Minimum Spanning Trees

MA252 Project

Ayaz Anis & Rathod Vijay Mahendra

May 20, 2020

IIT Guwahati

Summary

- Problem, Aim & Application.
- State of the Art and respective findings.
- Work focus:

    1. Cut property, Cycle property.
    2. Karger, Klein and Tarjan (1995) - Randomized - $O(E)$.
        - Boruvka's steps.
        - F - heavy edges.
        - Overall algorithm.
        - Time analysis.

- Conclusion.
- References.

Problem, Aim & Applications

- Problem Statement

  Given an undirected, weighted simple graph we attempt to find Minimum Spanning Tree (MST) or Minimum Spanning Forest (MSF).

- Aim

  The objective of this project is to learn and understand the state of art in Minimum Spanning Tree and to look if it is possible to extend ideas further.

- Applications

  1. Network Design
  2. Image Segmentation
  3. Single linkage clustering
  4. Broadcasting in computer networks

State Of The Art

- Boruvka's algorithm is the oldest minimum spanning tree algorithm, discovered by Boruvka in 1926 long before computers even existed - $O(E \log V)$.
- Prim's algorithm which was invented by Vojtech Jarnik in 1930 and rediscovered by Prim in 1957 and Dijkstra in 1959 - $O(E \log V)$
- Kruskal's algorithm was discovered by Joseph Kruskal in 1956 - $O(E \log V)$.
- In 1987 Fredman and Tarjan lowered the complexity to $O(E\ \beta(E,V))$, where $\beta(E,V)$ is the number of log-iterations to map V to number less than E/V. Soon after, the complexity was further reduced to $O(E \log\beta(E,V))$ by Gabow et al.
- Karger, Klein and Tarjan in 1995 found a linear time randomized algorithm based on a combination of Boruvka's algorithm along with an other algorithm.
- In 2000, a non-randomized comparison based algorithm was given by Bernard Chazelle - $O(E\alpha(E,V))$, based on the soft heap, an approximate priority queue, where α is extremely slow growing inverse of the Ackerman, and can be approximated as constant less than 4 for practical purposes.
- In 2002, Pettie & Ramachandran found a provably optimal deterministic comparison-based minimum spanning tree algorithm.The runtime of all steps in the algorithm is $O(E)$, *except for the step of using the decision trees*. Thus, this algorithm has the peculiar property that it is *provably optimal* although its runtime complexity is *unknown*.

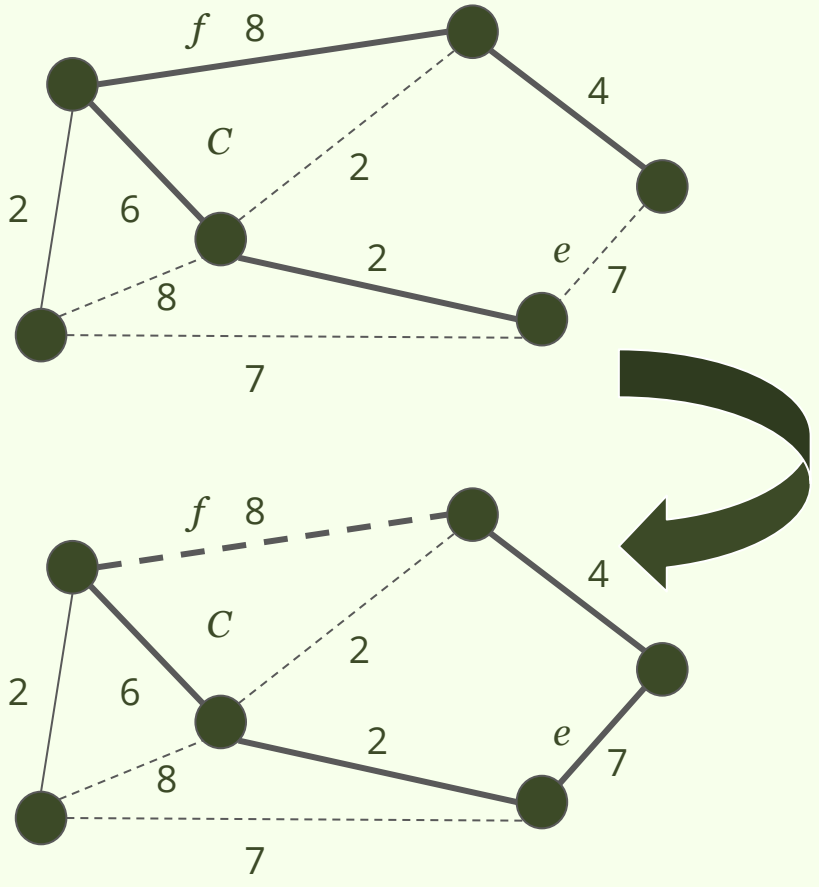
Cycle Property

Cycle Property:

- Let T be a minimum spanning tree of a weighted graph G.
- Let f be an edge of G that is not in T and C be the cycle formed by f with T.
- For every edge e of C, $weight(e) \leq weight(f)$.

Proof by contradiction:

- If $weight(f) > weight(e)$ we can get a spanning tree of smaller weight by replacing e with f.



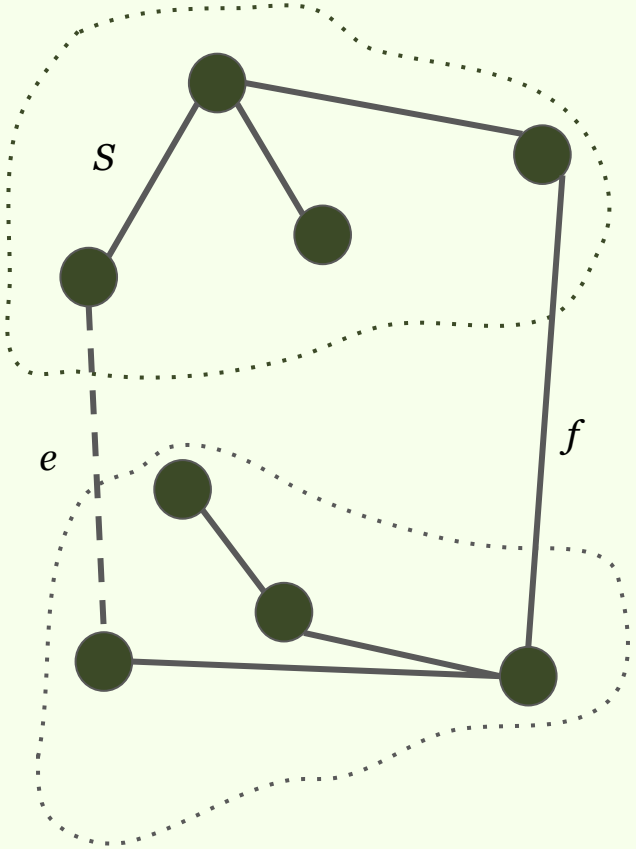
Cut Property

Cut Property:

- For any proper nonempty subset S of the vertices, the lightest edge with exactly one endpoint in S belongs to the minimum spanning forest/tree T.

Proof by Contradiction:

- Suppose e does not belong T, adding e to T creates a cycle C in T.
- Edge e is in both the cycle C and in the cutset D corresponding to $S \rightarrow$ there exists another edge, say f, that is in both C and D.
- $T^* = T \cup \{e\} - \{f\}$ is also spanning tree.
- Since, cost(T^*) < cost(T). Contradiction.



Karger, Klein & Tarjan's Algorithm

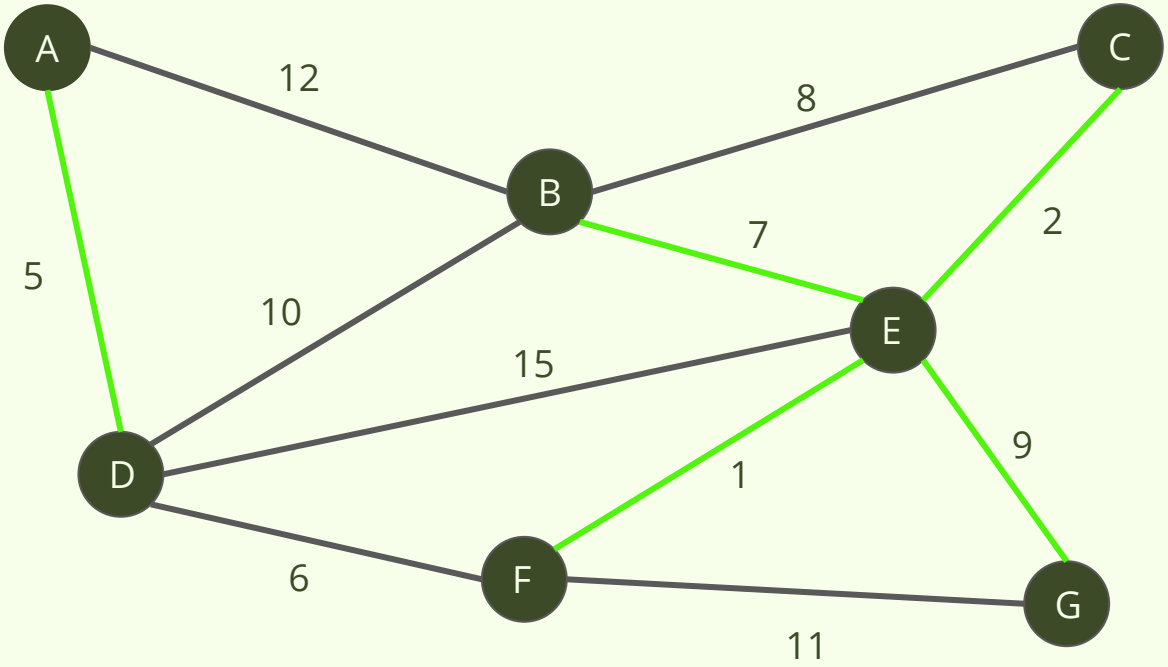
The Boruvka Step

Input: A graph G with no isolated vertices

1. For each vertex v, select the lightest edge incident on v.
2. Create a contracted graph G' by replacing each component of G connected by the edges selected in step 1 with a single vertex.
3. Remove all self-loops, and non-minimal repetitive edges from G'.

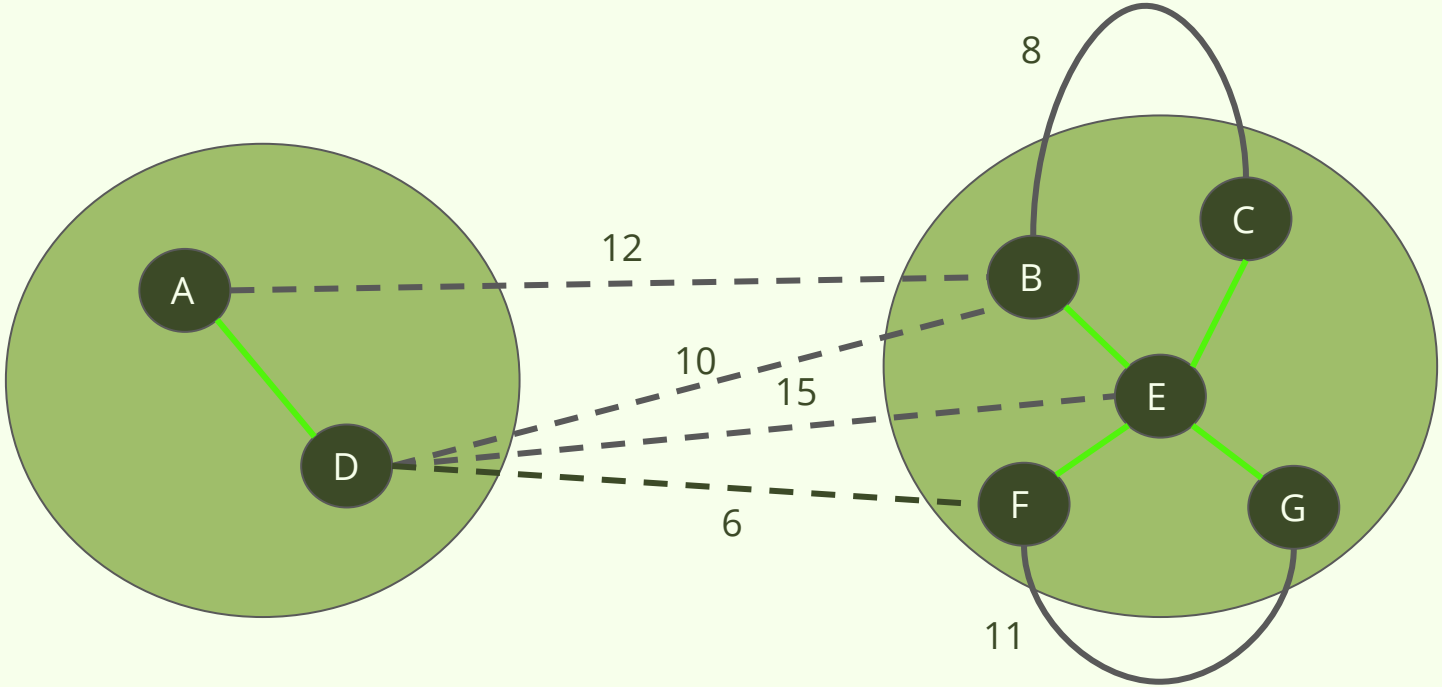
Output: The edges selected in step 1 and the contracted graph G'.

Execution Of Boruvka Step - 1



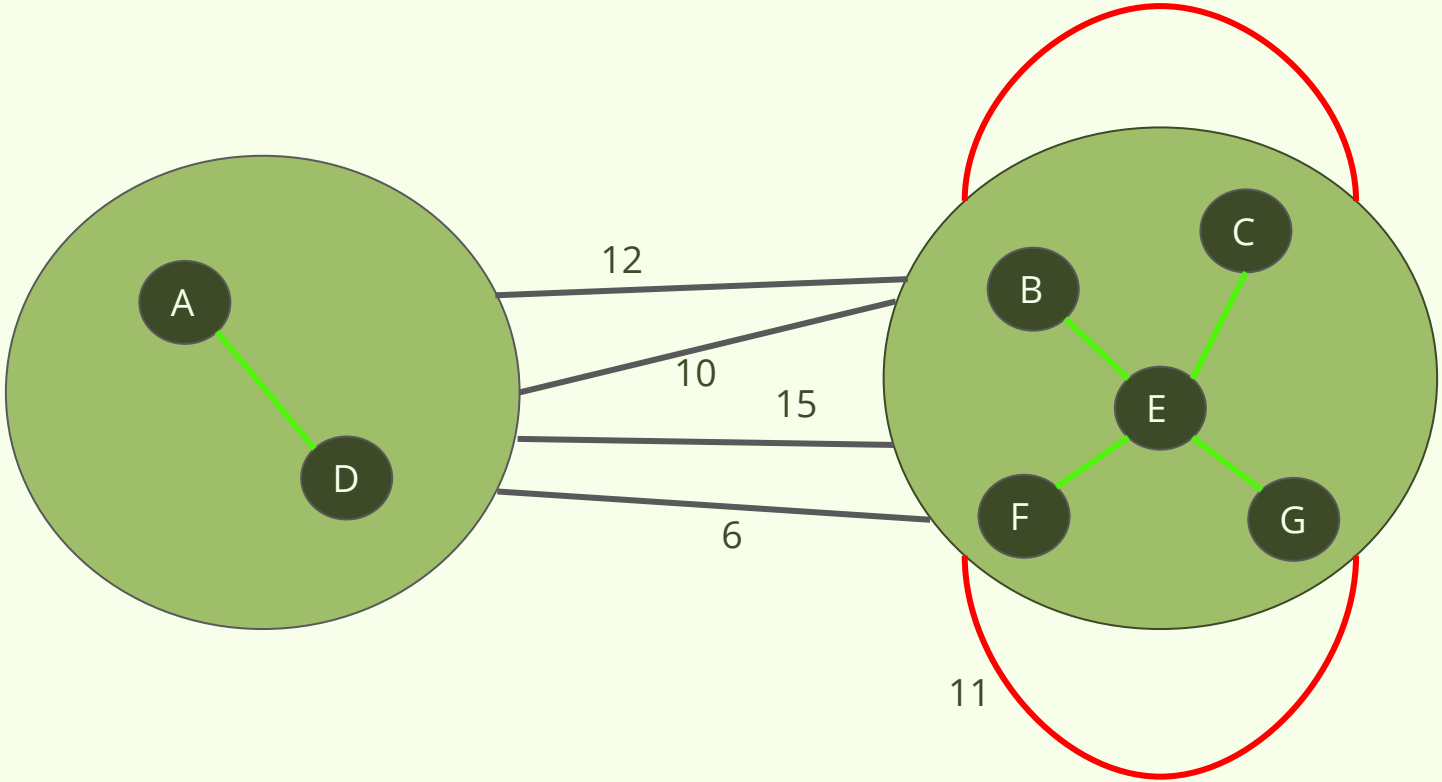
The lightest edge incident on each vertex is highlighted in green.

Execution Of Boruvka Step - 2



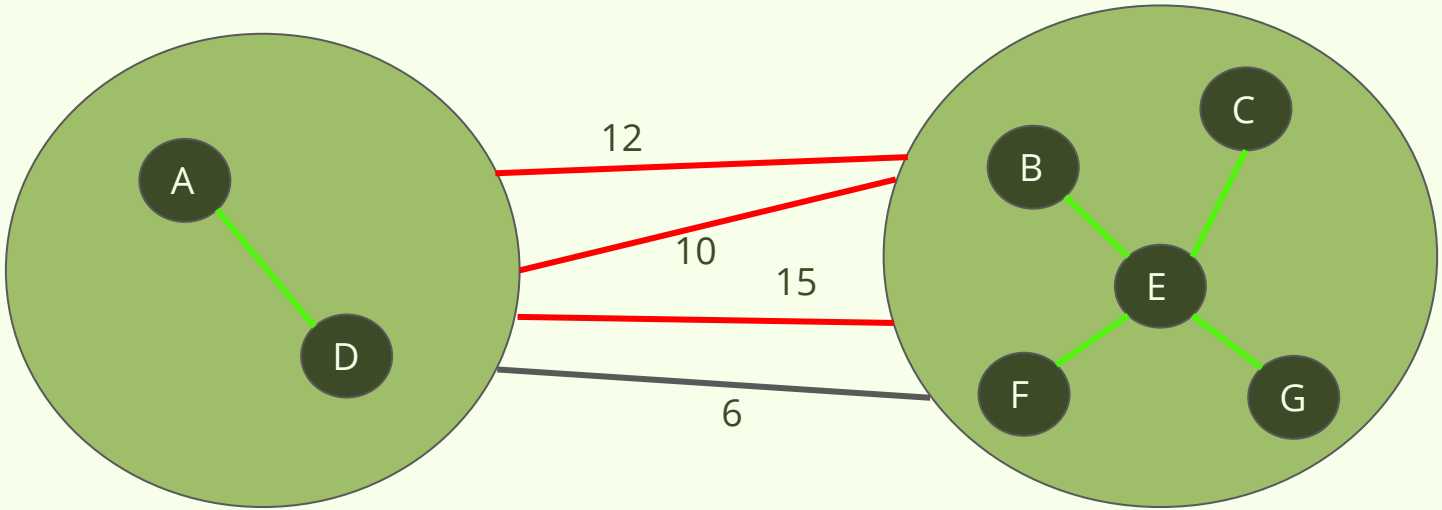
The graph is contracted and each component by the edges selected in step 1 is replaced by a single vertex. This creates two supernodes. All edges from the original graph remain.

Execution Of Boruvka Step - 3



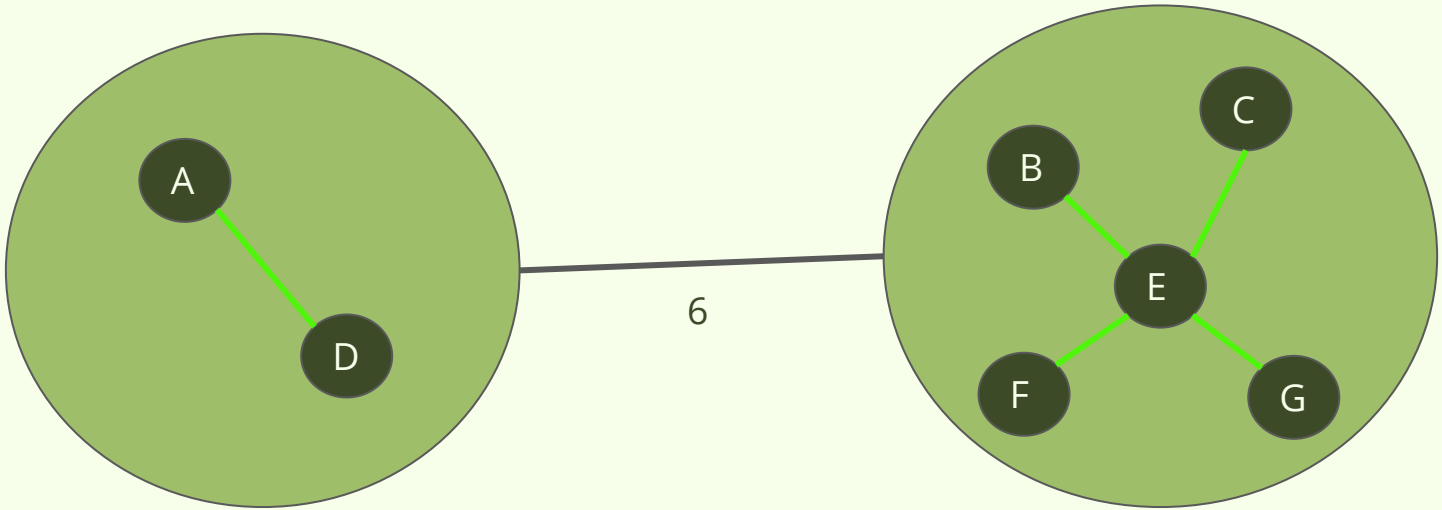
Edges that form self loops to the supernodes are deleted.

Execution Of Boruvka Step - 3



Non-minimal redundant edges between supernodes are deleted.

Execution Of Boruvka Result



The result of one Boruvka Step on the sample graph is a graph with two supernodes connected by a single edge.

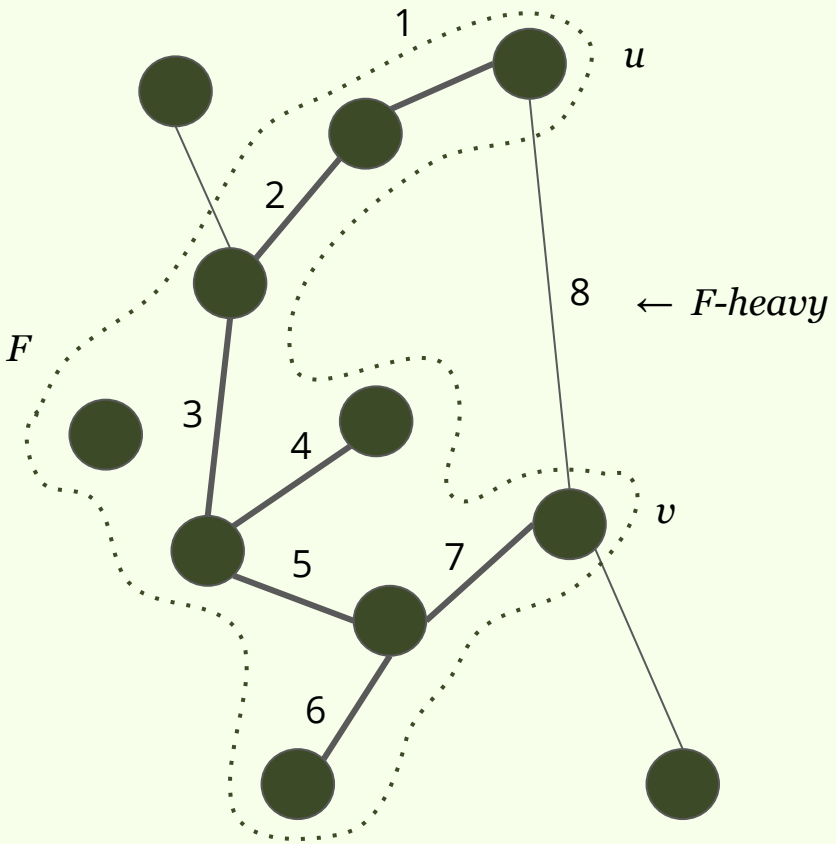
F - Heavy Edges

- The Heaviest edge on any cycle is not in the MST.
- Let F be a forest.
- Let $w_F(u,v)$ be the maximum weight of an edge on the path from u to v in F (or ∞ if the path does not exist.)
- Edge (u,v) is F-*heavy* if $w(u,v) > w_F(u,v)$ and F-*light* otherwise.

Claim:

- Let F be any forest, let (u,v) be any edge. If (u,v) is F-*heavy*, then (u,v) is not in the MST.

This claim is a consequence of the cycle property.



Overall Algorithm

Input: A graph G with no isolated vertices.

1. If G is empty return an empty set.
2. Create a contracted graph G' by running two successive Boruvka steps on G.
3. Create a subgraph H by selecting each edge in G' with probability 1/2. Recursively apply the algorithm to H to get its minimum spanning forest F.
4. Remove all F-heavy edges from G' (where F is the forest from step 3) using a linear time minimum spanning tree verification algorithm.
5. Recursively apply the algorithm to G' to get its minimum spanning forest.

Output: The minimum spanning forest of G' and contracted edges from Boruvka steps which together form MST.

J. Komlos in 1985 discovered linear time minimum spanning tree verification algorithm.

Time Analysis

Considering a single invocation of the algorithm

- Step 1 takes constant time = $O(1)$.
- Step 2 (Boruvka's) executes in time linear in the number of edges = $O(E)$.
- Step 3, i.e. creation of subgraph H and recursively applying the algorithm to it, takes $O(E)$.
- Step 4 executes in linear time using modified linear time minimum spanning tree verification algorithm.

Since the time required by one iteration of algorithm is linear in the number of edges, total cost of one complete run of this algorithm (including all recursive calls) is bounded by a constant factor times total number of edges in the original problem and recursive problems.

Here $\mathrm{E}[X]$ = Expectation of X.
E[Total no. of edges] = E[Total no. of edges in left subproblems] + E[Total no. of edges in right subproblems]
E[Total no. of edges] $\leq 2*E+V$
Hence expected running time = $O(2*E+V)$.
Since $V \leq 2*E$ for a graph with no isolated vertices, the algorithm runs in expected time $O(E)$.
Note: Its worst case running time is $O(\min\{V^2, E \log V\})$ which occurs with an exponentially small probability.

Conclusions

- In this report, we have seen that the algorithm given be Karger et al. runs in expected time $O(m)$ for a graph with m edges and n vertices, and we have also noted that in the worst case its runtime is $O(\min\{n^2, m\log n\})$ which occurs with an exponentially small probability. Hence in practical cases we can expect the algorithm to run in linear time except for a small fraction of cases.
- Minimum spanning trees (MST) finds its application in a range of fields. There have been quite a lot of algorithms for solving the MST problem, however, as practical problems are often quite large (road networks sometimes have billions of edges), performance is a key factor.
- Several researchers have been trying to find more computationally-efficient algorithms over the years. One of them is the algorithm given by Bernard Chazelle in 2000 which is the fastest non-randomized comparison-based algorithm (with known complexity) till date.
- In addition to providing a new complexity bound, Chazelle's work also contributes largely to the introduction of a non greedy approach to matroid optimization, which proves useful beyond minimum spanning trees.
- It remains an open problem whether a linear-time algorithm exists for finding a minimum spanning tree. Thus a trivial lower bound for the MST problem is $\Omega(m)$; and the best upper bound is $O(m\alpha(m, n))$

References

- 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.
- CHAZELLE, BERNARD 2000. A minimum spanning tree algorithm with inverse-Ackermann type complexity. Journal of the ACM, 47 (6): 1028–1047
- DIXON, B., RAUCH, M., AND TARJAN, R. E. 1992. Verification and sensitivity analysis of minimum spanning trees in linear time. SIAM J. Comput. 21, 1184– 1192.
- FREDMAN, M. L., AND TARJAN, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM 34, 596–615.
- GABOW, H. N., GALIL, Z., SPENCER, T., AND TARJAN, R. E. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica 6, 109–122.
- KARGER, D. R., KLEIN, P. N., AND TARJAN, R. E. 1995. A randomized linear-time algorithm to find minimum spanning trees. J. ACM 42, 321–328.
- KRUSKAL, J. B. 1956. On the shortest spanning subtree of a graph and the travelling salesman problem. Proc. Amer. Math. Soc., 7:48–50.
- PRIM., R. C. 1957. Shortest connection networks and some generalizations. Bell. Syst. Tech. J., 36:1389–1401.
- PETTIE, S., RAMACHANDRAN V. 2002. An optimal minimum spanning tree algorithm. J. ACM, 49:16–34

THANK YOU