Proof of Correctness for Prim's MST Algorithm

Robin Dawes

February 27, 2021

Presents an inductive proof that Prim's Agorithm successfully finds an MST for any connected graph with non-negative edge weights. This proof can be adapted to other algorithms.

.

The Long-Promised Proof

Here is Prim's Algorithm.

```
def Prim(G):                    # G is a connected graph with weighted edges

    choose any vertex v
    chosen_edges = {}
    T = {v}                     # T is the tree we are growing
    R = {all vertices except v} # R is the rest of the vertices

    while |T| < n: # keep going until T contains all vertices
        let e be the least-weight edge that has one end in T and one end in R
        suppose e = (x,y) with x in T and y in R
        add e to chosen_edges
        add y to T
        remove y from R

   return chosen_edges
```

Regardless of the implementation, the algorithm is based on the pseudo-code given above … but up to this point we have given no reason to accept that this algorithm does what it is supposed to. Does it actually find an MST of the graph?

Theorem: Prim's Algorithm finds an MST of G.

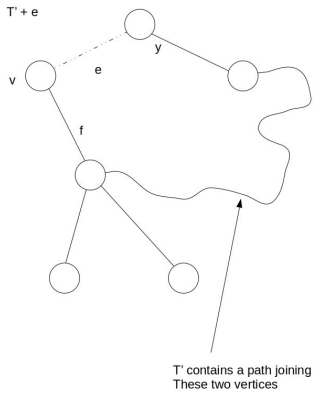
Proof:

Consider the situation before the first iteration. At this point the tree contains only the vertex v. The algorithm chooses the least-weight

edge that joins v to another vertex in the graph. We will now show that there is an MST that contains this edge. For this discussion we will call this edge e and we will call its other end-vertex y.

Let T' be any MST of the graph. If T' contains the edge e the claim is satisfied.

So suppose T' does not contain e. If we add the edge e to T', we get a graph that contains exactly one cycle, and the cycle must contain another edge (call it f) that has v as one of its end-vertices.

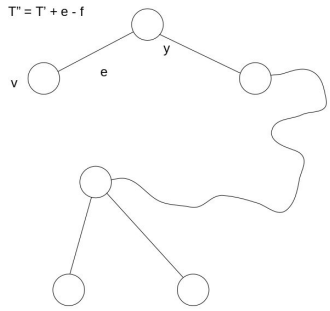


What do we know about f? We know the weight of f must be $\geq$ the weight of e ... we know this because the algorithm chooses e instead of f.

Consider $T'' = T' + e - f$. This consists of $n - 1$ edges and it has no cycles (by removing f we have broken the only cycle) ... and that means it is a tree.

Furthermore since it contains all the vertices, it is a spanning tree. Since we get from T' to T'' by removing an edge (f) and adding an edge (e) such that the edge we are adding has weight $\leq$ the weight of the edge we are removing, we see

$weight(T'') \leq weight(T')$



But since T' is an MST, $weight(T'')$ cannot be $< weight(T')$, so we have

$weight(T'') = weight(T')$

which means that T'' is also an MST, and it contains e.

Thus the claim is true – there is an MST of the graph that contains the first edge chosen.

We now continue with an induction-style proof.

Assume that for some $k \geq 1$, after the k^{th} iteration of the main loop the set of edges chosen so far is a subset of some MST. We need to show that this is still true after iteration $k + 1$.

The proof of this follows exactly the same structure as the proof for the base case. Let T be the (partial) tree constructed up to the end of iteration k. Let edge g be the edge chosen during iteration $k + 1$. We know that one end of g is in T and the other end is in R (the rest of the vertices). If there is an MST that contains T and also contains the edge

g, we are done.

So suppose $T + g$ is not contained in any MST. Let T' be a MST that contains T (we know T' exists). Consider $T' + g$. As in the base case, we know this contains a cycle, and the cycle contains the edge g. Suppose $g = (x, y)$ where $x \in T$ and $y \in R$. Then the cycle must contain another edge – call it h – with one end in T and the other end in R. The crucial observation is that the edge h could have been chosen during iteration $k + 1 \ldots$ but it wasn't $\ldots$ edge g was chosen instead. This means $w(g) \leq w(h)$.

So consider $T'' = T' + g - h$. As in the base case we see that T'' must be an MST, and it contains all the edges chosen by the algorithm up to and including iteration $k + 1$.

Thus by induction, we can claim:

"After each iteration, the set of chosen edges is a subset of the edges of an MST."

and when we reach $|T| = n$, the chosen $n - 1$ edges actually form an MST.

Now you have a template ... try to prove the correctness of Kruskal's MST Algorithm and Dijkstra's Shortest Path Algorithm.