CS 225

Data Structures

December 5 – Dijkstra's Algorihtm

Wade Fagen-Ulmschneider

MST Algorithm Runtime:

We know that MSTs are always run on a minimally connected graph:

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

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

MST Algorithm Runtime:

- Kruskal's Algorithm:

  O(n + m lg(n))

Sparse Graph:

Dense Graph:

- Prim's Algorithm:

  O(n lg(n) + m lg(n))

Sparse Graph:

Dense Graph:

Suppose I have a new heap:

	Binary Heap	Fibonacci Heap
Remove Min	O(lg(n))	O(lg(n))
Decrease Key	O(lg(n))	O(1)*

What's the updated running time?

```
    PrimMST(G, s):
 6    foreach (Vertex v : G):
 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 cost(v, m) < d[v]:
20          d[v] = cost(v, m)
21          p[v] = m
```

MST Algorithm Runtimes:

- Kruskal's Algorithm:

  O(m lg(n))

- Prim's Algorithm:

  O(n lg(n) + m lg(n))

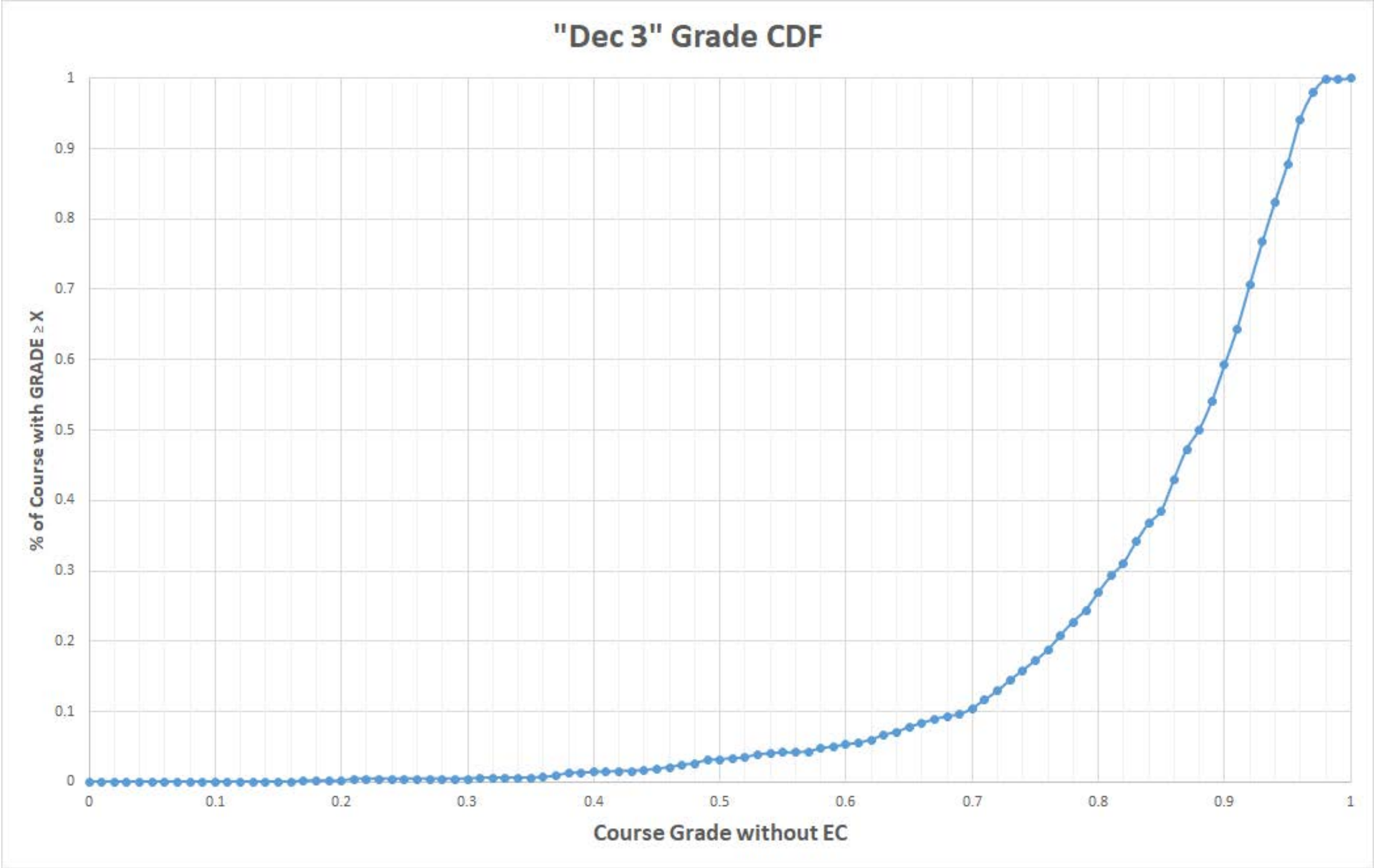
Final Big-O MST Algorithm Runtimes:

- Kruskal's Algorithm:

  O(m lg(n))

- Prim's Algorithm:

  O(n lg(n) + m)

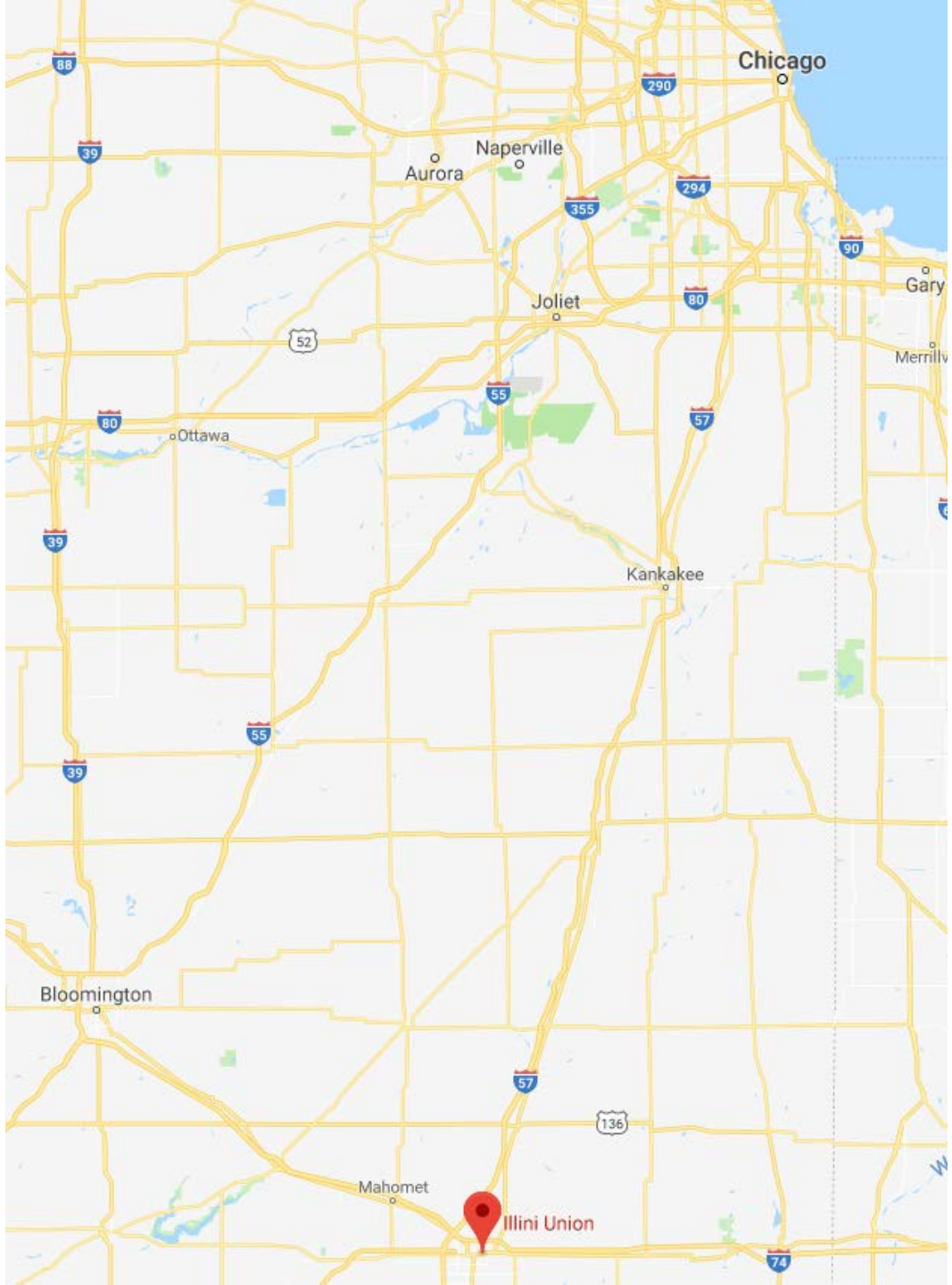

"Dec 3" Grade CDF
% of Course with GRADE ≥ X
Course Grade without EC
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

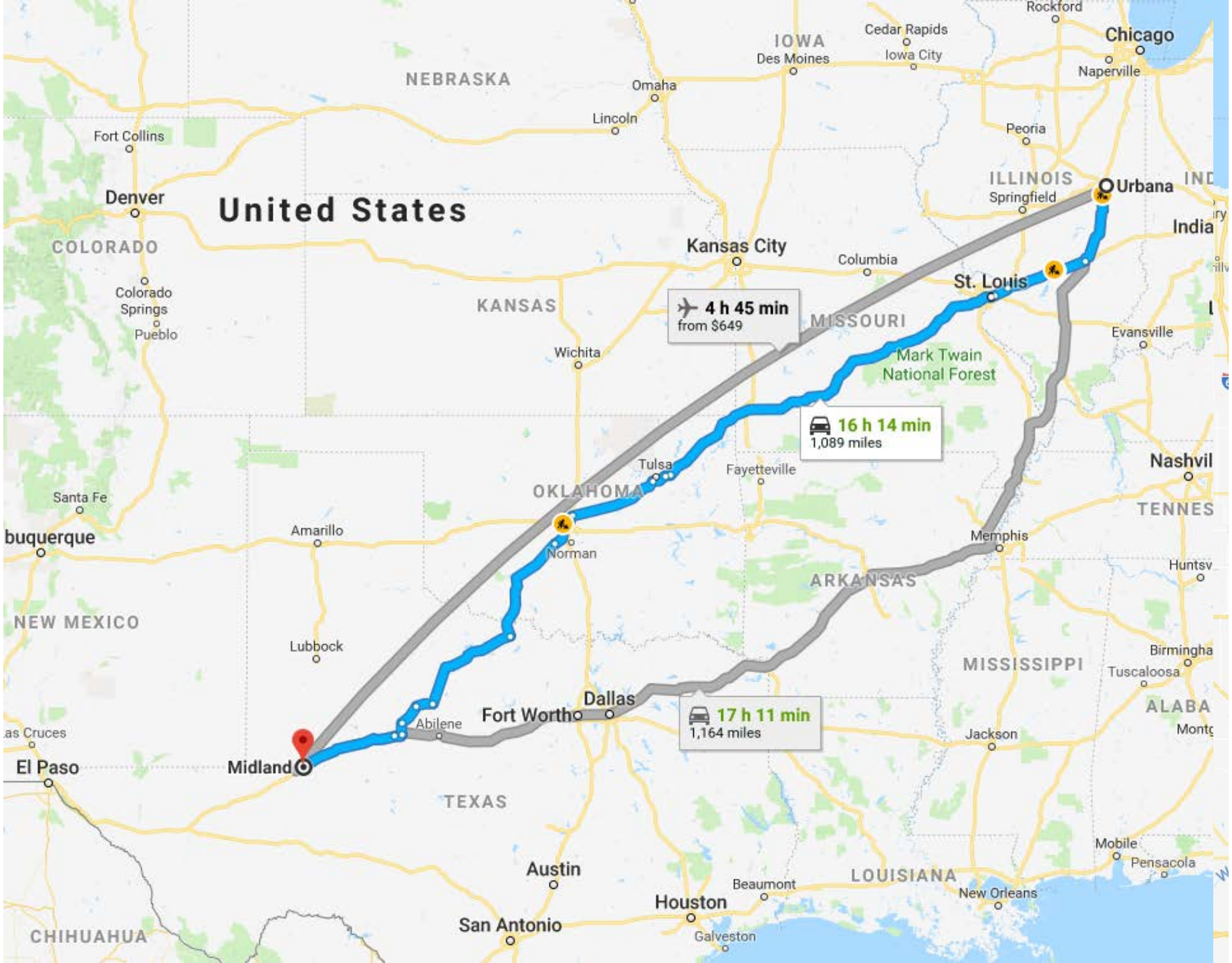
MP7

Extra Credit: Due TONIGHT! (+7 points!)

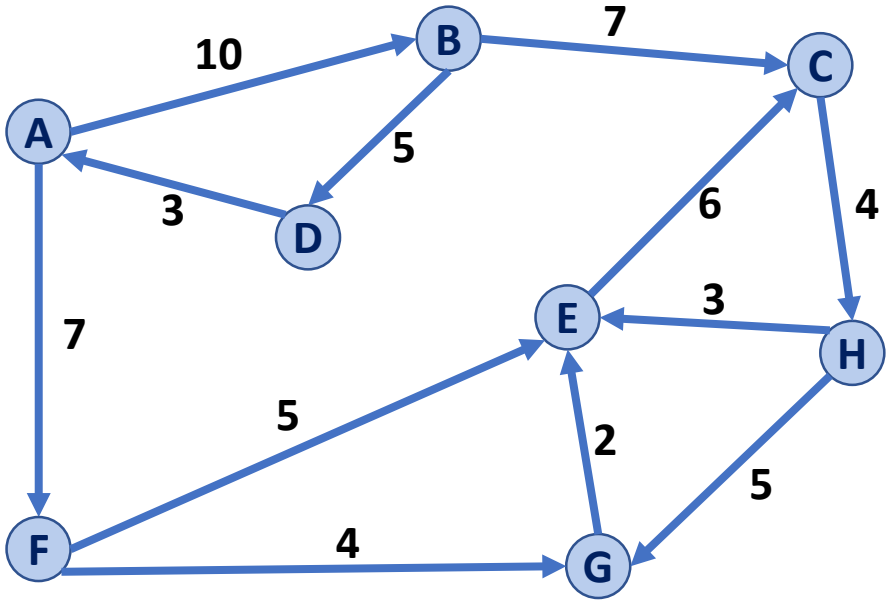
Part 3: Released Tomorrow

Shortest Path




United States
4 h 45 min
from $649
16 h 14 min
1,089 miles
17 h 11 min
1,164 miles
Midland
Urbana
Chicago
Kansas City
St. Louis
Dallas
Fort Worth
Memphis
Nashville
Denver
El Paso
Austin
San Antonio
Houston
New Orleans
Omaha
Lincoln
Wichita
Tulsa
Oklahoma City
Amarillo
Lubbock
Abilene
Norman
Fayetteville
Columbia
Springfield
Peoria
Rockford
Naperville
Cedar Rapids
Iowa City
Des Moines
Fort Collins
Colorado Springs
Pueblo
Santa Fe
Albuquerque
Las Cruces
Evansville
Mark Twain National Forest
Huntsville
Birmingham
Tuscaloosa
Montgomery
Jackson
Mobile
Pensacola
Beaumont
Galveston
NEBRASKA
IOWA
ILLINOIS
INDIANA
COLORADO
KANSAS
MISSOURI
OKLAHOMA
ARKANSAS
TENNESSEE
MISSISSIPPI
ALABAMA
LOUISIANA
TEXAS
NEW MEXICO
CHIHUAHUA

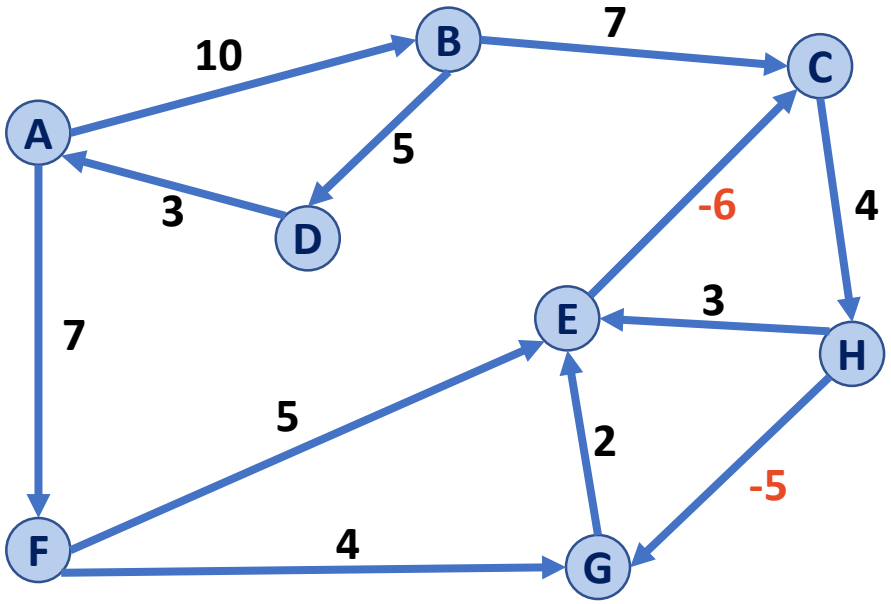
Dijkstra's Algorithm (SSSP)




```
     DijkstraSSSP(G, s):
 6     foreach (Vertex v : G):
 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 u = Q.removeMin()
17       T.add(u)
18       foreach (Vertex v : neighbors of u not in T):
19         if _________________ < d[v]:
20           d[v] = ____________________
21           p[v] = m
```

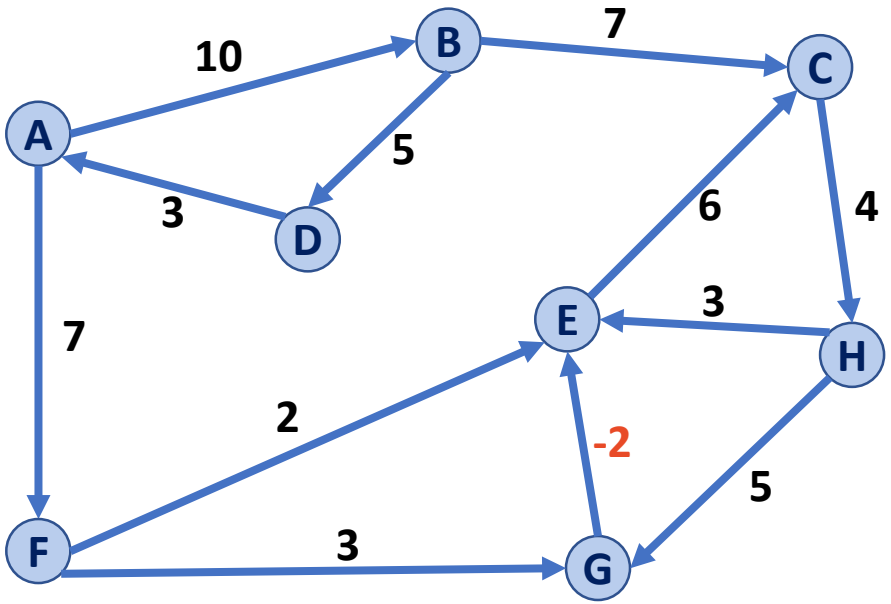
Dijkstra's Algorithm (SSSP)

What about negative weight cycles?



Dijkstra's Algorithm (SSSP)

What about negative weight edges, without negative weight cycles?



Dijkstra's Algorithm (SSSP)

What is the running time?

```
    DijkstraSSSP(G, s):
 6    foreach (Vertex v : G):
 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 u = Q.removeMin()
17      T.add(u)
18      foreach (Vertex v : neighbors of u not in T):
19        if _________________ < d[v]:
20          d[v] = ____________________
21          p[v] = m
```