PRIM’S ALGORITHM for MST
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 $T$. 
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

all vertices not in T have a weight = lightest edge connecting to T.
PRIM'S ALGORITHM for MST

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

4 types of edges
\[
\begin{cases}
\text{in } T \\
\text{not in } T; \text{ between vertices of } T \rightarrow \text{ inactive} \\
\text{not in } T; \text{ only 1 end in } T \rightarrow \text{ next to be considered} \\
\text{others not in } T
\end{cases}
\]

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

- All vertices not in T have a weight $= \text{lightest edge connecting to } T$.
- Then we could extract min-weight vertex & add it (w/ edge) to T.

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

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

if \( v_i \) not in \( T \)

\[
\text{weight}(v_i) \leftarrow \min (c, w(x,v_i))
\]

(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

For each neighbor $v_i$ of $x$

\[
\text{if } v_i \text{ not in } T \quad \begin{align*}
    c &= \text{weight}(v_i) \\
    \text{weight}(v_i) &\leftarrow \min(c, w(x,v_i))
\end{align*}
\]

(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
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
PRIM'S ALGORITHM for MST

priority queue
PRIM'S ALGORITHM for MST

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

priority queue
Prim’s Algorithm for MST

- Extract min: $O(1)$ & update: $O(\log V)$
- Priority queue
- Decrease key: $O(\log V)$ (swap up)
PRIM'S ALGORITHM for MST

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

\[ \text{\# rounds} = \lvert V \rvert \]
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 $x \to$ in $T$.

(if $x=s$, no edge to add)

("add edge" -> find an edge from $x$ to $T$, w/ min weight)
PRIM'S ALGORITHM for MST

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

$V$ rounds

total $O(E)$
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 $x$ in $T$
   - for each unmarked neighbor $v$ of $x$
     - if $w(v) > w(v, x)$ then decrease.

$N$ rounds

\[ O(E) \text{ total} \]

\[ O(\log V) \text{ per round} \]
**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 $T$ for each unmarked neighbor $v$ of $x$
   - if $w(v) > w(v,x)$ then decrease

$V$ rounds

total $O(E)$

$\text{All: } O(E \cdot \log V)$ for connected
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.
   for each unmarked neighbor $v$ of $x$
   if $w(v) > w(v,x)$ then decrease.
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 array
3) while pr. queue not empty $\exists v$ not in $T$

scan array

scan row($x$) in matrix

$x$: extract-min & add edge to $T$
mark $x \rightarrow$ in $T$.

for each unmarked neighbor $v$ of $x$
if $w(v) > w(v, x)$ then decrease.
PRIM'S ALGORITHM for MST

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

\[ O(V^2) \text{ time & space} \]

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 \( T \)

\[ \text{scan array} \]

\[ \text{scan row}(x) \text{ in matrix} \]

\[ \text{O}(V) \{ x: \text{extract-min} \} \text{ & add edge to } T \]

\[ \text{mark } x \Rightarrow \text{ in } T. \]

\[ \text{for each unmarked neighbor } v \text{ of } x \]

\[ \text{if } w(v) > w(v, x) \text{ then decrease.} \]
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.