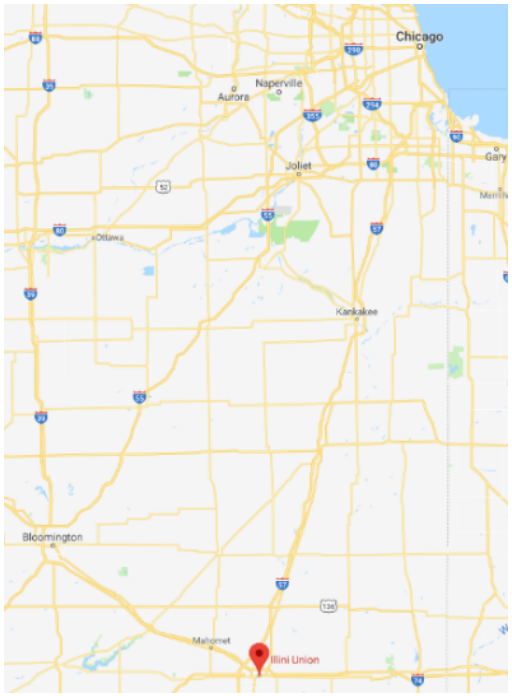
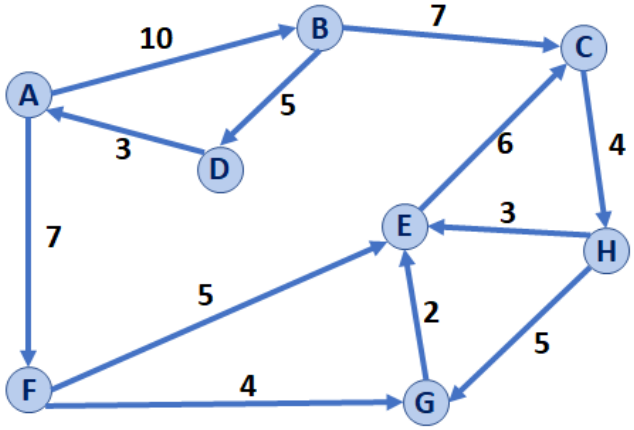
#41: Single Source Shortest Path

Shortest Path:



Dijkstra's Algorithm (Single Source Shortest Path)



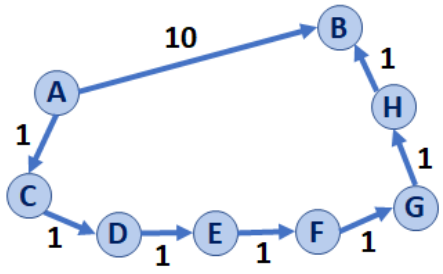
Dijkstra's Algorithm Overview:

- 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

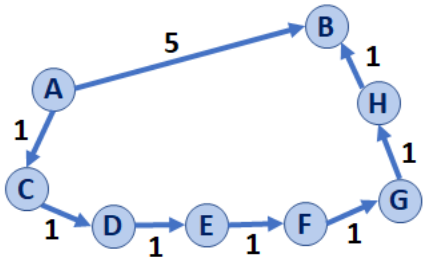
Pseudocode for Dijkstra's SSSP Algorithm

```
1  DijkstraSSSP(G, s):
2    Input: G, Graph;
3           s, vertex in G, starting vertex of algorithm
4    Output: T, DAG w/ shortest paths (and distances) to s
5
6    foreach (Vertex v : G.vertices()):
7      d[v] = +inf
8      p[v] = NULL
9    d[s] = 0
10
11   PriorityQueue Q   // min distance, defined by d[v]
12   Q.buildHeap(G.vertices())
13   Graph T           // "labeled set"
14
15   repeat n times:
16     Vertex m = Q.removeMin()
17     T.add(m)
18     foreach (Vertex v : neighbors of m not in T):
19       if _____________________ < d[v]:
20         d[v] = _____________________
21         p[v] = m
22
23   return T
```

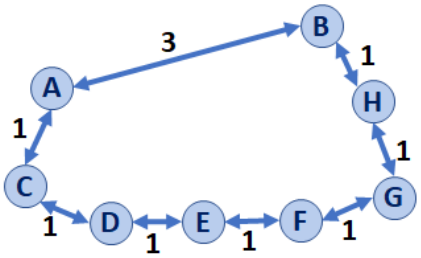
Dijkstra: One heavy-weight edge vs. faster light-weight edges?



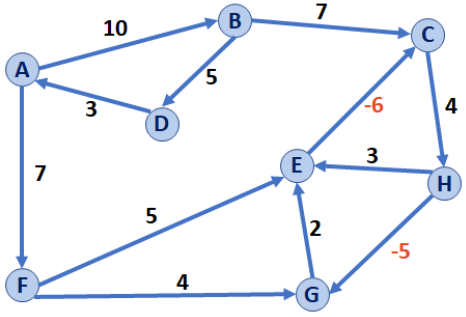
Dijkstra: One medium-weight edge vs. many light-weight edges?



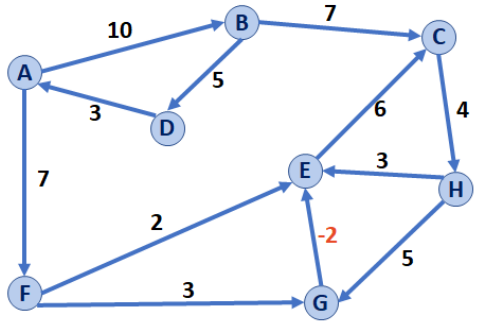
Dijkstra: Undirected graphs?



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



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



…what assumption does Dijkstra's algorithm make?

Dijkstra: What is the running time?

Landmark Path Problem: Best path to G from A, stopping at L along the way?