The Minimum Spanning Tree Problem

Given a connected graph, find a spanning tree of minimum total edge cost.

where,

\[ n = \text{the number of vertices} \]

\[ m = \text{the number of edges} \]

\[ n - 1 \leq m \leq \binom{n}{2} \]
Applications

Network Construction

Clustering

Minimum Tour Relaxation (Held-Karp 1-trees)
A Simple Solution From the 80's
(with apologies to Oliver Stone)

Gorden Gecko: "Greed is Good"

Repeatedly select the cheapest unselected edge and add it to the tree under construction if it connects two previously disconnected pieces.

Kruskal, 1956
The greedy method generalizes to matroids.

We shall generalize the method rather than the domain of application.
Generalized Greedy Method

Beginning with all edges uncolored, sequentially color edges blue (accepted) or red (rejected).

Blue Rule:

Color blue any minimum-cost uncolored edge crossing a cut with no blue edges crossing.

Red Rule:

Color red any maximum-cost uncolored edge on a cycle with no red edges.

E MST with all blue, no red
Jarnik’s Algorithm

Grow a tree from a single start vertex.
At each step add a cheapest edge with exactly one end in the tree.
Boruvka's Algorithm

Repeat the following step until all vertices are connected:

For each blue component, select a cheapest edge connecting to another component; color all selected edges blue.

For correctness, a tie-breaking rule is needed.

Henceforth, assume all edge costs are distinct.

Then there is a unique spanning tree.
"Classical" Algorithms

(before algorithm analysis)

Kruskal’s algorithm, 1956

$O(m \log n)$ time

Jarnik’s algorithm, 1930

$O(n^2)$ time

also Prim, Dijkstra

Boruvka’s algorithm, 1926

$O(\min\{m \log n, n^2\})$ time

and many others
Selected History

Boruvka, 1926 \quad O(\min \{m \log n, n^2\})

Jarnik, 1930
Prim, 1957
Dijkstra, 1959 \quad O(n^2)

Kruskal, 1956 \quad O(m \log n)

Williams, Floyd, 1964
heaps \quad O(m \log n)

Yao, 1975
packets in Boruvka’s algorithm \quad O(m \log \log n)

Fredman, Tarjan, 1984
F-heaps in:
\quad Jarnik’s algorithm \quad O(n \log n + m)
\quad a hybrid Jarnik-Boruvka algorithm \quad O(m \log^* n)

Gabow, Galil, Spencer, 1984
Packets in F-T algorithm \quad O(m \log \log \log \ldots \log n \leq 1)

\log^* n = \min \{i \mid \log \log \log \ldots \log n \leq 1\}

where the logarithm is iterated i times
Models of Computation

We assume comparison of the two edge costs takes unit time, and no other manipulation of edge costs is allowed.

Another model:

bit manipulation of the binary representations of edge costs is allowed.

In this model,

Fredman-Willard, 1990, achieved $O(m)$ time.

(fast small heaps by bit manipulation)
Goal: An $O(m)$-time algorithm without bit manipulation of edge weights

Boruvka’s algorithm with contraction:

If $G$ contains at least two vertices:

select cheapest edge incident to each vertex;

Contract all selected edges;

Recur on contracted graph.

If contraction preserves sparsity ($m = O(n)$), this algorithm runs in $O(n) = O(m)$ time on sparse graphs.

E.g. planar graphs
How to handle non-sparse graphs?

Thinning: remove all but $O(n)$ edges by finding edges that can’t be in the minimum spanning tree.

How to thin?
Verification:

Given a spanning tree, is it minimum?

Thinning: Given a spanning tree, delete any non-tree edge larger than every edge on tree path joining its ends (red rule).

If all non-tree edges can be thinned, tree is verified.
History of Verification Algorithms

Tarjan, 1979 \quad O(m \alpha (m,n)) time

Komlos, 1984 \quad O(m) comparisons

Dixon, Rauch, Tarjan, 1992 \quad O(m) time

King, 1993 \quad O(m) time (simplified)

All these algorithms will thin.
Thinning by Random Sampling (1993)

Select half the edges at random.

Build a minimum spanning forest of the sample.

Thin.

How many edges remain?

Karger: $O(n \log n)$ on average

Klein, Tarjan: $< 2n$ on average
Minimum Spanning Forest Algorithm

If \# edges/\# vertices < 5, then

(Boruvka step) Select the cheapest edge incident to each vertex.

Contract all selected edges.

Recur on contracted graph.

Else

(Sampling and Thinning Step) Sample the edges, each with probability 1/2.

Construct a minimum spanning forest of the sample, recursively.

Thin using this forest.

Recur on Thinned Graph
Analysis

Boruvka step

\[ m < 5n \text{ implies } m' < 9m/10 \text{ since at least } n/2 \text{ edges are contracted} \]

\[ T(m) = O(m) + T(9m/10) \]

Thinning Step

\[ m > 5n \text{ implies } 2n < 2m/5 \]

\[ T(m) = O(m) + T(m/2) + T(2m/5) \]

where \( T(m/2) \) and \( T(2m/5) \) are expected time

\[ T(m) = O(m) \text{ by induction} \]
Bound on Number of Edges Not Thinned

Let $e_1, e_2, \ldots, e_m$ be the edges, in increasing cost.

Run the following variant of Kruskal’s algorithm.

Initialize $F = \emptyset$.

Process the edges in order.

To process $e_i$, flip a coin to see if $e_i$ is in the sample.

If $e_i$ forms a cycle with edges in $F$, discard it as thinned.

Otherwise, if $e_i$ is sampled, add $e_i$ to $F$. (Whether or not $e_i$ is sampled, it is not thinned.)

$F$ is the minimum spanning forest of the sample.
How many edges are not thinned?

The only relevant coin flips are those on unthinned edges, each of which has a chance of 1/2 of adding an edge to F (a success).

There can be at most n-1 successes.

For there to be more than k unthinned edges, the first k relevant coin flips must give at most n-2 successes.

The chance of this is at most

\[ \left( \frac{1}{2} \right)^k \sum_{i=0}^{n-2} \binom{k}{i} \left( \frac{1}{2} \right)^k \sum_{i=0}^{n} \binom{k}{i} \]

In particular, the average number of unthinned edges is at most 2n.
Preprocessing – Table Lookup

**Idea:** Given enough time (exponential or superexponential) one can build an optimum algorithm for a given problem in a given computational model, such as a decision tree. (The algorithm itself may be exponential in size.)

This means that sufficiently small (log or log-log size) subproblems can be solved optimally by table lookup using only linear preprocessing time.
Fast Divide and Conquer

Rapidly divide the problem into polylog-size subproblems. If combining time is linear, this yields a recursive $O(n \log^* n)$-time algorithm.

With table lookup to solve subproblems, the overall solution time can be reduced, possibly to linear. The algorithm becomes non-recursive: only $O(1)$ division steps are applied.
Overall approach:

Verification

each nontree edge:
cost as large as
max on tree path
Note:

This method can give algorithms optimal to within a constant factor \textit{without} offering a tight estimate of how fast they are.
Further Results

\( \Theta(\max(n) \log(n)) \rightarrow \Theta(\max(n)) \) deterministic

Chazelle: "soft" heaps

Optimal to within a constant factor

Pettie + Rama Chandran:

Chazelle + optimal on small subproblems
Open Problems

Deterministic $O(m)$?

Simpler verification?

Other applications?
  - directed spanning trees?
  - shortest paths?