PRIM'S ALGORITHM for MST

Uses basic principle:
given a subtree $T$ of MST, the "lightest" edge connecting to a vertex not in $T$ can be added (greedily) to $T$. 
PRIM'S ALGORITHM for MST

Instead of growing a forest, grow a single tree $T$.

We would like to identify these black edges between vertices in $T$ & not in $T$ ...

... & pick the lightest.
**PRIM'S ALGORITHM** for MST

4 types of edges:
- \( \text{in } T \): between vertices of \( T \) \( \rightarrow \) inactive
- not in \( T \): between vertices of \( T \) \( \rightarrow \) inactive
- only 1 end in \( T \) \( \rightarrow \) next to be considered
- others not in \( T \)

All vertices not in \( T \) have a weight = lightest edge connecting to \( T \).

Then we could extract min-weight vertex & add it (w/ edge) to \( T \).
PRIM'S ALGORITHM for MST

For each neighbor \( v_i \) of \( x \)

- if \( v_i \) not in \( T \)
  - \( c = \text{weight}(v_i) \)
  - \( \text{weight}(v_i) \leftarrow \min(c, w(x, v_i)) \)

(some neighbors of \( x \) might lose weight)

Next we must update the set of vertices not in \( T \) (incl. weights)
PRIM'S ALGORITHM for MST

Recap
- maintain set of vertices neighboring current free $T$ (implicitly just maintain all not in $T$) w/ weights
- must be able to extract min & decrease values (= weight)
PRIM’S ALGORITHM for MST

extract min : $O(1)$
& update : $O(\log V)$

priority queue

Decrease key : $O(\log V)$
(swaps up)
PRIM'S ALGORITHM for MST

1) start w/ any vertex $s$; set $w(s) = 0$
2) set $w(\neq s) = \infty$ & put all in pr. queue
3) while pr. queue not empty
   x: extract-min & add edge to $T$
   mark $x \rightarrow$ in $T$.

("if $x = s$, no edge to add")
("add edge" $\rightarrow$ find an edge from $x \rightarrow T$, w/ min weight)
**PRIM’S ALGORITHM** for MST

1) start w/ any vertex \( s \); set \( w(s) = 0 \)

2) set \( w(\neq s) = \infty \) & put all in pr.queue

3) while pr.queue not empty

   - \( x: \text{extract-min} \) & add edge to \( T \)
   - mark \( x \to \) in \( T \).

   - for each unmarked neighbor \( v \) of \( x \)
     - if \( w(v) > w(v, x) \) then decrease.

\( N \) rounds
PRIM'S ALGORITHM for MST

1) start w/ any vertex \( s \); set \( w(s) = 0 \)
2) set \( w(\neq s) = \infty \) & put all in pr. queue
3) while pr. queue not empty
   \[ x: \text{extract-min} \] & add edge to \( T \)
   \[ \text{mark } x \to \text{ in } T. \]
   for each unmarked neighbor \( v \) of \( x \)
   if \( w(v) > w(v,x) \) then decrease.

\( V \) rounds

total \( O(E) \)

All: \( O(E \cdot \log V) \) for connected
**PRIM'S ALGORITHM** for MST

Using adj. matrix w/ weighted entries & no pr. queue

1) start w/ any vertex $s$; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr. queue
3) while pr. queue not empty
   $\exists v$ not in $T$
   - $O(V)\{x: \text{extract-min} \& \text{add edge to } T$
     - mark $x \rightarrow$ in $T$.
   - for each unmarked neighbor $v$ of $x$
     - if $w(v) > w(v,x)$ then decrease.

$O(V^2)$ time & space
Final comments:

Both algorithms, by Kruskal and Prim, can be improved with more advanced data structures. This is discussed in CLRS briefly, but is beyond the scope of this class.