CS 225

Data Structures

April 22 – Prim's Algorihtm

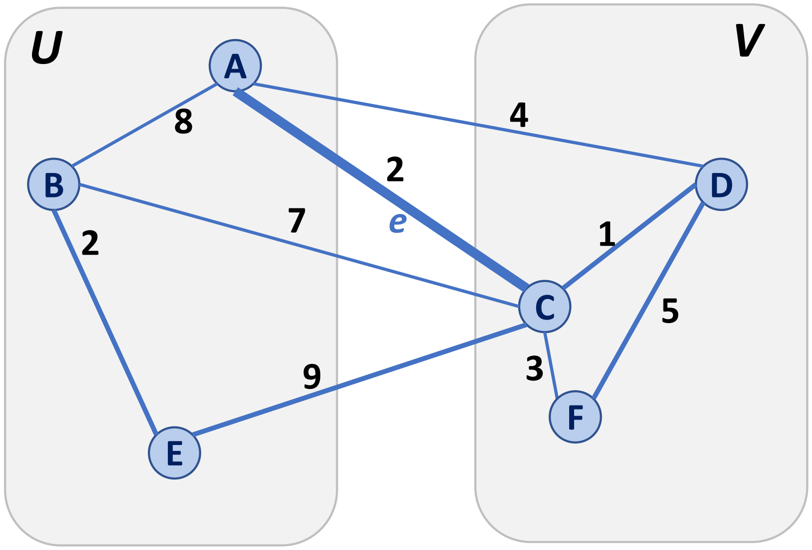
Wade Fagen-Ulmschneider, Craig Zilles

Partition Property

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

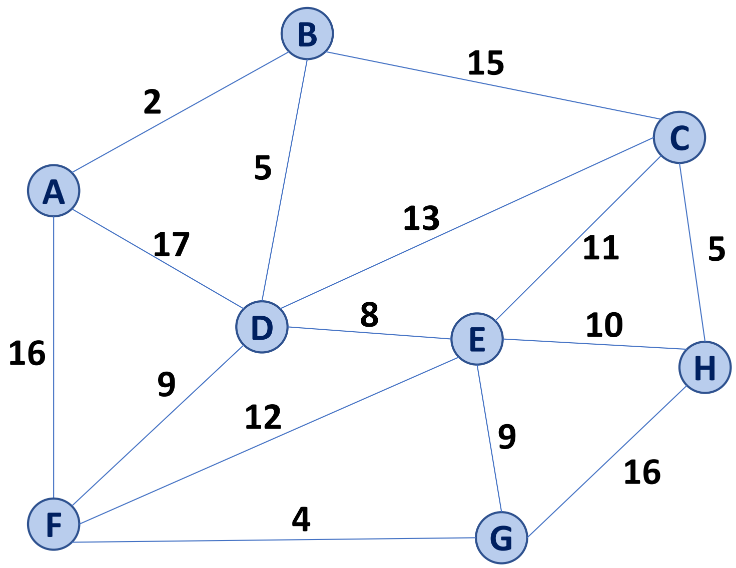
Let **e** be an edge of minimum weight across the partition.

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

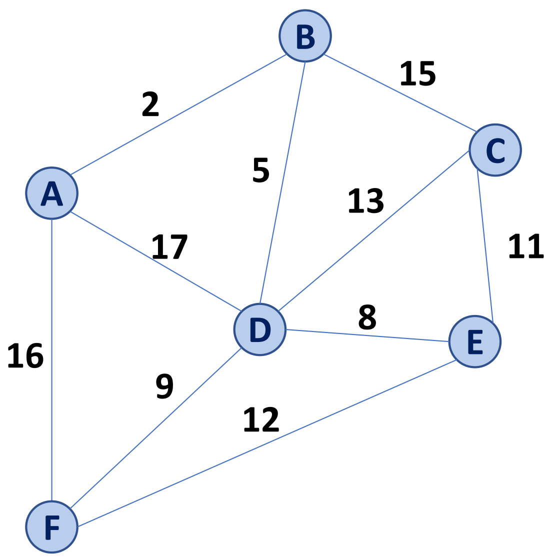


Partition Property

The partition property suggests an algorithm:



Prim's Algorithm




```
PrimMST(G, s):
  Input: G, Graph;
         s, vertex in G, starting vertex
  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
```

Prim’s Algorithm

```
PrimMST(G, 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 cost(v, m) < d[v]:
        d[v] = cost(v, m)
        p[v] = m
```

	Adj. Matrix	Adj. List
Heap		
Unsorted Array		

Prim's Algorithm

```
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
```


```
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
```

Prim's Algorithm

Sparse Graph:

Dense Graph:

```
PrimMST(G, 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 cost(v, m) < d[v]:
        d[v] = cost(v, m)
        p[v] = m
```

	Adj. Matrix	Adj. List
Heap	$O(n^2 + m \lg(n))$	$O(n \lg(n) + m \lg(n))$
Unsorted Array	$O(n^2)$	$O(n^2)$

MST Algorithm Runtime:

- Kruskal's Algorithm:
  O(n + m lg(n))
- Prim's Algorithm:
  O(n lg(n) + m lg(n))

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

  - How does n and m relate?

MST Algorithm Runtime:

- Kruskal's Algorithm:

  O(n + m lg(n))

- Prim's Algorithm:

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

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):
  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
```

Final Big-O MST Algorithm Runtimes:

- Kruskal's Algorithm:
  O(m lg(n))
- Prim's Algorithm:
  O(n lg(n) + m)

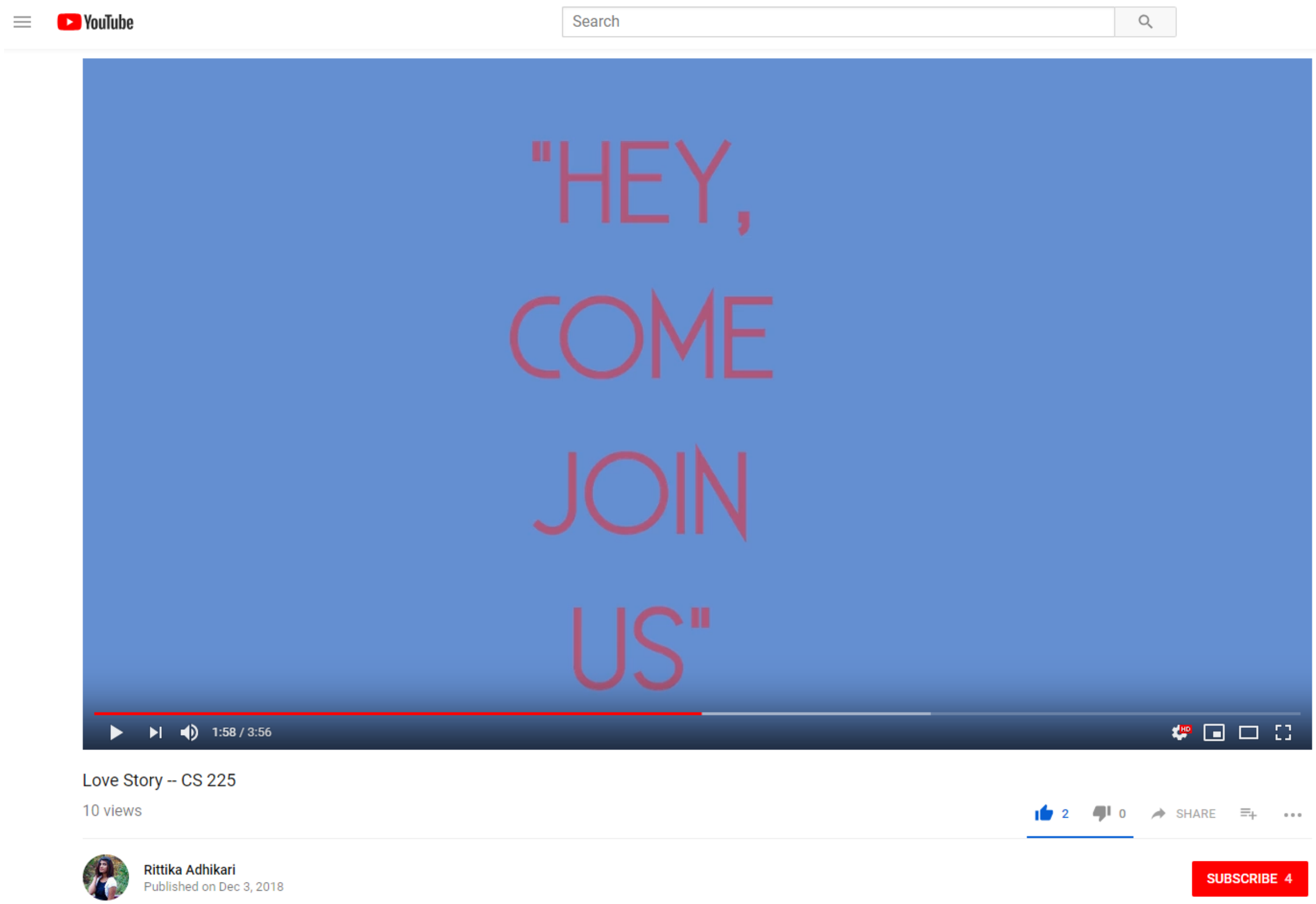
End of Semester Logistics

Lab: Your final CS 225 lab is this week.

Final Exam: Final exams start on Reading Day (May. 2)

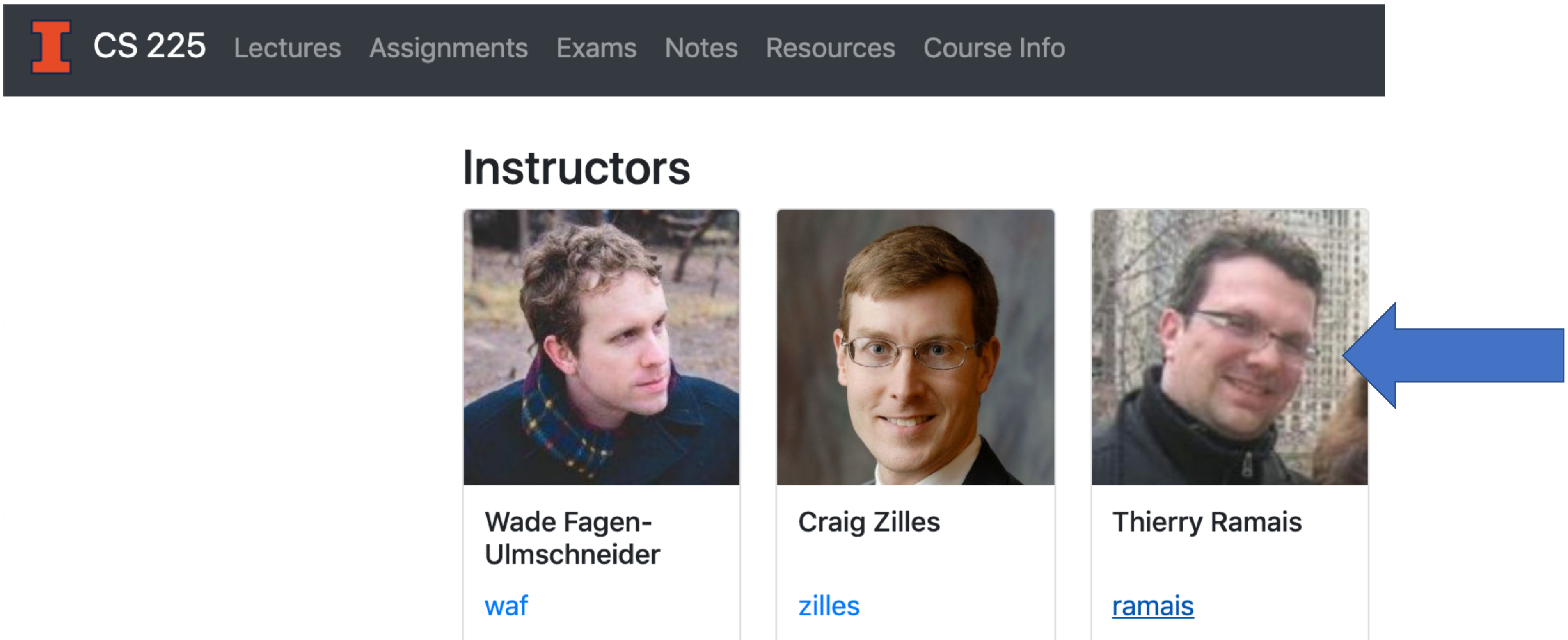
- **Final is [One Theory Exam] + [One Programming Exam] together in a single exam.**
- **Time: 3 hours**

Grades: There will be an April grade update posted this week with all grades up until now.

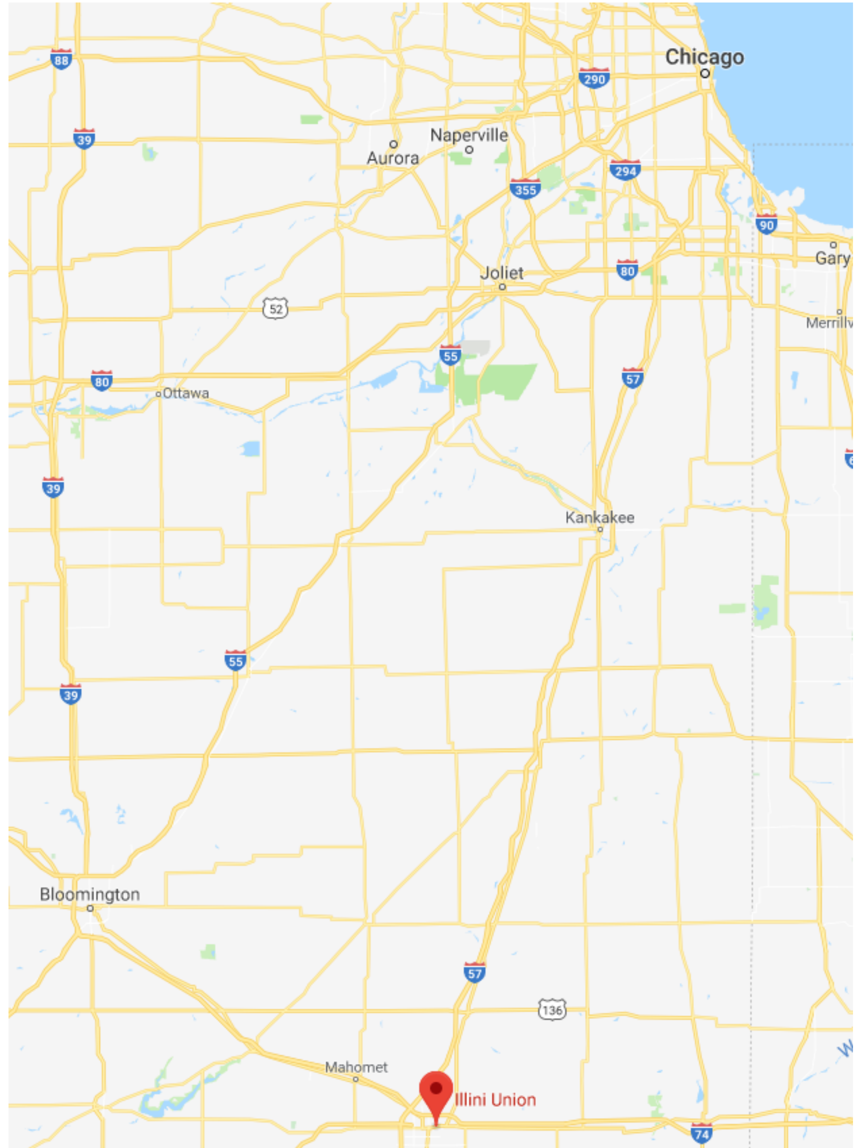


https://www.youtube.com/watch?v=7Ug1fr_ID_s

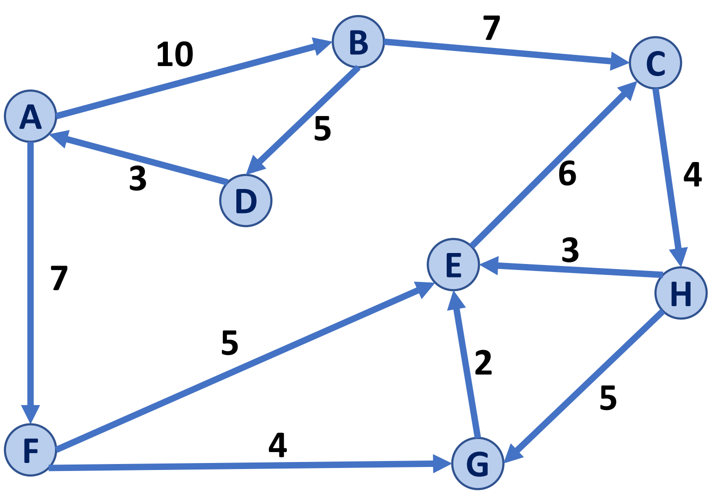
CAs


CS 225
Lectures
Assignments
Exams
Notes
Resources
Course Info
Instructors
Wade Fagen-Ulmschneider
waf
Craig Zilles
zilles
Thierry Ramais
ramais

Shortest Path



Dijkstra's Algorithm (SSSP)



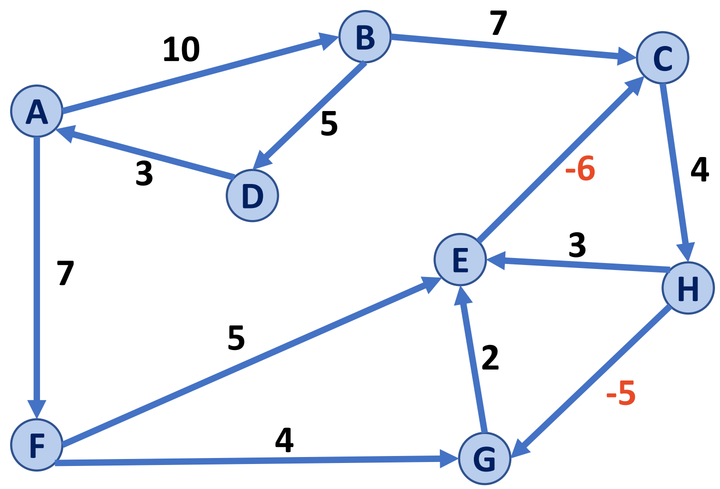

```
DijkstraSSSP(G, s):
  foreach (Vertex v : G):
    d[v] = +inf
    p[v] = NULL
  d[s] = 0

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

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

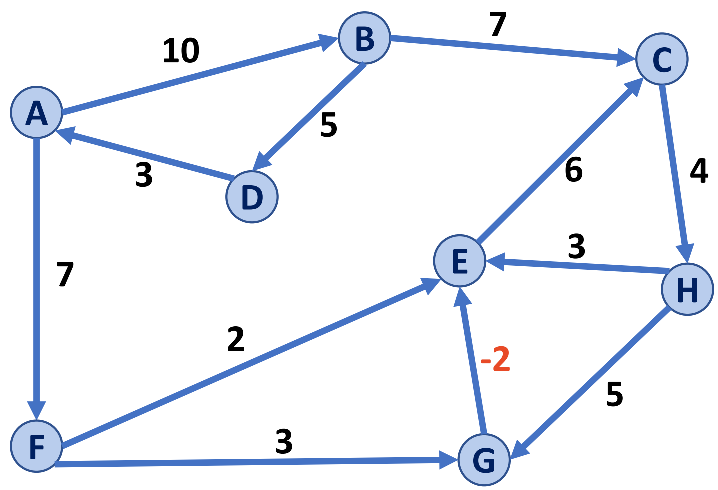
Dijkstra's Algorithm (SSSP)

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):
  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 u = Q.removeMin()
    T.add(u)
    foreach (Vertex v : neighbors of u not in T):
      if _______________ < d[v]:
        d[v] = __________________
        p[v] = m
```