CS 225

#39: Dijkstra's Algorithm

December 2, 2020 · *G Carl Evans*

Kruskal's Running Time Analysis

We have multiple choices on which underlying data structure to use to build the Priority Queue used in Kruskal's Algorithm:

Priority Queue Implementations:	Heap	Sorted Array
Building :6-8		
Each removeMin :13		

Based on our algorithm choice:

Priority Queue Implementation:	Total Running Time
Heap	
Sorted Array	

xBest Running Time of MST Algorithms (so far):

Kruskal's MST	Prim's MST
$O(n + m \lg(n))$	$O(n \lg(n) + m \lg(n))$

...however, we know that, for an MST algorithm, the graph must be at least minimally connected. This means there must be at least one edge on every vertex. The number of edges must be:

$$n - 1 \leq m \leq n(n-1) / 2$$

...or in terms of big-O...

$$O(n) \leq O(m) \leq O(n^2)$$

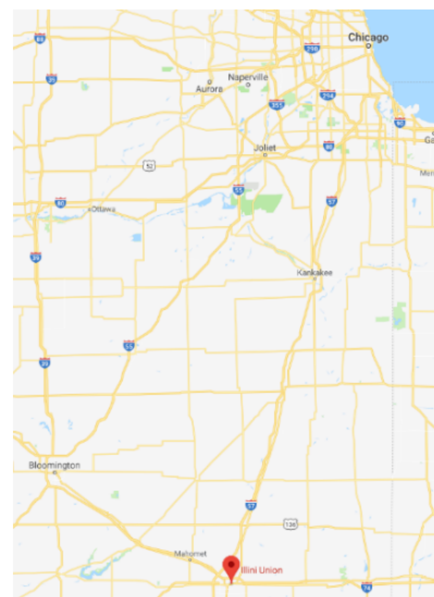
Using the fact they are connected:

Kruskal's MST	Prim's MST
Sparse Graph ($m = n$): **Dense Graph ($m = n^2$):** **All Graphs:**	**Sparse Graph ($m = n$):** **Dense Graph (m = n2):** **All Graphs:**

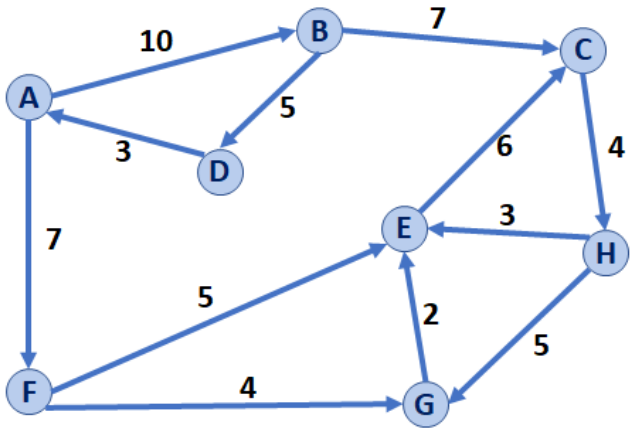
Q: Suppose we built a new heap that optimized the decrease-key operation, where decreasing the value of a key in a heap updates the heap in amortized constant time, or O(1)*. How does that change Prim's Algorithm runtime?

Kruskal's MST	Prim's MST

Shortest Path Home:



Dijkstra's Algorithm (Single Source Shortest Path)



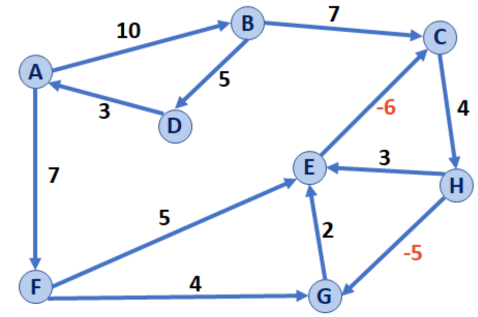
Dijkstra's Algorithm Overview:

- 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

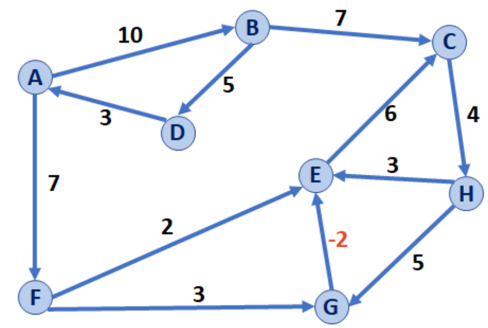
Pseudocode for Dijkstra's SSSP Algorithm

```
DijkstraSSSP(G, s):
  Input: G, Graph;
         s, vertex in G, starting vertex of algorithm
  Output: T, DAG w/ shortest paths (and distances) to s

  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 ________________________ < d[v]:
        d[v] = _________________________
        p[v] = m

  return T
```

Dijkstra: What if we have a negative-weight cycle?



Dijkstra: What if we have a minimum-weight edge, without having a negative-weight cycle?



Dijkstra makes an assumption:

Dijkstra: What is the running time?

CS 225 – Things To Be Doing:

1. **Final Project Check Points this week.**
2. lab_flow starts today
3. Daily POTDs are ongoing for +1 point /problem