**Prim's Algorithm** (finds Min. Spanning Tree of Connected, weighted, undirected graph)

**Input:** Graph $G=(V,E)$ means Vertices $V$ and Edges $E$  
**Output:** Graph $G=(V_{new}, E_{new})$ $\leftarrow$ Minimum Spanning Tree

1) Pick an arbitrary vertex from the graph and add it to a new vertex set $V_{new}$
2) Take minimum weighted edge $(u,v)$ such that $u\in V_{new}$ and $v\in V\setminus V_{new}$ add $v$ to $V_{new}$, add $(u,v)$ to $E_{new}$.
3) Repeat step 2 until all vertices are in $V_{new}$

**Example:** $G=(V,E)$  

$V=\{A,B,C,D,E\}$  

Run Prim on $G$:

1) Pick arbitrary vertex. Let's take $A$.  

$V_{new}=\{A\}$  
$E_{new}=\{(A,B)\}$  

2) Select smallest edge from $V_{new}$ to $V\setminus V_{new}$. i.e. smallest edge leaving $A$. This is edge $(A,B)$ with cost 1.

and add $B$ to $V_{new}$: 

$V_{new}=\{A,B\}$

Repeat 2) Select smallest edge leaving $V_{new}$ going to $V\setminus V_{new}$.  

This is edge $(A,E)$ with cost 2. 

$E_{new}=\{(A,B),(A,E)\}$  
$V_{new}=\{A,B,E\}$

Repeat 2) Least edge from $V_{new}$ to $V\setminus V_{new}$. This is $(E,D)$, cost 1  

$E_{new}=\{(A,B),(A,E),(E,D)\}$  
$V_{new}=\{A,B,D,E\}$

Repeat 2) Need edge going to $C$. $(A,C),(B,C),(D,C)$ all have cost 3. Any of these are ok. W.I.O.G. Take $(A,C)$, cost 3

$E_{new}=\{(A,B),(A,E),(E,D),(A,C)\}$  
$V_{new}=\{A,B,C,D,E\}$ $\rightarrow$ Stop!
Define $T = (V_{mw}, E_{mw})$.  
$T$ is a (not unique!) MST.

**Kruskal's Algorithm**

Finds MST of connected weighted graph $G = (V,E)$.

1) Create forest $F$ where each vertex is a tree.

i.e., if $G = (V,E)$ w/ $V = \{A,B,C,\ldots\}$, $F = (\{A,B,C,\ldots\}, \emptyset)$

2) Let $S = E = \{\text{edges}\}$

3) While ($|S| > 0$ and $F$ not spanning):
   
a) take smallest edge from $S$
   
b) if that edge connects two different trees in $F$, add it to $F$.

**Example:**

$G:$

![Graph](image)
1) $F$: \[ \text{F is forest of nodes.} \]


3) a) Take smallest edge from $S \rightarrow (C,D)$, cost 3.

   $F = (\{A,B,C,D,E\}, \{(C,D)\})$

   b) $F$: \[ \text{Repeat: a) Take smallest edge from } S \rightarrow (A,D) \text{ cost 5} \]

   \[ \text{b) } F: \]
Repeat: a) Take smallest edge from $S \rightarrow (E, D)$ cost 6

b)

Repeat: a) Take smallest edge from $S \rightarrow (A, E)$ cost 6

b) Wait! $(A, E)$ doesn't connect two disjoint parts of $F$: causes a cycle! Don't add it to $F$!

Repeat: a) Take smallest edge from $S \rightarrow (B, C)$ cost 7

b)

Now we stop, $F$ is spanning!
**Dijkstra's Algorithm**: Finds single-source shortest paths i.e., starting at a given node, finds shortest path to all other nodes in graph.

1) Pick starting node, set distance to 0. Assign distance $\infty$ to all other nodes.
2) Mark all nodes unfinished.
3) Calculate tentative distances to all neighbors of current node. Compare to shortest distance yet seen and keep smaller of the two.
4) Mark current node finished.
5) Select unfinished node with smallest tentative distance, make it current.
6) Stop when all nodes are finished.

**Example**: $G = (V, E)$

$V = \{A, B, C, D, E, F\}$

$E = \{(A, B), (A, D), (A, E), (A, F), (B, C), (B, D), (C, D), (C, F), (E, F)\}$

![Graph Diagram]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tentative Distances</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>$\infty$</td>
<td>7</td>
<td>13</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>13</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>13</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>13</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>