Psuedo-code for Prim's algorithm for Minimum Spanning Tree

```
Input: weighted digraph G = (V,E,wt) assumed to be symmetric
Output: graph T = (V, ET) that is symmetric and has the
    same vertices/edges and ET is a MST of G

T := new graph with vertices V (and no edges)
r := choose some element of V
Q := new min-priority queue
link: array indexed by V of vertices
inQ := array that indicates if a vertex is in the queue
handles := array that keeps track of our handles to our data

forall v:
     link[v] := 0
     handles[v] := null
     inQ := 1

/* We write (v,d) for an object where the distance d is mutable.
 * Comparison in Q based on d. */

handles[0] := Q.enqueue((r,0))       /* distance 0 for r */
for all v in V with v!=r:
     handles[v] := Q.enqueue( (v, infinite))
while Q nonempty
     (v,d) := Q.dequeueMin()
     handles[v] := null
     inQ[v] := 0
     for all u in successors(G,v)
          if inQ[u] and wt(u,v) < distance of u then
              link[u] := v
              update distance of u to wt(u,v)
              Q.decreasedKey(handles[u]) /*signal to the queue that
              we decreased out key, so that the Queue re-orders it*/
forall v:
     add to T the edges (v,link[v]) and (link[v],v) //makes it symmetric


Invariants of main loop:
 A: the edges in T, and vertices in V — Q, are part of a MST
 B: the set { (v,link[v]) I v in V, v!=r, v not in Q } is a tree
    and is part of an MST of G
 C: for all (u,d) in Q,
       if link[u]!=0 then d < infinity
         and d is the weight of a lightest edge connecting u to the
         rest of the current tree
```