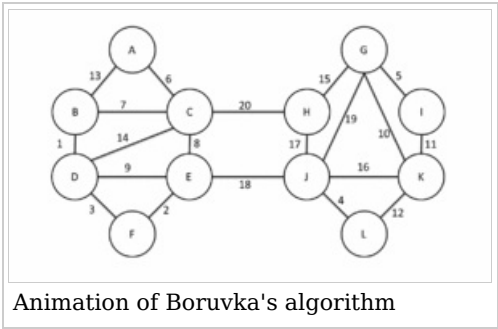


Borůvka's algorithm

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, or a minimum spanning forest in the case of a graph that is not connected.

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.



The algorithm begins by finding the minimum-weight edge incident to each vertex of the graph, and adding all of those edges to the forest. Then, it repeats a similar process of finding the minimum-weight edge from each tree constructed so far to a different tree, and adding all of those edges to the forest. Each repetition of this process reduces the number of trees, within each connected component of the graph, to at most half of this former value, so after logarithmically many repetitions the process finishes. When it does, the set of edges it has added forms the minimum spanning forest.

Contents

- 1 Pseudocode
- 2 Complexity
- 3 Example
- 4 Other algorithms
- 5 Notes

Pseudocode

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

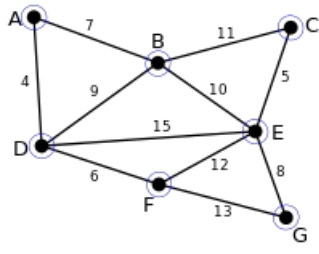
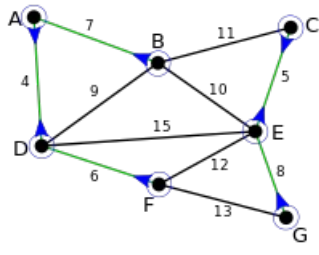
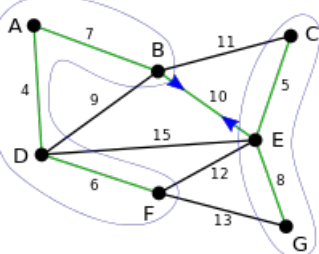
```
Input: A graph G whose edges have distinct weights
Initialize a forest F to be a set of one-vertex trees, one for each vertex of the graph.
While F has more than one component:
  Find the connected components of F and label each vertex of G by its component
  Initialize the cheapest edge for each component to "None"
  For each edge uv of G:
    If u and v have different component labels:
      If uv is cheaper than the cheapest edge for the component of u:
        Set uv as the cheapest edge for the component of u
      If uv is cheaper than the cheapest edge for the component of v:
        Set uv as the cheapest edge for the component of v
  For each component whose cheapest edge is not "None":
    Add its cheapest edge to F
Output: F is the minimum spanning forest of G.
```

If edges do not have distinct weights, then a consistent tie-breaking rule (e.g. breaking ties by the object identifiers of the edges) can be used. An optimization (not necessary for the analysis) is to remove from *G* each edge that is found to connect two vertices in the same component as each other.

Complexity

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

Image	components	Description
	{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,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}	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

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

- Borůvka, Otakar (1926). "O jistém problému minimálním" [About a certain minimal problem]. *Práce mor. přírodověd. spol. v Brně III* (in Czech and German). **3**: 37–58.
- 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.
- 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. MR 1825599 (https://www.ams.org/mathscinet-getitem?mr=1825599). doi:10.1016/S0012-365X(00)00224-7 (https://doi.org/10.1016%2FS0012-365X%2800%2900224-7).
- Choquet, Gustave (1938). "Étude de certains réseaux de routes". *Comptes-rendus de l'Académie des Sciences* (in French). **206**: 310–313.
- Florek, Kazimierz (1951). "Sur la liaison et la division des points d'un ensemble fini". *Colloquium Mathematicum* (in French). **2**: 282–285.
- Sollin, M. (1965). "Le tracé de canalisation". *Programming, Games, and Transportation Networks* (in French).
- 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" (http://www.emis.de/journals/AM/04-3/am1139.pdf) (PDF). *Archivum mathematicum*. **40** (3): 315–320..
- 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 (https://doi.org/10.1016%2Fj.jpdc.2006.06.001).
- 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 (https://doi.org/10.1145%2F201019.201022).
- Chazelle, Bernard (2000). "A minimum spanning tree algorithm with inverse-Ackermann type complexity" (http://www.cs.princeton.edu/~chazelle/pubs/mst.pdf) (PDF). *J. ACM*. **47** (6): 1028–1047. doi:10.1145/355541.355562 (https://doi.org/10.1145%2F355541.355562).

Retrieved from "https://en.wikipedia.org/w/index.php?title=Borůvka%27s_algorithm&oldid=792122702"

Categories: Graph algorithms | Spanning tree

-
- This page was last edited on 24 July 2017, at 16:15.
 - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.