WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

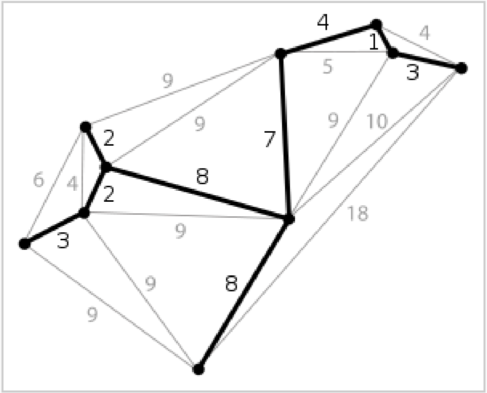
Print/export
Create a book
Download as PDF
Printable version

Languages
Ελληνικά
Español
Esperanto
فارسی
Français
Հայերեն
Italiano
עברית
Magyar
Nederlands
Polski
Português
Русский
Simple English
Slovenčina
Slovenščina
Српски / srpski
Svenska
ไทย
Українська
اردو
Tiếng Việt
中文
Edit links

Minimum spanning tree

From Wikipedia, the free encyclopedia

Given a connected, undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. We can also assign a *weight* to each edge, which is a number representing how unfavorable it is, and use this to assign a weight to a spanning tree by computing the sum of the weights of the edges in that spanning tree. A **minimum spanning tree** (**MST**) or **minimum weight spanning tree** is then a spanning tree with weight less than or equal to the weight of every other spanning tree. More generally, any undirected graph (not necessarily connected) has a **minimum spanning forest**, which is a union of minimum spanning trees for its connected components.



The only minimum spanning tree of a planar graph. Each edge is labeled with its weight, which here is roughly proportional to its length.

One example would be a telecommunications company laying cable to a new neighborhood. If it is constrained to bury the cable only along certain paths (e.g. along roads), then there would be a graph representing which points are connected by those paths. Some of those paths might be more expensive, because they are longer, or require the cable to be buried deeper; these paths would be represented by edges with larger weights. Currency is an acceptable unit for edge weight — there is no requirement for edge lengths to obey normal rules of geometry such as the triangle inequality. A *spanning tree* for that graph would be a subset of those paths that has no cycles but still connects to every house; there might be several spanning trees possible. A *minimum spanning tree* would be one with the lowest total cost, thus would represent the least expensive path for laying the cable.

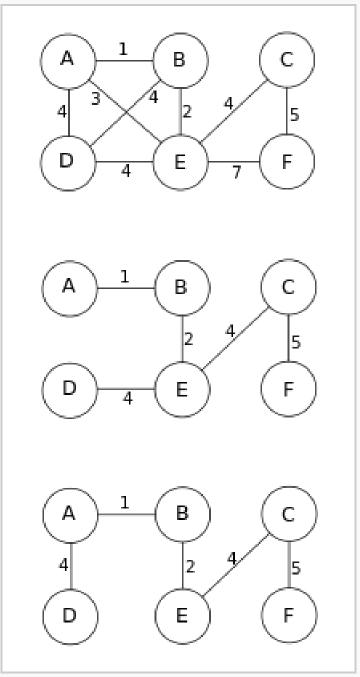
Contents [hide]

Properties [edit]

Possible multiplicity [edit]

There may be several minimum spanning trees of the same weight having a minimum number of edges; in particular, if all the edge weights of a given graph are the same, then every spanning tree of that graph is minimum.

If there are *n* vertices in the graph, then each spanning tree has *n*-1 edges.



This figure shows there may be more than one minimum spanning tree in a graph. In the figure, the two trees below the graph are two possibilities of minimum spanning tree of the given graph.

Uniqueness [edit]

If each edge has a distinct weight then there will be only one, unique minimum spanning tree. This is true in many realistic situations, such as the telecommunications company example above, where it's unlikely any two paths have *exactly* the same cost. This generalizes to spanning forests as well.

If the edge weights are not unique, only the (multi-)set of weights in minimum spanning trees is unique, that is the same for all minimum spanning trees.[1]

Proof:

1. Assume the contrary, that there are two different MSTs *A* and *B*.
2. Let e_1 be the edge of least weight that is in one of the MSTs and not the other. Without loss of generality, assume e_1 is in *A* but not in *B*.
3. As *B* is a MST, $\{e_1\} \cup B$ must contain a cycle *C*.
4. Then *C* has an edge e_2 whose weight is greater than the weight of e_1, since all edges in *B* with less weight are in *A* by the choice of e_1, and *C* must have at least one edge that is not in *A* because otherwise *A* would contain a cycle in contradiction with its being an MST.
5. Replacing e_2 with e_1 in *B* yields a spanning tree with a smaller weight.
6. This contradicts the assumption that *B* is a MST.

Minimum-cost subgraph [edit]

If the weights are *positive*, then a minimum spanning tree is in fact a minimum-cost subgraph connecting all vertices, since subgraphs containing cycles necessarily have more total weight.

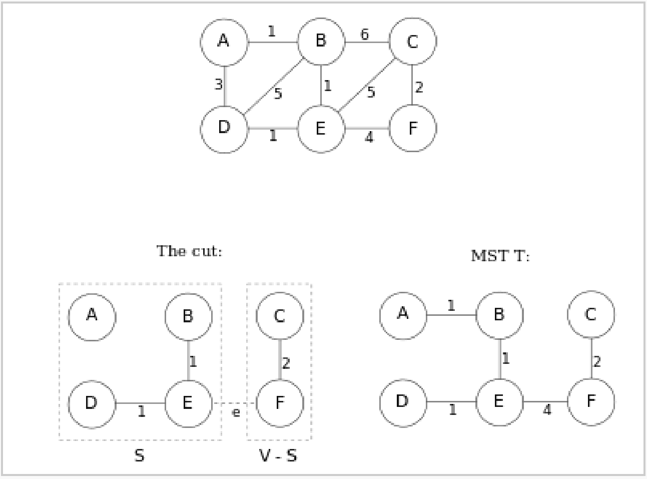
Cycle property [edit]

For any cycle C *in the graph, if the weight of an edge* e *of* C *is larger than the individual weights of all other edges of* C*, then this edge cannot belong to a MST.*

Proof: Assume the contrary, i.e. that *e* belongs to an MST T1. Then deleting *e* will break T1 into two subtrees with the two ends of *e* in different subtrees. The remainder of *C* reconnects the subtrees, hence there is an edge *f* of *C* with ends in different subtrees, i.e., it reconnects the subtrees into a tree T2 with weight less than that of T1, because the weight of *f* is less than the weight of *e*.

Cut property [edit]

For any cut C *in the graph, if the weight of an edge* e *of* C *is strictly smaller than the weights of all other edges of* C*, then this edge belongs to all MSTs of the graph.*



Proof: assume the contrary, i.e., in the figure at right, make edge BC (weight 6) part of the MST T instead of edge e (weight 4). Adding e to T will produce a cycle, while replacing BC with e would produce MST of smaller weight. Thus, a tree containing BC is not a MST, a contradiction that violates our assumption. By a similar argument, if more than one edge is of minimum weight across a cut, then each such edge is contained in some minimum spanning tree.

This figure shows the cut property of MSP. T is the only MST of the given graph. If S = {A,B,D,E}, thus V-S = {C,F}, then there are 3 possibilities of the edge across the cut(S,V-S), they are edges BC, EC, EF of the original graph. Then, e is one of the minimum-weight-edge for the cut, therefore S ∪ {e} is part of the MST T.

Minimum-cost edge [edit]

If the edge of a graph with the minimum cost e *is unique, then this edge is included in any MST.*

Proof: if *e* was not included in the MST, removing any of the (larger cost) edges in the cycle formed after adding *e* to the MST, would yield a spanning tree of smaller weight.

Contraction [edit]

If T is a tree of MST edges, then we can contract *T into a single vertex while maintaining the invariant that the MST of the contracted graph plus T gives the MST for the graph before contraction.*[2]

Algorithms [edit]

In all of the algorithms below, "m" is the number of edges in the graph and "n" is the number of vertices.

Classic algorithms [edit]

The first algorithm for finding a minimum spanning tree was developed by Czech scientist Otakar Borůvka in 1926 (see Borůvka's algorithm). Its purpose was an efficient electrical coverage of Moravia. The algorithm proceeds in a sequence of stages. In each stage, called *Boruvka step*, it identifies a forest *F* consisting of the minimum-weight edge incident to each vertex in the graph *G*, then forms the graph G1=G\F as the input to the next step. Here G\F denotes the graph derived from G by contracting edges in F (by the Cut property, these edges belong to the MST). Each Boruvka step takes linear time. Since the number of vertices is reduced by at least half in each step, Boruvka's algorithm takes O(m log n) time.[2]

A second algorithm is Prim's algorithm, which was invented by Jarnik in 1930 and rediscovered by Prim in 1957 and Dijkstra in 1959. Basically, it grows the MST (T) one edge at a time. Initially, T contains an arbitrary vertex. In each step, T is augmented with the least-weight edge (x,y) such that x is in T and y is not yet in T. By the Cut property, all edges added to T are in the MST. Its run-time is either O(m log n) or O(m + n log n), depending on the data-structures used.

A third algorithm commonly in use is the Kruskal's algorithm, which also takes O(m log n) time.

A fourth algorithm, not as commonly used, is the reverse-delete algorithm, which is the reverse of Kruskal's algorithm. Its runtime is O(m log n (log log n)^3).

All these four are greedy algorithms. Since they run in polynomial time, the problem of finding such trees is in **FP**, and related decision problems such as determining whether a particular edge is in the MST or determining if the minimum total weight exceeds a certain value are in **P**.

Faster algorithms [edit]

Several researchers have tried to find more computationally-efficient algorithms.

In a comparison model, in which the only allowed operations on edge weights are pairwise comparisons, Karger, Klein & Tarjan (1995) found a linear time randomized algorithm based on a combination of Borůvka's algorithm and the reverse-delete algorithm.[3][4]

The fastest non-randomized comparison-based algorithm with known complexity, by Bernard Chazelle, is based on the soft heap, an approximate priority queue.[5][6] Its running time is O(m α(m,n)), where α is the classical functional inverse of the Ackermann function. The function α grows extremely slowly, so that for all practical purposes it may be considered a constant no greater than 4; thus Chazelle's algorithm takes very close to linear time.

Linear-time algorithms in special cases [edit]

Dense graphs [edit]

If the graph is dense (i.e. m/n ≥ log log log n), then a deterministic algorithm by Fredman and Tarjan finds the MST in time O(m).[7] The algorithm executes a number of phases. Each phase executes Prim's algorithm many times, each for a limited number of steps. The run-time of each phase is O(m+n). If the number of vertices before a phase is n', the number of vertices remaining after a phase is at most $n'/2^{m/n'}$. Hence, at most $\log *n$ phases are needed, which gives a linear run-time for dense graphs.[2]

There are other algorithms that work in linear time on dense graphs.[5][8]

Integer weights [edit]

If the edge weights are integers represented in binary, then deterministic algorithms are known that solve the problem in *O*(*m* + *n*) integer operations.[9] Whether the problem can be solved *deterministically* for a *general graph* in *linear time* by a comparison-based algorithm remains an open question.

Decision trees [edit]

Given graph *G* where the nodes and edges are fixed but the weights are unknown, it is possible to construct a binary decision tree (DT) for calculating the MST for any permutation of weights. Each internal node of the DT contains a comparison between two edges, e.g. "Is the weight of the edge between *x* and *y* larger than the weight of the edge between *w* and *z*?". The two children of the node correspond to the two possible answers "yes" or "no". In each leaf of the DT, there is a list of edges from *G* that correspond to an MST. The runtime complexity of a DT is the largest number of queries required to find the MST, which is just the depth of the DT. A DT for a graph *G* is called *optimal* if it has the smallest depth of all correct DTs for *G*.

For every integer *r*, it is possible to find optimal decision trees for all graphs on *r* vertices by brute-force search. This search proceeds in two steps.

A. Generating all potential DTs

- There are $2^{\binom{r}{2}}$ different graphs on *r* vertices.
- For each graph, an MST can always be found using *r*(*r*-1) comparisons, e.g. by Prim's algorithm.
- Hence, the depth of an optimal DT is less than r^2.
- Hence, the number of internal nodes in an optimal DT is less than 2^{r^2}.
- Every internal node compares two edges. The number of edges is at most r^2 so the different number of comparisons is at most r^4.
- Hence, the number of potential DTs is less than: $(r^4)^{(2^{r^2})} = r^{2^{(r^2+2)}}$.

B. Identifying the correct DTs To check if a DT is correct, it should be checked on all possible permutations of the edge weights.

- The number of such permutations is at most $(r^2)!$.
- For each permutation, solve the MST problem on the given graph using any existing algorithm, and compare the result to the answer given by the DT.
- The running time of any MST algorithm is at most (r^2), so the total time required to check all permutations is at most $(r^2+1)!$.

Hence, the total time required for finding an optimal DT for *all* graphs with *r* vertices is: $2^{\binom{r}{2}} \cdot r^{2^{(r^2+2)}} \cdot (r^2+1)!$, which is less than: $2^{2^{r^2+o(r)}}$.[2]

See also: Decision tree model

Optimal algorithm [edit]

Seth Pettie and Vijaya Ramachandran have found a provably optimal deterministic comparison-based minimum spanning tree algorithm.[2] The following is a simplified description of the algorithm.

1. Let $r = \log\log\log n$, where *n* is the number of vertices. Find all optimal decision trees on *r* vertices. This can be done in time O(*n*) (see Decision trees above).
2. Partition the graph to components with at most *r* vertices in each component. This partition can be done in time O(*m*).
3. Use the optimal decision trees to find an MST for each component.
4. Contract each connected component spanned by the MSTs to a single vertex.
5. It is possible to prove that the resulting graph has at most *n*/*r* vertices. Hence, the graph is dense and we can use any algorithm which works on Dense graphs in time O(*m*).

The runtime of all steps in the algorithm is O(*m*), *except for the step of using the decision trees*. We don't know the runtime of this step, but we know that it is optimal - no algorithm can do better than the optimal decision tree.

Thus, this algorithm has the peculiar property that it is *provably optimal* although its runtime complexity is *unknown*.

Parallel and distributed algorithms [edit]

Research has also considered parallel algorithms for the minimum spanning tree problem. With a linear number of processors it is possible to solve the problem in $O(\log n)$ time.[10][11] Bader & Cong (2003) demonstrate an algorithm that can compute MSTs 5 times faster on 8 processors than an optimized sequential algorithm.[12]

Other specialized algorithms have been designed for computing minimum spanning trees of a graph so large that most of it must be stored on disk at all times. These *external storage* algorithms, for example as described in "Engineering an External Memory Minimum Spanning Tree Algorithm" by Roman, Dementiev et al.,[13] can operate, by authors' claims, as little as 2 to 5 times slower than a traditional in-memory algorithm. They rely on efficient external storage sorting algorithms and on graph contraction techniques for reducing the graph's size efficiently.

The problem can also be approached in a distributed manner. If each node is considered a computer and no node knows anything except its own connected links, one can still calculate the distributed minimum spanning tree.

MST on complete graphs [edit]

Alan M. Frieze showed that given a complete graph on *n* vertices, with edge weights that are independent identically distributed random variables with distribution function F satisfying $F'(0) > 0$, then as *n* approaches +∞ the expected weight of the MST approaches $\zeta(3)/F'(0)$, where ζ is the Riemann zeta function. Frieze and Steele also proved convergence in probability. Svante Janson proved a central limit theorem for weight of the MST.

For uniform random weights in $[0, 1]$, the exact expected size of the minimum spanning tree has been computed for small complete graphs.[14]

Vertices	Expected size	Approximative expected size
2	1 / 2	0.5
3	3 / 4	0.75
4	31 / 35	0.8857143
5	893 / 924	0.9664502
6	278 / 273	1.0183151
7	30739 / 29172	1.053716
8	199462271 / 184848378	1.0790588
9	126510063932 / 115228853025	1.0979027

Applications [edit]

Minimum spanning trees have direct applications in the design of networks, including computer networks, telecommunications networks, transportation networks, water supply networks, and electrical grids (which they were first invented for, as mentioned above).[15] They are invoked as subroutines in algorithms for other problems, including the Christofides algorithm for approximating the traveling salesman problem,[16] approximating the multi-terminal minimum cut problem (which is equivalent in the single-terminal case to the maximum flow problem),[17] and approximating the minimum-cost weighted perfect matching.[18]

Other practical applications based on minimal spanning trees include:

- Taxonomy.[19]
- Cluster analysis: clustering points in the plane,[20] single-linkage clustering (a method of hierarchical clustering),[21] graph-theoretic clustering,[22] and clustering gene expression data.[23]
- Constructing trees for broadcasting in computer networks.[24] On Ethernet networks this is accomplished by means of the Spanning tree protocol.
- Image registration[25] and segmentation[26] — see minimum spanning tree-based segmentation.
- Curvilinear feature extraction in computer vision.[27]
- Handwriting recognition of mathematical expressions.[28]
- Circuit design: implementing efficient multiple constant multiplications, as used in finite impulse response filters.[29]
- Regionalisation of socio-geographic areas, the grouping of areas into homogeneous, contiguous regions.[30]
- Comparing ecotoxicology data.[31]

- Topological observability in power systems.[32]
- Measuring homogeneity of two-dimensional materials.[33]
- Minimax process control.[34]
- Minimum spanning trees can also be used to describe financial markets.[35] A correlation matrix can be created by calculating a coefficient of correlation between any two stocks.This matrix can be represented topologically as a complex network and a minimum spanning tree can be constructed to visualize relationships.

Related problems [edit]

The problem of finding the Steiner tree of a subset of the vertices, that is, minimum tree that spans the given subset, is known to be NP-Complete.[36]

A related problem is the *k*-minimum spanning tree (*k*-MST), which is the tree that spans some subset of *k* vertices in the graph with minimum weight.

A set of *k-smallest spanning trees* is a subset of *k* spanning trees (out of all possible spanning trees) such that no spanning tree outside the subset has smaller weight.[37][38][39] (Note that this problem is unrelated to the *k*-minimum spanning tree.)

The Euclidean minimum spanning tree is a spanning tree of a graph with edge weights corresponding to the Euclidean distance between vertices which are points in the plane (or space).

The rectilinear minimum spanning tree is a spanning tree of a graph with edge weights corresponding to the rectilinear distance between vertices which are points in the plane (or space).

In the distributed model, where each node is considered a computer and no node knows anything except its own connected links, one can consider distributed minimum spanning tree. The mathematical definition of the problem is the same but there are different approaches for a solution.

The capacitated minimum spanning tree is a tree that has a marked node (origin, or root) and each of the subtrees attached to the node contains no more than a *c* nodes. *c* is called a tree capacity. Solving CMST optimally is NP-hard,[40] but good heuristics such as Esau-Williams and Sharma produce solutions close to optimal in polynomial time.

The degree constrained minimum spanning tree is a minimum spanning tree in with each vertex is connected to no more than *d* other vertices, for some given number *d*. The case *d* = 2 is a special case of the traveling salesman problem, so the degree constrained minimum spanning tree is NP-hard in general.

For directed graphs, the minimum spanning tree problem is called the Arborescence problem and can be solved in quadratic time using the Chu–Liu/Edmonds algorithm.

A **maximum spanning tree** is a spanning tree with weight greater than or equal to the weight of every other spanning tree. Such a tree can be found with algorithms such as Prim's or Kruskal's after multiplying the edge weights by -1 and solving the MST problem on the new graph. A path in the maximum spanning tree is the widest path in the graph between its two endpoints: among all possible paths, it maximizes the weight of the minimum-weight edge.[41] Maximum spanning trees find applications in parsing algorithms for natural languages[42] and in training algorithms for conditional random fields.

The **dynamic MST** problem concerns the update of a previously computed MST after an edge weight change in the original graph or the insertion/deletion of a vertex.[43][44][45]

The minimum labeling spanning tree problem is to find a spanning tree with least types of labels if each edge in a graph is associated with a label from a finite label set instead of a weight.[46]

A bottleneck edge is the highest weighted edge in a spanning tree. A spanning tree is a **minimum bottleneck spanning tree** (or **MBST**) if the graph does not contain a spanning tree with a smaller bottleneck edge weight. A MST is necessarily a MBST (provable by the cut property), but a MBST is not necessarily a MST.[47][48]

References [edit]

1. ^ Do the minimum spanning trees of a weighted graph have the same number of edges with a given weight?
2. ^ *a b c d e* Pettie, Seth; Ramachandran, Vijaya (2002), "An optimal minimum spanning tree algorithm", *Journal of the Association for Computing Machinery* **49** (1): 16–34, doi:10.1145/505241.505243, MR 2148431.
3. ^ Karger, David R.; Klein, Philip N.; Tarjan, Robert E. (1995), "A randomized linear-time algorithm to find minimum spanning trees", *Journal of the Association for Computing Machinery* **42** (2): 321–328, doi:10.1145/201019.201022, MR 1409738
4. ^ Pettie, Seth; Ramachandran, Vijaya (2002), "Minimizing randomness in minimum spanning tree, parallel connectivity, and set maxima algorithms", *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA*

'02), San Francisco, California, pp. 713–722.
5. ^ *a* *b* Chazelle, Bernard (2000), "A minimum spanning tree algorithm with inverse-Ackermann type complexity", *Journal of the Association for Computing Machinery* **47** (6): 1028–1047, doi:10.1145/355541.355562, MR 1866456.
6. ^ Chazelle, Bernard (2000), "The soft heap: an approximate priority queue with optimal error rate", *Journal of the Association for Computing Machinery* **47** (6): 1012–1027, doi:10.1145/355541.355554, MR 1866455.
7. ^ Fredman, M. L.; Tarjan, R. E. (1987). "Fibonacci heaps and their uses in improved network optimization algorithms". *Journal of the ACM* **34** (3): 596. doi:10.1145/28869.28874.
8. ^ Gabow, H. N.; Galil, Z.; Spencer, T.; Tarjan, R. E. (1986). "Efficient algorithms for finding minimum spanning trees in undirected and directed graphs". *Combinatorica* **6** (2): 109. doi:10.1007/bf02579168.
9. ^ Fredman, M. L.; Willard, D. E. (1994), "Trans-dichotomous algorithms for minimum spanning trees and shortest paths", *Journal of Computer and System Sciences* **48** (3): 533–551, doi:10.1016/S0022-0000(05)80064-9, MR 1279413.
10. ^ Chong, Ka Wong; Han, Yijie; Lam, Tak Wah (2001), "Concurrent threads and optimal parallel minimum spanning trees algorithm", *Journal of the Association for Computing Machinery* **48** (2): 297–323, doi:10.1145/375827.375847, MR 1868718.
11. ^ Pettie, Seth; Ramachandran, Vijaya (2002), "A randomized time-work optimal parallel algorithm for finding a minimum spanning forest", *SIAM Journal on Computing* **31** (6): 1879–1895, doi:10.1137/S0097539700371065, MR 1954882.
12. ^ Bader, David A.; Cong, Guojing (2006), "Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs", *Journal of Parallel and Distributed Computing* **66** (11): 1366–1378, doi:10.1016/j.jpdc.2006.06.001.
13. ^ Dementiev, Roman; Sanders, Peter; Schultes, Dominik; Sibeyn, Jop F. (2004), "Engineering an external memory minimum spanning tree algorithm", *Proc. IFIP 18th World Computer Congress, TC1 3rd International Conference on Theoretical Computer Science (TCS2004)* (PDF), pp. 195–208.
14. ^ Steele, J. Michael (2002), "Minimal spanning trees for graphs with random edge lengths", *Mathematics and computer science, II (Versailles, 2002)*, Trends Math., Basel: Birkhäuser, pp. 223–245, MR 1940139.
15. ^ Graham, R. L.; Hell, Pavol (1985), "On the history of the minimum spanning tree problem", *Annals of the History of Computing* **7** (1): 43–57, doi:10.1109/MAHC.1985.10011, MR 783327.
16. ^ Nicos Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Report 388, Graduate School of Industrial Administration, CMU, 1976.
17. ^ Dahlhaus, E.; Johnson, D. S.; Papadimitriou, C. H.; Seymour, P. D.; Yannakakis, M. (August 1994). "The complexity of multiterminal cuts" (PDF). *SIAM Journal on Computing* **23** (4): 864–894. doi:10.1137/S0097539792225297. Retrieved 17 December 2012.
18. ^ Supowit, Kenneth J.; Plaisted, David A.; Reingold, Edward M. (1980). *Heuristics for weighted perfect matching*. 12th Annual ACM Symposium on Theory of Computing (STOC '80). New York, NY, USA: ACM. pp. 398–419. doi:10.1145/800141.804689.
19. ^ Sneath, P. H. A. (1 August 1957). "The Application of Computers to Taxonomy". *Journal of General Microbiology* **17** (1): 201–226. doi:10.1099/00221287-17-1-201. PMID 13475686.
20. ^ Asano, T.; Bhattacharya, B.; Keil, M.; Yao, F. (1988). *Clustering algorithms based on minimum and maximum spanning trees*. Fourth Annual Symposium on Computational Geometry (SCG '88) **1**. pp. 252–257. doi:10.1145/73393.73419.
21. ^ Gower, J. C.; Ross, G. J. S. (1969). "Minimum Spanning Trees and Single Linkage Cluster Analysis". *Journal of the Royal Statistical Society*. C (Applied Statistics) **18** (1): 54–64. doi:10.2307/2346439.
22. ^ Päivinen, Niina (1 May 2005). "Clustering with a minimum spanning tree of scale-free-like structure". *Pattern Recognition Letters* **26** (7): 921–930. doi:10.1016/j.patrec.2004.09.039.
23. ^ Xu, Y.; Olman, V.; Xu, D. (1 April 2002). "Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees". *Bioinformatics* **18** (4): 536–545. doi:10.1093/bioinformatics/18.4.536. PMID 12016051.
24. ^ Dalal, Yogen K.; Metcalfe, Robert M. (1 December 1978). "Reverse path forwarding of broadcast packets". *Communications of the ACM* **21** (12): 1040–1048. doi:10.1145/359657.359665.
25. ^ Ma, B.; Hero, A.; Gorman, J.; Michel, O. (2000). *Image registration with minimum spanning tree algorithm* (PDF). International Conference on Image Processing **1**. pp. 481–484. doi:10.1109/ICIP.2000.901000.
26. ^ P. Felzenszwalb, D. Huttenlocher: Efficient Graph-Based Image Segmentation. IJCV 59(2) (September 2004)
27. ^ Suk, Minsoo; Song, Ohyoung (1 June 1984). "Curvilinear feature extraction using minimum spanning trees". *Computer Vision, Graphics, and Image Processing* **26** (3): 400–411. doi:10.1016/0734-189X(84)90221-4.
28. ^ Tapia, Ernesto; Rojas, Raúl (2004). "Recognition of On-line Handwritten Mathematical Expressions Using a Minimum Spanning Tree Construction and Symbol Dominance". *Graphics Recognition. Recent Advances and Perspectives* (PDF). Lecture Notes in Computer Science **3088**. Berlin Heidelberg: Springer-Verlag. pp. 329–340. ISBN 3540224785.
29. ^ Ohlsson, H. (2004). *Implementation of low complexity FIR filters using a minimum spanning tree*. 12th IEEE Mediterranean Electrotechnical Conference (MELECON 2004) **1**. pp. 261–264. doi:10.1109/MELCON.2004.1346826.
30. ^ AssunÇão, R. M.; M. C. Neves; G. Câmara; C. Da Costa Freitas (2006). "Efficient regionalization techniques for

socioleconomic geographical units using minimum spanning trees". *International Journal of Geographical Information Science* **20** (7): 797–811. doi:10.1080/13658810600665111.

31. ^ Devillers, J.; Dore, J.C. (1 April 1989). "Heuristic potency of the minimum spanning tree (MST) method in toxicology". *Ecotoxicology and Environmental Safety* **17** (2): 227–235. doi:10.1016/0147-6513(89)90042-0. PMID 2737116.
32. ^ Mori, H.; Tsuzuki, S. (1 May 1991). "A fast method for topological observability analysis using a minimum spanning tree technique". *IEEE Transactions on Power Systems* **6** (2): 491–500. doi:10.1109/59.76691.
33. ^ Filliben, James J.; Kafadar, Karen; Shier, Douglas R. (1 January 1983). "Testing for homogeneity of two-dimensional surfaces". *Mathematical Modelling* **4** (2): 167–189. doi:10.1016/0270-0255(83)90026-X.
34. ^ Kalaba, Robert E. (1963), *Graph Theory and Automatic Control* (PDF)
35. ^ Djauhari, M., & Gan, S. (2015). Optimality problem of network topology in stocks market analysis. Physica A: Statistical Mechanics and Its Applications, 419, 108-114.
36. ^ Garey, Michael R.; Johnson, David S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, ISBN 0-7167-1045-5. ND12
37. ^ Gabow, Harold N. (1977), "Two algorithms for generating weighted spanning trees in order", *SIAM Journal on Computing* **6** (1): 139–150, doi:10.1137/0206011, MR 0441784.
38. ^ Eppstein, David (1992), "Finding the *k* smallest spanning trees", *BIT* **32** (2): 237–248, doi:10.1007/BF01994879, MR 1172188.
39. ^ Frederickson, Greg N. (1997), "Ambivalent data structures for dynamic 2-edge-connectivity and *k* smallest spanning trees", *SIAM Journal on Computing* **26** (2): 484–538, doi:10.1137/S0097539792226825, MR 1438526.
40. ^ Jothi, Raja; Raghavachari, Balaji (2005), "Approximation Algorithms for the Capacitated Minimum Spanning Tree Problem and Its Variants in Network Design", *ACM Trans. Algorithms* **1** (2): 265–282, doi:10.1145/1103963.1103967
41. ^ Hu, T. C. (1961), "The maximum capacity route problem", *Operations Research* **9** (6): 898–900, doi:10.1287/opre.9.6.898, JSTOR 167055.
42. ^ McDonald, Ryan; Pereira, Fernando; Ribarov, Kiril; Hajič, Jan (2005). "Non-projective dependency parsing using spanning tree algorithms" (PDF). *Proc. HLT/EMNLP*.
43. ^ Spira, P. M.; Pan, A. (1975), "On finding and updating spanning trees and shortest paths", *SIAM Journal on Computing* **4** (3): 375–380, doi:10.1137/0204032, MR 0378466.
44. ^ Holm, Jacob; de Lichtenberg, Kristian; Thorup, Mikkel (2001), "Poly-logarithmic deterministic fully dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity", *Journal of the Association for Computing Machinery* **48** (4): 723–760, doi:10.1145/502090.502095, MR 2144928.
45. ^ Chin, F.; Houck, D. (1978), "Algorithms for updating minimal spanning trees", *Journal of Computer and System Sciences* **16** (3): 333–344, doi:10.1016/0022-0000(78)90022-3.
46. ^ Chang, R.S.; Leu, S.J. (1997), "The minimum labeling spanning trees", *Information Processing Letters* **63** (5): 277–282, doi:10.1016/s0020-0190(97)00127-0.
47. ^ http://flashing-thoughts.blogspot.ru/2010/06/everything-about-bottleneck-spanning.html
48. ^ http://pages.cpsc.ucalgary.ca/~dcatalin/413/t4.pdf

Additional reading [edit]

- Otakar Boruvka on Minimum Spanning Tree Problem (translation of the both 1926 papers, comments, history) (2000) Jaroslav Nesetril, Eva Milková, Helena Nesetrilová. (Section 7 gives his algorithm, which looks like a cross between Prim's and Kruskal's.)
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Chapter 23: Minimum Spanning Trees, pp. 561–579.
- Eisner, Jason (1997). State-of-the-art algorithms for minimum spanning trees: A tutorial discussion. Manuscript, University of Pennsylvania, April. 78 pp.
- Kromkowski, John David. "Still Unmelted after All These Years", in Annual Editions, Race and Ethnic Relations, 17/e (2009 McGraw Hill) (Using minimum spanning tree as method of demographic analysis of ethnic diversity across the United States).

External links [edit]

- Implemented in BGL, the Boost Graph Library
- The Stony Brook Algorithm Repository - Minimum Spanning Tree codes
- Implemented in QuickGraph for .Net

Categories: Spanning tree | Polynomial-time problems