CS 163 & CS 265: Graph Algorithms

Week 2: Spanning trees

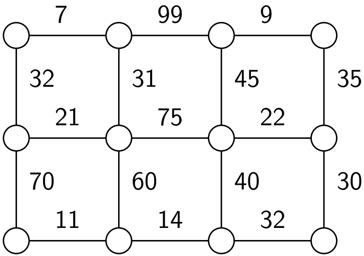
Lecture 2b: Minimum spanning tree algorithms

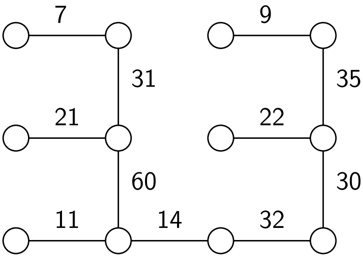
David Eppstein
University of California, Irvine

Winter Quarter, 2024

What we did last time

Minimum spanning tree problem definition





Input:

- Undirected graph
- Numbers ("weights") on its edges

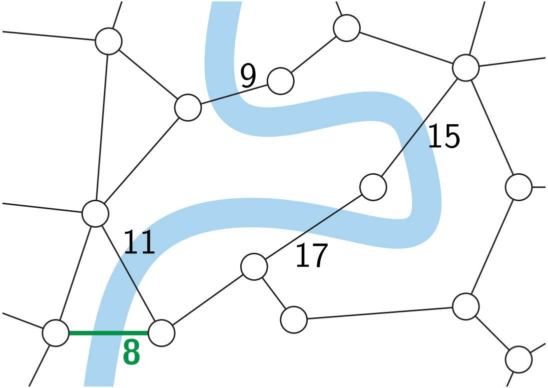
Output:

- Tree connecting all vertices
- Minimize total weight

Cut property (more useful in algorithms)

If we cut the vertices of the graph into any two subsets X and $G - X$, and e is the lightest edge with endpoints in both subsets

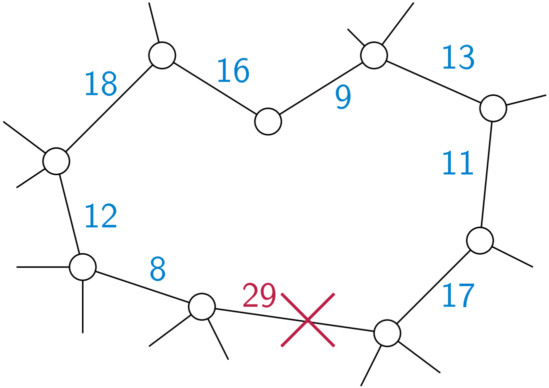
then e must be in in the MST



We can safely add it to the output

Cycle property (sometimes useful, less often)

If C is any cycle and e is its heaviest edge
then e cannot be in the MST



We can safely remove it from the graph

History

Three classical algorithms

Borůvka's algorithm (Otakar Borůvka, 1926)

- Rediscovered by Choquet (1938), Florek, Łukaszewicz, Perkal, Steinhaus, & Zubrzycki (1951), and Sollin (1965)
- Often called Sollin's algorithm

Jarník's algorithm (Vojtěch Jarník, 1930)

- Rediscovered by Prim (1957) and Dijkstra (1959)
- Often called Prim's algorithm or Prim–Dijkstra algorithm

Kruskal's algorithm (Joseph Kruskal, 1956)

- Rediscovered by Loberman & Weinberger (1957)
- Closely related to single-linkage hierarchical clustering (Florek et al. 1951; McQuitty 1957; Sneath 1957)

All use only the cut rule: they find edges to include by cutting vertices into subsets and choosing min-weight edge across the cut

All slower than linear time by a logarithm (similar to sorting)

Some other faster algorithms

Fredman & Willard 1994

- Use bit-manipulation operations on weights
- Linear time when weights are machine integers

Karger, Klein, & Tarjan 1995

- Randomized, linear expected time on any graph
- Only uses comparisons of weights
- Alternates between two methods: Borůvka-like (cut rule) reducing #vertices, and cycle rule reducing #edges

Chazelle 2000

- Non-random, very slightly non-linear (inverse Ackermann func.)

Pettie & Ramachandran 2002

- Non-random, optimal but unknown time complexity

An unsolved research question

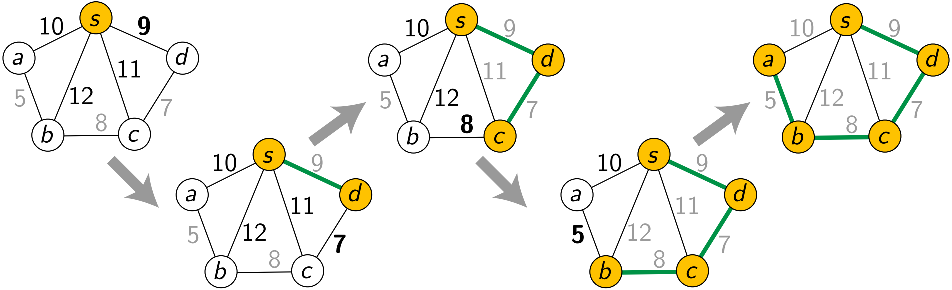
Is it possible for a non-random comparison-based algorithm to find minimum spanning trees in linear time?

Recent algorithms with small O-notation are too complicated to be practical so it would also be interesting to find the best complexity for a practical algorithm

Jarník's algorithm

Main idea of Jarník's algorithm

- Choose an arbitrary starting vertex s (this choice affects the steps of the algorithm but not its output)
- Build a tree T one edge at a time, starting with a one-vertex tree containing only s
- Repeat:
  - Partition the graph into two subsets T and $G - T$
  - Find the minimum-weight edge e connecting T to $G - T$
  - Add e and its endpoint in $G - T$ to the tree



Data structures for Jarník's algorithm

Root the tree T at s, and decorate each vertex v with its parent, so the tree edges are pairs v — parent[v]

When a vertex v is not yet in T, use parent[v] to store the minimum-weight edge connecting v to T

Maintain a priority queue Q of vertices that are not yet in T, prioritized by the weight of this connecting edge

Jarník pseudocode

```
def jarnik(G):
    let s be any vertex of G
    parent = {v : none for v in G}
    Q = priority queue of all vertices,
        priority = 0 for s, infinity for others

    while Q is non-empty:
        remove the minimum-priority vertex v from Q
        for each edge v-w in G:
            if w in Q and weight(v-w) < priority(w):
                parent[w] = v
                change priority of w to weight(v-w)
```

Jarník analysis

Two nested loops:

- Outer loop over all the vertices, in priority queue order
- Inner loop over the edges at that vertex

Each edge looped over twice (once for each endpoint) so total number of times through inner loop is $2m$

Except for priority queue, everything else is $O(m)$

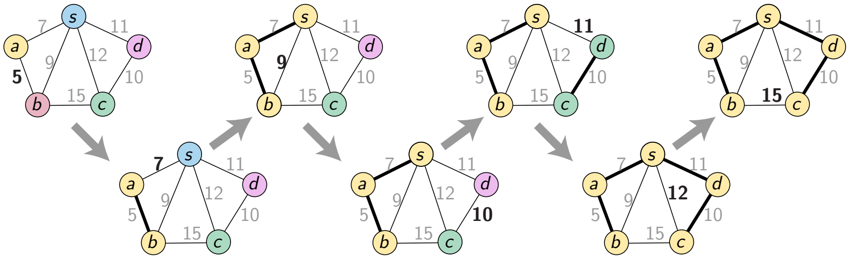
Priority queue:

- n find-and-remove operations, $\leq m$ reduce-priority operations
- With binary heap, all operations $O(\log n) \Rightarrow$ total $O(m \log n)$
- With Fibonacci heap [Fredman and Tarjan 1987], reduce-priority operations take only $O(1)$ time $\Rightarrow$ total $O(m + n \log n)$

Other classical algorithms

Kruskal's algorithm

- Start with a forest of one-vertex trees, one for each vertex
- Sort the edges from smallest to largest weight
- For each edge in sorted order, if it connects two different trees, add it to the forest

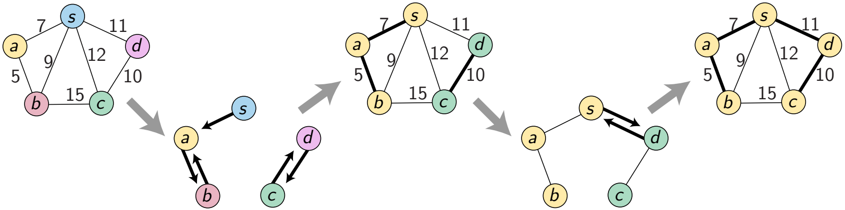


Analysis:

- Sorting: your favorite sorting algorithm
  $O(m \log m)$ for comparison sorting
- Testing for same tree: "union-find data structure", slightly more than constant per edge (much faster than sorting)

Borůvka's algorithm

- Start with a forest of one-vertex trees, one for each vertex
- While there is more than one tree:
  - Label vertices by their tree (connected components)
  - Assign graph edges to sets of edges that go out of each tree (two sets/edge), ignoring edges with both endpoints in one tree
  - Add to forest the minimum weight edge out of each tree



Analysis:

- Each time through while loop, #trees goes down by factor ≥ 2
- Assigning edges to sets takes linear time $\Rightarrow$ total $O(m \log n)$

A hybrid algorithm

- Run Borůvka's algorithm until the number of trees goes down from n to $\leq n/\log n$
- Then switch to Jarník's algorithm with Fibonacci heaps, using a priority queue on trees rather than on individual vertices

Analysis:

- Borůvka's algorithm repeats $\leq \log\log n$ times $\Rightarrow$ time $O(m \log\log n)$
- Jarník's algorithm takes $O(m + (n/\log n)\log(n/\log n)) = O(m)$
- Both together: $O(m \log\log n) + O(m) = O(m \log\log n)$

Morals of the story

We can compute minimum spanning trees efficiently in practice (comparable time to sorting) using several classical algorithms

Theoretical improvements are possible, including randomized linear expected time, but the best possible time for a non-random comparison-based algorithm remains an unsolved research problem

References and image credits I

Otakar Borůvka. O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti*, 3:37–58, 1926. URL `https://dml.cz/handle/10338.dmlcz/500114`.

Bernard Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the ACM*, 47(6):1028–1047, 2000. doi:`10.1145/355541.355562`.

Gustave Choquet. Étude de certains réseaux de routes. *Comptes Rendus de l'Académie des Sciences*, 206:310–313, 1938.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959. doi:`10.1007/BF01386390`. URL `https://www-m3.ma.tum.de/twiki/pub/MN0506/WebHome/dijkstra.pdf`.

Kazimierz Florek, Jan Łukaszewicz, Julian Perkal, Hugo Steinhaus, and Stefan Zubrzycki. Sur la liaison et la division des points d'un ensemble fini. *Colloquium Mathematicae*, 2(3–4): 282–285, 1951. doi:`10.4064/cm-2-3-4-282-285`. URL `https://eudml.org/doc/209969`.

Michael L. Fredman and Robert E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987. doi:`10.1145/28869.28874`.

References and image credits II

Michael L. Fredman and D. E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *Journal of Computer and System Sciences*, 48(3):533–551, 1994. doi:10.1016/S0022-0000(05)80064-9.

Steaphan Greene. SVG icon of two standard gaming dice. CC-BY-SA image, December 20 2008. URL https://commons.wikimedia.org/wiki/File:2-Dice-Icon.svg.

Vojtěch Jarník. O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti*, 6 (4):57–63, 1930. URL https://hdl.handle.net/10338.dmlcz/500726.

David R. Karger, Philip N. Klein, and Robert E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, 1995. doi:10.1145/201019.201022.

Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. doi:10.1090/S0002-9939-1956-0078686-7. URL https://www.jstor.org/stable/2033241.

References and image credits III

H. Loberman and A. Weinberger. Formal Procedures for connecting terminals with a minimum total wire length. *Journal of the ACM*, 4(4):428–437, October 1957. doi:10.1145/320893.320896.

Louis L. McQuitty. Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies. *Educational and Psychological Measurement*, 17(2):207–229, 1957. doi:10.1177/001316445701700204.

Seth Pettie and Vijaya Ramachandran. Minimizing randomness in minimum spanning tree, parallel connectivity, and set maxima algorithms. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 713–722, San Francisco, California, 2002. URL https://portal.acm.org/citation.cfm?id=545477.

Robert C. Prim. Shortest connection networks And some generalizations. *Bell System Technical Journal*, 36(6):1389–1401, November 1957. doi:10.1002/j.1538-7305.1957.tb01515.x. URL https://archive.org/details/bstj36-6-1389.

P. H. A. Sneath. The application of computers to taxonomy. *Microbiology*, 17(1):201–226, 1957. doi:10.1099/00221287-17-1-201.

Georges Sollin. Le tracé de canalisation. In Claude Berge and Alain Ghouila-Houri, editors, *Programming, Games, and Transportation Networks*. John Wiley & Sons, 1965.