

# MST - Prim's Algorithm

- “Grows” an MST by repeatedly finding the lowest cost edge that will connect a new vertex to MST
- For every vertex  $v$ , assume we have initialized  $v.\text{known} = \text{false}$  and  $v.\text{dist} = \text{infinity}$ .

Choose any vertex as the starting point  $s$ ;

```
s.dist = 0;
```

```
PQ.enqueue(s, 0);
```

```
while PQ is not empty
```

```
{
```

```
    v = PQ.dequeue();
```

```
    if(!v.known)
```

```
    {
```

```
        v.known = true;
```

```
        for (each vertex w adjacent to v)
```

```
            if((w.dist > A[v][w]) && (!w.known))
```

```
            {
```

```
                w.dist = A[v][w];
```

```
                w.path = v;
```

```
                PQ.enqueue(w, w.dist);
```

```
            }
```

```
    }
```

```
}
```

# MST - Kruskal's Algorithm

- Generates connected components that are eventually joined together.
- Continually select edges in order of smallest weight: accept an edge if it does not cause a cycle
- Assume there are  $M$  edges and  $N$  vertices in the graph, and graph is connected

```
Sort the list of edges  $E$  in increasing order;
MST = empty;
edgesAccepted = 0;
i = 0;
while(edgesAccepted < N-1)
{
    Suppose  $(u,v)$  is the  $i$ 'th edge in  $E$ 
    if (no path from  $u$  to  $v$  in MST) // diff. components
    {
        add  $(u,v)$  to MST;
        edgesAccepted++;
    }
    i++;
}
```