# #39: Prim's Algorithm

December 3, 2018 · *Wade Fagen-Ulmschneider*

A **Minimum Spanning Tree** is a spanning tree with the **minimal total edge weights** among all spanning trees.

- Every edge must have a weight
  - The weights are unconstrained, except they must be additive *(eg: can be negative, can be non-integers)*
- Output of a MST algorithm produces G':
  - G' is a spanning graph of G
  - G' is a tree

G' has a minimal total weight among all spanning trees. *There may be multiple minimum spanning trees, but they have equal total weight!*

- We covered the first classical algorithm (Kruskal) already!

---

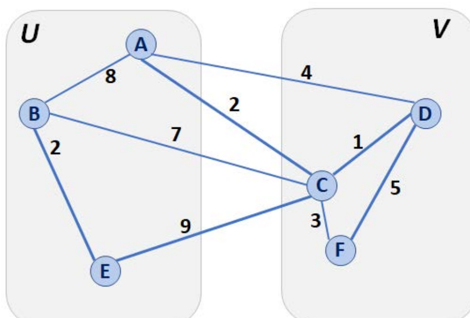
## Partition Property

Consider an arbitrary partition of the vertices on **G** into two subsets **U** and **V**.

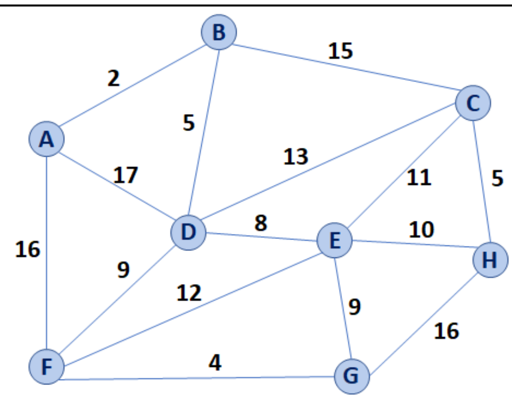
Let **e** be an edge of minimum weight across the partition.

Then **e** is part of some minimum spanning tree.

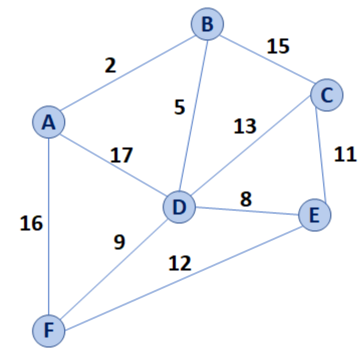
*Proof in CS 374!*




---




## Prim's MST Algorithm




| Pseudocode for Prim's MST Algorithm |
|---|

```
PrimMST(G, s):
  Input: G, Graph;
         s, vertex in G, starting vertex of algorithm
  Output: T, a minimum spanning tree (MST) of G

  foreach (Vertex v : G):
    d[v] = +inf
    p[v] = NULL
  d[s] = 0

  PriorityQueue Q   // min distance, defined by d[v]
  Q.buildHeap(G.vertices())
  Graph T           // "labeled set"

  repeat n times:
    Vertex m = Q.removeMin()
    T.add(m)
    foreach (Vertex v : neighbors of m not in T):
      if cost(v, m) < d[v]:
        d[v] = cost(v, m)
        p[v] = m

  return T
```

| | **Adj. Matrix** | **Adj. List** |
|---|---|---|
| **Heap** | | |
| **Unsorted Array** | | |

## Running Time of MST Algorithms

| Kruskal's MST | Prim's MST |
| --- | --- |
| | |

**Q:** What must be true about the connectivity of a graph when running an MST algorithm?

...what does this imply about the relationship between **n** and **m**?

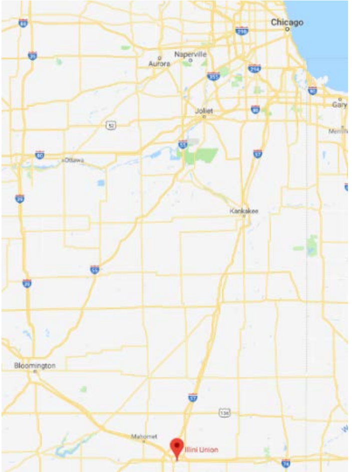
| Kruskal's MST | Prim's MST |
| --- | --- |
| | |

**Q:** Suppose we built a new heap that optimized the decrease-key operation, where decreasing the value of a key in a heap updates the heap in amortized constant time, or O(1)*. How does that change Prim's Algorithm runtime?

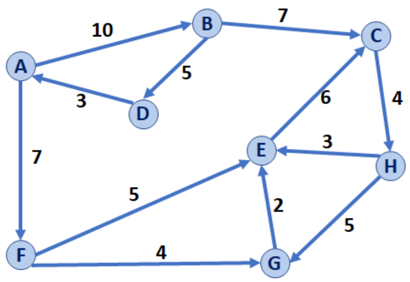
## Final big-O Running Times of classical MST algorithms:

| Kruskal's MST | Prim's MST |
| --- | --- |
| | |

## Shortest Path Home:



## Dijkstra's Algorithm (Single Source Shortest Path)




## Dijkstra's Algorithm Overview:

- The overall logic is the same as Prim's Algorithm
- We will modify the code in only two places – both involving the update to the distance metric.
- The result is a directed acyclic graph or DAG

| CS 225 – Things To Be Doing: |
| --- |
| 1. **Programming Exam C is on-going**<br>Exam: Sunday, Dec 2 – Tuesday, Dec 4<br>2. **MP7 Released – Slightly different structure:**<br>**Hard Deadline** on Monday, Dec. 3 (TONIGHT) for Part 1<br>EC Due on Wednesday, Dec. 5 (combing story)<br>3. lab_finale in lab this week!<br>4. Daily POTDs are ongoing for +1 point /problem |