Create account Log in

Article Talk Read Edit View history Search

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

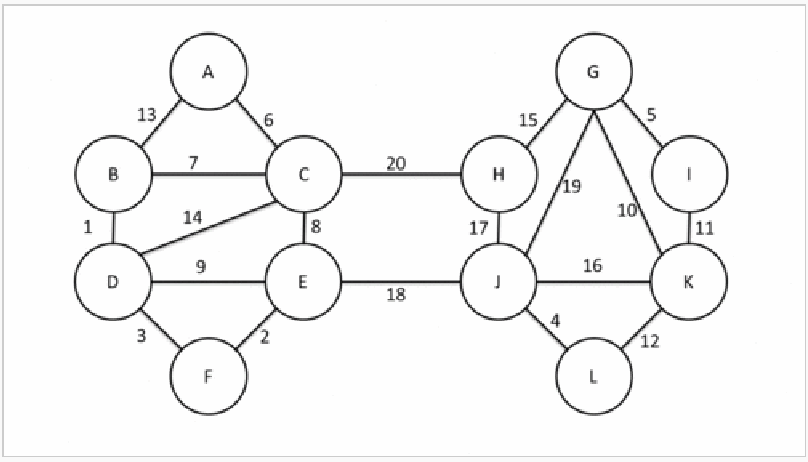
Languages
Čeština
Deutsch
Español
فارسی
Français
Italiano
Polski
Português
Русский
Српски / srpski
ไทย
Українська
Tiếng Việt
Edit links

Borůvka's algorithm

From Wikipedia, the free encyclopedia

Borůvka's algorithm is an algorithm for finding a minimum spanning tree in a graph for which all edge weights are distinct.

It was first published in 1926 by Otakar Borůvka as a method of constructing an efficient electricity network for Moravia.[1][2][3] The algorithm was rediscovered by Choquet in 1938;[4] again by Florek, Łukasiewicz, Perkal, Steinhaus, and Zubrzycki[5] in 1951; and again by Sollin [6] in 1965. Because Sollin was the only computer scientist in this list living in an English speaking country, this algorithm is frequently called **Sollin's algorithm**, especially in the parallel computing literature.



An animation, describing Boruvka's (Sollin's) algorithm, for finding a minimum spanning tree in a graph - An example on the runtime of the algorithm

The algorithm begins by first examining each vertex and adding the cheapest edge from that vertex to another in the graph, without regard to already added edges, and continues joining these groupings in a like manner until a tree spanning all vertices is completed.

Graph and tree search algorithms

α–β · A* · B* · Backtracking · Beam · Bellman–Ford · Best-first · Bidirectional · **Borůvka** · Branch & bound · BFS · British Museum · D* · DFS · Depth-limited · Dijkstra · Edmonds · Floyd–Warshall · Fringe search · Hill climbing · IDA* · Iterative deepening · Johnson · Jump point · Kruskal · Lexicographic BFS · Prim · SMA*

Listings

Graph algorithms · *Search algorithms* · *List of graph algorithms*

Related topics

Dynamic programming · Graph traversal · Tree traversal · Search games

v · t · e

Contents [hide]

Pseudocode [edit]

Designating each vertex or set of connected vertices a "component", pseudocode for Borůvka's algorithm is:

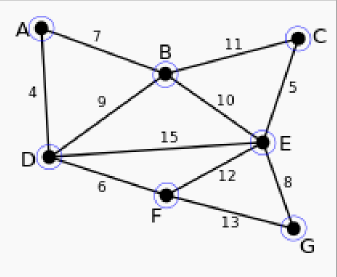
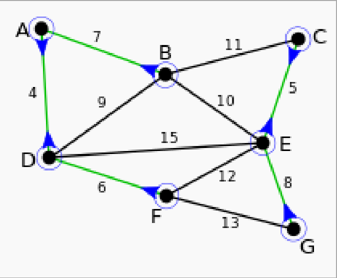
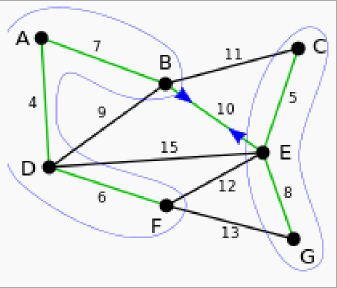
```
Input: A connected graph G whose edges have distinct weights
Initialize a forest T to be a set of one-vertex trees, one for each vertex of the graph.
While T has more than one component:
  For each component C of T:
    Begin with an empty set of edges S
    For each vertex v in C:
      Find the cheapest edge from v to a vertex outside of C, and add it to S
    Add the cheapest edge in S to T
Output: T is the minimum spanning tree of G.
```

As in Kruskal's algorithm, tracking components of T can be done efficiently using a disjoint-set data structure. In graphs where edges have identical weights, edges with equal weights can be ordered based on the lexicographic order of their endpoints.

Complexity [edit]

Borůvka's algorithm can be shown to take O(log *V*) iterations of the outer loop until it terminates, and therefore to run in time O(*E* log *V*), where *E* is the number of edges, and *V* is the number of vertices in *G*. In planar graphs, and more generally in families of graphs closed under graph minor operations, it can be made to run in linear time, by removing all but the cheapest edge between each pair of components after each stage of the algorithm.[7]

Example [edit]

Image	components	Description
 A B C D E F G 7 11 4 9 10 5 15 12 8 6 13	{A} {B} {C} {D} {E} {F} {G}	This is our original weighted graph. The numbers near the edges indicate their weight. Initially, every vertex by itself is a component (blue circles).
 A B C D E F G 7 11 4 9 10 5 15 12 8 6 13	{A,B,D,F} {C,E,G}	In the first iteration of the outer loop, the minimum weight edge out of every component is added. Some edges are selected twice (AD, CE). Two components remain.
 A B C D E F G 7 11 4 9 10 5 15 12 8 6 13	{A,B,C,D,E,F,G}	In the second and final iteration, the minimum weight edge out of each of the two remaining components is added. These happen to be the same edge. One component remains and we are done. The edge BD is not considered because both endpoints are in the same component.

Other algorithms [edit]

Other algorithms for this problem include Prim's algorithm and Kruskal's algorithm. Fast parallel algorithms can be obtained by combining Prim's algorithm with Borůvka's.[8]

A faster randomized minimum spanning tree algorithm based in part on Borůvka's algorithm due to Karger, Klein, and Tarjan runs in expected $O(E)$ time.[9] The best known (deterministic) minimum spanning tree algorithm by Bernard Chazelle is also based in part on Borůvka's and runs in $O(E\,\alpha(E,V))$ time, where α is the inverse of the Ackermann function.[10] These randomized and deterministic algorithms combine steps of Borůvka's algorithm, reducing the number of components that remain to be connected, with steps of a different type that reduce the number of edges between pairs of components.

Notes [edit]

1. ^ Borůvka, Otakar (1926). "O jistém problému minimálním (About a certain minimal problem)". *Práce mor. přírodověd. spol. v Brně III* (in Czech and German summary) **3**: 37–58.
2. ^ Borůvka, Otakar (1926). "Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí (Contribution to the solution of a problem of economical construction of electrical networks)". *Elektronický Obzor* (in Czech) **15**: 153–154.
3. ^ Nešetřil, Jaroslav; Milková, Eva; Nešetřilová, Helena (2001). "Otakar Borůvka on minimum spanning tree problem: translation of both the 1926 papers, comments, history". *Discrete Mathematics* **233** (1–3): 3–36. doi:10.1016/S0012-365X(00)00224-7. MR 1825599.
4. ^ Choquet, Gustave (1938). "Étude de certains réseaux de routes". *Comptes-rendus de l'Académie des Sciences* (in French) **206**: 310–313.
5. ^ Florek, Kazimierz (1951). "Sur la liaison et la division des points d'un ensemble fini". *Colloquium Mathematicum 2 (1951)* (in French): 282–285.

6. ^ Sollin, M. (1965). "Le tracé de canalisation". *Programming, Games, and Transportation Networks* (in French).
7. ^ Eppstein, David (1999). "Spanning trees and spanners". In Sack, J.-R.; Urrutia, J. *Handbook of Computational Geometry*. Elsevier. pp. 425–461.; Mareš, Martin (2004). "Two linear time algorithms for MST on minor closed graph classes" (PDF). *Archivum mathematicum* **40** (3): 315–320..
8. ^ 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.
9. ^ Karger, David R.; Klein, Philip N.; Tarjan, Robert E. (1995). "A randomized linear-time algorithm to find minimum spanning trees". *Journal of the ACM* **42** (2): 321–328. doi:10.1145/201019.201022.
10. ^ Chazelle, Bernard (2000). "A minimum spanning tree algorithm with inverse-Ackermann type complexity" (PDF). *J. ACM* **47** (6): 1028–1047. doi:10.1145/355541.355562.

Categories: Graph algorithms | Spanning tree