Hromkovič, Manfred Nagl, and Bernhard Westfechtel. Vol. 3353. Lecture Notes in Computer Science. Honnef, Germany: Springer Berlin Heidelberg, June 2004, pp. 33–45. DOI: 10.1007/978-3-540-30559-0_3.

[174] Shin-ichi Nakano and Takeaki Uno. *Efficient generation of rooted trees*. Tech. rep. Tech. Rep. NII-2003-005E. Japan: National Institute for Infomatics, 2003.

[175] Shin-ichi Nakano and Takeaki Uno. “Generating colored trees”. In: *WG 2005: the 31st International Workshop on Graph-Theoretic Concepts in Computer Science*. Ed. by Dieter Kratsch. Vol. 3787. Lecture Notes in Computer Science. Metz, France: Springer Berlin Heidelberg, June 2005, pp. 249–260. DOI: 10.1007/11604686_22.

[176] Shin-ichi Nakano and Takeaki Uno. “More Efficient Generation of Plane Triangulations”. In: *the11th International Symposium on Graph Drawing - GD 2004*. Ed. by Giuseppe Liotta. Vol. 2912. Lecture Notes in Computer Science. Perugia, Italy: Springer Berlin Heidelberg, Sept. 2004, pp. 273–282. DOI: 10.1007/978-3-540-24595-7_25.

[177] T.V Narayana, R.M Mathsen, and J Sarangi. “An algorithm for generating partitions and its applications”. In: *Journal of Combinatorial Theory, Series A* 11.1 (July 1971), pp. 54–61. DOI: 10.1016/0097-3165(71)90007-0.

[178] Zeev Nutov. “Listing minimal edge-covers of intersecting families with applications to connectivity problems”. In: *Discrete Applied Mathematics* 157.1 (Jan. 2009), pp. 112–117. DOI: 10.1016/j.dam.2008.04.026.

[179] Y Okamoto, T Uno, and R Uehara. “Linear time counting algorithms for independent sets in chordal graphs”. In: *WG 2005: the 31st International Workshop on Graph-Theoretic Concepts in Computer Science*. Ed. by Dieter Kratsch. Vol. 3787. Lecture Notes in

Computer Science. Metz, France: Springer Berlin Heidelberg, June 2005, pp. 433–444. DOI: 10.1007/11604686_38.

[180] Yoshio Okamoto, Takeaki Uno, and Ryuhei Uehara. "Counting the number of independent sets in chordal graphs". In: *Journal of Discrete Algorithms* 6.2 (June 2008), pp. 229–242. DOI: 10.1016/j.jda.2006.07.006.

[181] Akimitsu Ono and Shin-ichi Nakano. "Constant Time Generation of Linear Extensions". In: *FCT 2005: the 15th International Symposium on Fundamentals of Computation Theory*. Ed. by Maciej Liśkiewicz and Rüdiger Reischuk. Vol. 3623. Lecture Notes in Computer Science. Lübeck, Germany: Springer Berlin Heidelberg, Aug. 2005, pp. 445–453. DOI: 10.1007/11537311_39.

[182] J. Pallo. "Generating Trees With N Nodes and M Leaves". In: *International Journal of Computer Mathematics* 21.2 (Jan. 1987), pp. 133–144. DOI: 10.1080/00207168708803562.

[183] Long Pan and Eunice E. Santos. "An anytime-anywhere approach for maximal clique enumeration in social network analysis". In: *the IEEE International Conference on Systems, Man and Cybernetics - SMC 2008*. Singapore: IEEE, Oct. 2008, pp. 3529–3535. DOI: 10.1109/ICSMC.2008.4811845.

[184] Panos M. Pardalos and Jue Xue. "The maximum clique problem". In: *Journal of Global Optimization* 4.3 (Apr. 1994), pp. 301–328. DOI: 10.1007/BF01098364.

[185] Mohammad Tanvir Parvez, Md Saidur Rahman, and Shin-ichi Nakano. "Generating all triangulations of plane graphs : Extended abstract". In: *WALCOM 2009: the 3rd International Workshop on Algorithms and Computation*. Ed. by Sandip Das and Ryuhei Uehara. Vol. 5431. Lecture Notes in Computer Science. Kolkata, India: Springer Berlin Heidelberg, Feb. 2009, pp. 151–164. DOI: 10.1007/978-3-642-00202-1_14.

[186] J. Ponstein. "Self-Avoiding Paths and the Adjacency Matrix of a Graph". In: *SIAM Journal on Applied Mathematics* 14.3 (Mar. 1966), pp. 600–609. DOI: 10.1137/0114051.

[187] Andrzej Proskurowski. "On the Generation of Binary Trees". In: *Journal of the ACM* 27.1 (Jan. 1980), pp. 1–2. DOI: 10.1145/322169.322170.

[188] Andrzej Proskurowski and Frank Ruskey. "Binary Tree Gray Codes". In: *Journal of Algorithms* 6.2 (June 1985), pp. 225–238. DOI: 10.1016/0196-6774(85)90040-9.

[189] J. S. Provan and D. R. Shier. "A paradigm for listing (s, t)-cuts in graphs". In: *Algorithmica* 15.4 (Apr. 1996), pp. 351–372. DOI: 10.1007/BF01961544.

[190] Gara Pruesse and Frank Ruskey. “Generating Linear Extensions Fast”. In: *SIAM Journal on Computing* 23.2 (1994), pp. 373–386. DOI: 10.1137/S0097539791202647.

[191] Gara Pruesse and Frank Ruskey. “Generating the Linear Extensions of Certain Posets by Transpositions”. en. In: *SIAM Journal on Discrete Mathematics* 4.3 (Aug. 1991), pp. 413–422. DOI: 10.1137/0404037.

[192] David Rasmussen, Carla D Savage, and Douglas B West. “Gray code enumeration of families of integer partitions”. In: *Journal of Combinatorial Theory, Series A* 70.2 (May 1995), pp. 201–229. DOI: 10.1016/0097-3165(95)90090-X.

[193] Ronald C. Read and Robert Endre Tarjan. “Bounds on backtrack algorithms for listing cycles, paths, and spanning trees”. In: *Networks* 5.3 (1975), pp. 237–252.

[194] J.B. Rosen, S.-Z. Sun, and G.-L. Xue. “Algorithms for the Quickest Path Problem and the Enumeration of Quickest Paths”. In: *Computers & Operations Research* 18.6 (Jan. 1991), pp. 579–584. DOI: 10.1016/0305-0548(91)90063-W.

[195] Doron Rotem and Julia Urrutia. “Finding Maximum Cliques In Circle Graphs”. In: *Networks* 11.3 (1981), pp. 269–278. DOI: 10.1002/net.3230110305.

[196] Doron Rotem and Y. L. Varol. “Generation of Binary Trees from Ballot Sequences”. In: *Journal of the ACM* 25.3 (July 1978), pp. 396–404. DOI: 10.1145/322077.322082.

[197] F. Ruskey and T. C. Hu. “Generating Binary Trees Lexicographically”. en. In: *SIAM Journal on Computing* 6.4 (Dec. 1977), pp. 745–758. DOI: 10.1137/0206055.

[198] Frank Ruskey. “Adjacent Interchange Generation of Combinations”. In: *Journal of Algorithms* 9.2 (June 1988), pp. 162–180. DOI: 10.1016/0196-6774(88)90036-3.

[199] Frank Ruskey. “Generating linear extensions of posets by transpositions”. In: *Journal of Combinatorial Theory, Series B* 54.1 (Jan. 1992), pp. 77–101. DOI: 10.1016/0095-8956(92)90067-8.

[200] Frank Ruskey. “Generating T -Ary Trees Lexicographically”. en. In: *SIAM Journal on Computing* 7.4 (Nov. 1978), pp. 424–439. DOI: 10.1137/0207034.

[201] Frank Ruskey. “Listing and Counting Subtrees of a Tree”. In: *SIAM Journal on Computing* 10.1 (Feb. 1981), pp. 141–150. DOI: 10.1137/0210011.

[202] Frank Ruskey. “Simple combinatorial Gray codes constructed by reversing sublists”. In: *ISAAC 1993: the 4th International Symposium on Algorithms and Computation*. Ed. by K. W. Ng et al. Vol. 762. Lecture Notes in Computer Science. Hong Kong, China: Springer Berlin Heidelberg, Dec. 1993, pp. 201–208. DOI: 10.1007/3-540-57568-5_250.

[203] Frank Ruskey and Andrzej Proskurowski. “Generating Binary Trees by Transpositions”. In: *Journal of Algorithms* 11.1 (Mar. 1990), pp. 68–84. DOI: 10.1016/0196-6774(90)90030-I.

[204] Frank Ruskey, Carla Savage, and Terry Min Yih Wang. “Generating Necklaces”. In: *Journal of Algorithms* 13.3 (Sept. 1992), pp. 414–430. DOI: 10.1016/0196-6774(92)90047-G.

[205] Frank Ruskey and Joe Sawada. “An Efficient Algorithm for Generating Necklaces With Fixed Density”. In: *SIAM Journal on Computing* 29.2 (Jan. 1999), pp. 671–684. DOI: 10.1137/S0097539798344112.

[206] Frank Ruskey and Joe Sawada. “Generating necklaces and strings with forbidden substrings”. In: *COCOON 2000: the 6th Annual International Conference on Computing and Combinatorics*. Ed. by Ding-Zhu Du et al. Vol. 1858. Lecture Notes in Computer Science. Sydney, Australia: Springer Berlin Heidelberg, July 2000, pp. 330–339. DOI: 10.1007/3-540-44968-X_33.

[207] Gustavo Sacomoto, Vincent Lacroix, and Marie-france Sagot. “A Polynomial Delay Algorithm for the Enumeration of Bubbles with Length Constraints in Directed Graphs and Its Application to the Detection of Alternative Splicing in RNA-seq Data”. In: *WABI 2013: the 13th International Workshop on Algorithms in Bioinformatics*. Ed. by Aaron Darling and Jens Stoye. Vol. 8126. Lecture Notes in Computer Science. Sophia Antipolis, France: Springer Berlin Heidelberg, Sept. 2013, pp. 99–111. DOI: 10.1007/978-3-642-40453-5_9.

[208] Toshiki Saitoh et al. “Random generation and enumeration of bipartite permutation graphs”. In: *Journal of Discrete Algorithms* 10.1 (Jan. 2012), pp. 84–97. DOI: 10.1016/j.jda.2011.11.001.

[209] Toshiki Saitoh et al. “Random generation and enumeration of proper interval graphs”. In: *IEICE Transactions on Information and Systems* E93-D.7 (July 2010), pp. 1816–1823. DOI: 10.1587/transinf.E93.D.1816.

[210] Yasufumi Saruwatari and Tomomi Matsui. “A Note on K-Best Solutions to the Chinese Postman Problem”. In: *SIAM Journal on Optimization* 3.4 (1993), pp. 726–733. DOI: 10.1137/0803037.

[211] Carla Savage. “A Survey of Combinatorial Gray Codes”. In: *SIAM Review* 39.4 (Jan. 1997), pp. 605–629. DOI: 10.1137/S0036144595295272.

[212] Carla D Savage. “Gray Code Sequences of Partitions”. In: *Journal of Algorithms* 10.4 (Dec. 1989), pp. 577–595. DOI: 10.1016/0196-6774(89)90007-2.

[213] Joe Sawada. “A Fast Algorithm for Generating Nonisomorphic Chord Diagrams”. In: *SIAM Journal on Discrete Mathematics* 15.4 (Jan. 2002), pp. 546–561. DOI: 10.1137/S0895480100377970.

[214] Joe Sawada. “A Fast Algorithm to Generate Necklaces With Fixed Content”. In: *Theoretical Computer Science* 301.1-3 (May 2003), pp. 477–489. DOI: 10.1016/S0304-3975(03)00049-5.

[215] Joe Sawada. “Generating Bracelets in Constant Amortized Time”. In: *SIAM Journal on Computing* 31.1 (Jan. 2001), pp. 259–268. DOI: 10.1137/S0097539700377037.

[216] Joe Sawada. “Generating rooted and free plane trees”. In: *ACM Transactions on Algorithms* 2.1 (Jan. 2006), pp. 1–13. DOI: 10.1145/1125994.1125995.

[217] Joe Sawada and Frank Ruskey. “Generating Lyndon Brackets.” In: *Journal of Algorithms* 46.1 (Jan. 2003), pp. 21–26. DOI: 10.1016/S0196-6774(02)00286-9.

[218] Matthew C. Schmidt et al. “A Scalable, Parallel Algorithm for Maximal Clique Enumeration”. In: *Journal of Parallel and Distributed Computing* 69.4 (Apr. 2009), pp. 417–428. DOI: 10.1016/j.jpdc.2009.01.003.

[219] Benno Schwikowski and Ewald Speckenmeyer. “On enumerating all minimal solutions of feedback problems”. In: *Discrete Applied Mathematics* 117.1-3 (Mar. 2002), pp. 253–265. DOI: 10.1016/S0166-218X(00)00339-5.

[220] Luc Segoufin. “Enumerating with constant delay the answers to a query”. In: *ICDT 2013: the 16th International Conference on Database Theory*. Ed. by Wang Chiew Tan. Genoa, Italy: ACM Press, Mar. 2013, pp. 10–20. DOI: 10.1145/2448496.2448498.

[221] Ichiro Semba. “An Efficient Algorithm for Generating all Partitions of the Set{1, 2, ..., n}”. In: *Journal of information processing* 7.1 (1984), pp. 41–42.

[222] Andry Setiawan and Shin-ichi Nakano. “Listing all st-orientations”. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E94-A.10 (2011), pp. 1965–1970. DOI: 10.1587/transfun.E94.A.1965.

[223] Hong Shen and Weifa Liang. "Efficient Enumeration of All Minimal Separators in a Graph". In: *Theoretical Computer Science* 180.1-2 (June 1997), pp. 169–180. DOI: 10.1016/S0304-3975(97)83809-1.

[224] D. R. Shier. "On algorithms for finding the k shortest paths in a network". In: *Networks* 9.3 (1979), pp. 195–214. DOI: 10.1002/net.3230090303.

[225] Akiyoshi Shioura and Akihisa Tamura. "Efficiently Scanning all Spanning Trees of an Undirected Graph". In: *Journal of the Operations Research Society of Japan* 38.3 (1995), pp. 331–344. DOI: 10.1.1.30.4555.

[226] Akiyoshi Shioura, Akihisa Tamura, and Takeaki Uno. "An Optimal Algorithm for Scanning All Spanning Trees of Undirected Graphs". en. In: *SIAM Journal on Computing* 26.3 (June 1997), pp. 678–692. DOI: 10.1137/S0097539794270881.

[227] Kenneth Sörensen and Gerrit K. Janssens. "An algorithm to generate all spanning trees of a graph in order of increasing cost". In: *Pesquisa Operacional* 25.2 (Aug. 2005), pp. 219–229. DOI: 10.1590/S0101-74382005000200004.

[228] Volker Stix. "Finding All Maximal Cliques in Dynamic Graphs". In: *Computational Optimization and Applications* 27.2 (Feb. 2004), pp. 173–186. DOI: 10.1023/B:COAP.0000008651.28952.b6.

[229] Ivan Stojmenović and Antoine Zoghbi. "Fast Algorithms for Genegrating Integer Partitions". In: *International Journal of Computer Mathematics* 70.2 (Mar. 2007), pp. 319–332. DOI: 10.1080/00207169808804755.

[230] Maciej M. Sysło. "An Efficient Cycle Vector Space Algorithm for Listing All Cycles of a Planar Graph". In: *SIAM Journal on Computing* 10.4 (Nov. 1981), pp. 797–808. DOI: 10.1137/0210062.

[231] JL Szwarcfiter. "Generating all forest extensions of a partially ordered set". In: *CIAC 2003: the 5th Italian Conference on Algorithms and Complexity*. Ed. by Rossella Petreschi, Giuseppe Persiano, and Riccardo Silvestri. Vol. 2653. Lecture Notes in Computer Science. Rome, Italy: Springer Berlin Heidelberg, May 2003, pp. 132–139. DOI: 10.1007/3-540-44849-7_19.

[232] Masahiro Takagi and Shin-ichi Nakano. "Listing all rectangular drawings with certain properties". In: *Systems and Computers in Japan* 35.4 (2004), pp. 1–8. DOI: 10.1002/scj.10563.

[233] Fumihiko Takeuchi and Hiroshi Imai. "Enumerating triangulations for products of two simplices and for arbitrary configurations of points". In: *COCOON 1997: the 3rd Annual International Conference on Computing and Combinatorics*. Ed. by Tao Jiang and D. T. Lee. Vol. 1276. Lecture Notes in Computer Science. Shanghai, China: Springer Berlin Heidelberg, Aug. 1997, pp. 470–481. DOI: 10.1007/BFb0045114.

[234] Hisao Tamaki. "Space-efficient enumeration of minimal transversals of a hypergraph (in Japanese)". In: *SIGAL reports*. 103. Information Processing Society of Japan, Nov. 2000, pp. 29–36. DOI: MY/Tamaki_2000_110002812417.

[235] Robert Tarjan. "Enumeration of the Elementary Circuits of a Directed Graph". In: *SIAM Journal on Computing* 2.3 (Sept. 1973), pp. 211–216. DOI: 10.1137/0202017.

[236] James C. Tiernan. "An efficient search algorithm to find the elementary circuits of a graph". In: *Communications of the ACM* 13.12 (Dec. 1970), pp. 722–726. DOI: 10.1145/362814.362819.

[237] Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. "The Worst-Case Time Complexity for Generating All Maximal Cliques and Computational Experiments". In: *Theoretical Computer Science* 363.1 (Oct. 2006), pp. 28–42. DOI: 10.1016/j.tcs.2006.06.015.

[238] Anthony E. Trojanowski. "Ranking and Listing Algorithms for k -Ary Trees". In: *SIAM Journal on Computing* 7.4 (Nov. 1978), pp. 492–509. DOI: 10.1137/0207039.

[239] S. Tsukiyama et al. "An Algorithm to Enumerate All Cutsets of a Graph in Linear Time Per Cutset". In: *Journal of the ACM* 27.4 (Oct. 1980), pp. 619–632. DOI: 10.1145/322217.322220.

[240] Shuji Tsukiyama et al. "A New Algorithm for Generating All the Maximal Independent Sets". In: *SIAM Journal on Computing* 6.3 (Sept. 1977), pp. 505–517. DOI: 10.1137/0206036.

[241] Takeaki Uno. "A Fast Algorithm for Enumeration of Maximal Matchings in General Graphs". In: *Journal of National Institute of Informatics* 3.- (2001), pp. 89–97.

[242] Takeaki Uno. "A New Approach for Speeding Up Enumeration Algorithms and Its Application for Matroid Bases". In: *COCOON 1999: the 5th Annual International Conference on Computing and Combinatorics*. Ed. by Takano Asano et al. Vol. 1627. Lecture Notes in

Computer Science. Tokyo, Japan: Springer Berlin Heidelberg, July 1999, pp. 349–359. DOI: 10.1007/3-540-48686-0_35.

[243] Takeaki Uno. "Algorithms for Enumerating All Perfect, Maximum and Maximal Matchings in Bipartite Graphs". In: *ISAAC 1997: the 8th International Symposium on Algorithms and Computation*. Ed. by Hon Wai Leong, Hiroshi Imai, and Sanjay Jain. Vol. 1350. Lecture Notes in Computer Science. Singapore: Springer Berlin Heidelberg, Dec. 1997, pp. 92–101. DOI: 10.1007/3-540-63890-3_11.

[244] Takeaki Uno. "An algorithm for enumerating all directed spanning trees in a directed graph". In: *ISAAC 1996: the 7th International Symposium on Algorithms and Computation*. Ed. by Tetsuo Asano et al. Vol. 1178. Lecture Notes in Computer Science. Osaka, Japan: Springer Berlin Heidelberg, Dec. 1996, pp. 166–173. DOI: 10.1007/BFb0009492.

[245] Takeaki Uno. "An efficient algorithm for solving pseudo clique enumeration problem". In: *Algorithmica* 56.1 (Oct. 2010), pp. 3–16. DOI: 10.1007/s00453-008-9238-3.

[246] Takeaki Uno. "New Approach for Speeding Up Enumeration Algorithms". In: *ISAAC 1998: the 9th International Symposium on Algorithms and Computation*. Ed. by Kyung-Yong Chwa and Oscar H. Ibarra. Vol. 1533. Lecture Notes in Computer Science. Taejon, Korea: Springer Berlin Heidelberg, Dec. 1998, pp. 287–296. DOI: 10.1007/3-540-49381-6_31.

[247] Takeaki Uno and Hiroko Satoh. "An Efficient Algorithm for Enumerating Chordless Cycles and Chordless Paths". In: *DS 2014: the 17th International Conference on Discovery Science*. Ed. by Sašo Džeroski et al. Vol. 8777. Lecture Notes in Computer Science. Bled, Slovenia: Springer Berlin Heidelberg, Oct. 2014, pp. 313–324. DOI: 10.1007/978-3-319-11812-3_27.

[248] Y. L. Varol. "An Algorithm to Generate All Topological Sorting Arrangements". In: *The Computer Journal* 24.1 (Jan. 1981), pp. 83–84. DOI: 10.1093/comjnl/24.1.83.

[249] Timothy R. Walsh. "Generation of Well-Formed Parenthesis Strings in Constant Worst-Case Time". In: *Journal of Algorithms* 29.1 (Oct. 1998), pp. 165–173. DOI: 10.1006/jagm.1998.0960.

[250] Jianxin Wang et al. "An efficient fixed-parameter enumeration algorithm for weighted edge dominating set". In: *FAW 2009: the 3rd International Workshop on Frontiers in Algorithmics*. Ed. by Xiaotie Deng, John E. Hopcroft, and Jinyun Xue. Vol. 5598. Lecture Notes in Computer Science. Hefei, China: Springer Berlin Heidelberg, June 2009, pp. 237–250. DOI: 10.1007/978-3-642-02270-8_25.

[251] Terry Min Yih Wang and Carla D. Savage. "A Gray Code for Necklaces of Fixed Density". In: *SIAM Journal on Discrete Mathematics* 9.4 (Nov. 1996), pp. 654–673. DOI: 10.1137/S089548019528143X.

[252] Kunihiro Wasa, Hiroki Arimura, and Takeaki Uno. "Efficient Enumeration of Induced Subtrees in a K-Degenerate Graph". In: *ISAAC 2014: the 25th International Symposium on Algorithms and Computation*. Ed. by Hee-Kap Ahn and Chan-Su Shin. Vol. 6506. Lecture Notes in Computer Science. Jeonju, Korea: Springer Berlin Heidelberg, Dec. 2014, p. 488. DOI: 10.1007/978-3-319-13075-0_8.

[253] Kunihiro Wasa et al. "Constant Time Enumeration of Bounded-Size Subtrees in Trees and Its Application". In: *COCOON 2012: the 18th Annual International Conference on Computing and Combinatorics*. Ed. by Joachim Gudmundsson, Julián Mestre, and Taso Viglas. Vol. 7434. Lecture Notes in Computer Science. Sydney, Australia: Springer Berlin Heidelberg, Aug. 2012, pp. 347–359. DOI: 10.1007/978-3-642-32241-9_3

[254] Kunihiro Wasa et al. "Polynomial delay and space discovery of connected and acyclic sub-hypergraphs in a hypergraph". In: *DS 2013: the 16th International Conference on Discovery Science*. Ed. by Johannes Fürnkranz, Eyke Hüllermeier, and Tomoyuki Higuchi. Vol. 8140. Lecture Notes in Computer Science. Singapore: Springer Berlin Heidelberg, Oct. 2013, pp. 308–323. DOI: 10.1007/978-3-642-40897-7_21.

[255] Herbert Weinblatt. "A New Search Algorithm for Finding the Simple Cycles of a Finite Directed Graph". In: *Journal of the ACM* 19.1 (Jan. 1972), pp. 43–56. DOI: 10.1145/321679.321684.

[256] J T Welch. "Cycle Algorithms for Undirected Linear Graphs and Some immediate Applications". In: *Proceedings of the 1965 20th national conference*. New York, New York, USA: ACM Press, Aug. 1965, pp. 296–301. DOI: 10.1145/800197.806053.

[257] John T. Welch. "A Mechanical Analysis of the Cyclic Structure of Undirected Linear Graphs". In: *Journal of the ACM* 13.2 (Apr. 1966), pp. 205–210. DOI: 10.1145/321328.321331.

[258] Mark B. Wells. “Generation of permutations by transposition”. In: *Mathematics of Computation* 15.74 (1961), pp. 192–195. DOI: 10.1090/S0025-5718-1963-01597

[259] Manuel Wettstein. “Counting and Enumerating Crossing-free Geometric Graphs”. In: *Annual Symposium on Computational Geometry - SOCG’14*. New York, New York, USA: ACM Press, June 2014, pp. 1–10. DOI: 10.1145/2582112.2582145.

[260] John S. White. “Algorithm 374: Restricted partition generator [A1]”. In: *Communications of the ACM* 13.2 (Feb. 1970), p. 120. DOI: 10.1145/362007.362046.

[261] Marcel Wild. “Generating all cycles, chordless cycles, and Hamiltonian cycles with the principle of exclusion”. In: *Journal of Discrete Algorithms* 6.1 (Mar. 2008), pp. 93–102. DOI: 10.1016/j.jda.2007.01.005.

[262] Marcel Wild. “Output-Polynomial Enumeration of All Fixed-Cardinality Ideals of a Poset, Respectively All Fixed-Cardinality Subtrees of a Tree”. In: *Order* 31.1 (Apr. 2013), pp. 121–135. DOI: 10.1007/s11083-013-9292-6.

[263] Robert Alan Wright et al. “Constant Time Generation of Free Trees”. In: *SIAM Journal on Computing* 15.2 (May 1986), pp. 540–548. DOI: 10.1137/0215039.

[264] L. Xiang. “Grammar-Oriented Enumeration of Binary Trees”. In: *The Computer Journal* 40.5 (May 1997), pp. 278–291. DOI: 10.1093/comjnl/40.5.278.

[265] Takeo Yamada, Seiji Kataoka, and Kohtaro Watanabe. “Listing All the Minimum Spanning Trees in an Undirected Graph”. In: *International Journal of Computer Mathematics* 87.14 (Nov. 2010), pp. 3175–3185. DOI: 10.1080/00207160903329699.

[266] Katsuhisa Yamanaka and Shin-ichi Nakano. “Efficient Enumeration of All Ladder Lotteries with k Bars”. In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* E97.A.6 (2014), pp. 1163–1170. DOI: 10.1587/transfun.E97.A.1163.

[267] Katsuhisa Yamanaka and Shin-ichi Nakano. “Generating all realizers”. In: *Electronics and Communications in Japan (Part II: Electronics)* 89.7 (2006), pp. 40–47. DOI: 10.1002/ecjb.20276.

[268] Katsuhisa Yamanaka and Shin-ichi Nakano. “Listing All Plane Graphs”. In: *Journal of Graph Algorithms and Applications* 13.1 (2009), pp. 5–18. DOI: 10.7155/jgaa.00174.

[269] Katsuhisa Yamanaka, Yota Otachi, and Shin-ichi Nakano. "Efficient Enumeration of Ordered Trees with k Leaves". In: *WALCOM 2009: the 3rd International Workshop on Algorithms and Computation*. Ed. by Sandip Das and Ryuhei Uehara. Vol. 5431. Lecture Notes in Computer Science. Kolkata, India: Springer Berlin Heidelberg, Feb. 2009, pp. 141–150. DOI: 10.1007/978-3-642-00202-1_13.

[270] Katsuhisa Yamanaka et al. "Efficient Enumeration of All Ladder Lotteries and Its Application". In: *Theoretical Computer Science* 411.16-18 (Mar. 2010), pp. 1714–1722. DOI: 10.1016/j.tcs.2010.01.002.

[271] Yoshida Yasuko and Tomomi Matsui. "Finding All the Edge Colorings in Bipartite Graphs (in Japanese)". In: *IEEJ Transactions on Electronics, Information and Systems* 114.4 (1994), pp. 444–449.

[272] S. Yau. "Generation of all Hamiltonian Circuits, Paths, and Centers of a Graph, and Related Problems". In: *IEEE Transactions on Circuit Theory* 14.1 (Mar. 1967), pp. 79–81. DOI: 10.1109/TCT.1967.1082662.

[273] Li Pu Yeh, Biing Feng Wang, and Hsin Hao Su. "Efficient algorithms for the problems of enumerating cuts by non-decreasing weights". In: *Algorithmica* 56.3 (2010), pp. 297–312. DOI: 10.1007/s00453-009-9284-5.

[274] J. Y. Yen. "Finding the K Shortest Loopless Paths in a Network". In: *Management Science* 17.11 (1971), pp. 712–716. DOI: 10.1287/mnsc.17.11.712.

[275] Shmuel Zaks. "Generation and ranking of K-ary trees". In: *Information Processing Letters* 14.1 (Mar. 1982), pp. 44–48. DOI: 10.1016/0020-0190(82)90140-5.

[276] Shmuel Zaks. "Lexicographic generation of ordered trees". In: *Theoretical Computer Science* 10.1 (Jan. 1980), pp. 63–82. DOI: 10.1016/0304-3975(80)90073-0.

[277] Shumel Zaks and D. Richards. "Generating Trees and Other Combinatorial Objects Lexicographically". en. In: *SIAM Journal on Computing* 8.1 (Feb. 1979), pp. 73–81. DOI: 10.1137/0208006.

[278] David Zerling. "Generating Binary Trees Using Rotations". In: *Journal of the ACM* 32.3 (July 1985), pp. 694–701. DOI: 10.1145/3828.214141.

[279] Bingbing Zhuang and Hiroshi Nagamochi. "Constant Time Generation of Biconnected Rooted Plane Graphs". In: *FAW 2010: the 4th International Workshop on Frontiers in Algorithmics*. Ed. by Der-Tsai Lee, Danny Z. Chen, and Shi Ying. Vol. 6213. Lecture Notes in Computer Science. Wuhan, China: Springer Berlin Heidelberg, Aug. 2010, pp. 113–123. DOI: 10.1007/978-3-642-14553-7_13.

[280] Bingbing Zhuang and Hiroshi Nagamochi. "Generating Internally Triconnected rooted plane graphs". In: *TAMC 2010: the 7th Annual Conference on Theory and Applications of Models of Computation*. Ed. by Jan Kratochvíl et al. Vol. 6108. Lecture Notes in Computer Science. Prague, Czech Republic: Springer Berlin Heidelberg, June 2010, pp. 467–478. DOI: 10.1007/978-3-642-13562-0_42.

[281] Bingbing Zhuang and Hiroshi Nagamochi. "Listing triconnected rooted plane graphs". In: *COCOA 2010: the 4th International Conference on Combinatorial Optimization and Applications*. Ed. by Weili Wu and Ovidiu Daescu. Vol. 6509. Lecture Notes in Computer Science. Kailua-Kona, HI, USA: Springer Berlin Heidelberg, Dec. 2010, pp. 347–361. DOI: 10.1007/978-3-642-17461-2_28.