CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

By Wenjie

- **Running time of MST & Prim's algorithm**

  MST Algorithms Running Times

Kruskal's Algorithm	Prim's Algorithm
$O(n + m \cdot log(n))$	$O(n \cdot log(n) + m \cdot log(n))$

  - With MSTs, we are assuming that the graph is connected and that it has at least $n-1$ edges, $m \geq n-1$.
  - Therefore, O(n) is O(m), but not the opposite.

Kruskal's Algorithm	Prim's Algorithm
$O(m \cdot log(n))$	$O(m \cdot log(n))$

  - There is a way to make Prim's algorithm faster. We can use something called Fibonacci Heap to get $O(n \cdot log(n) + m)$ time, but we will not cover that in this class.

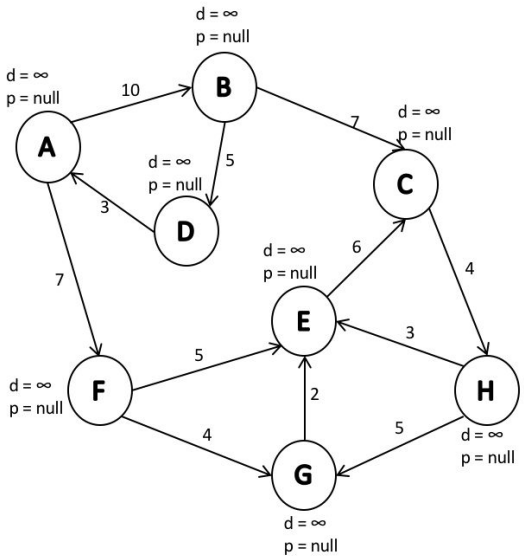
- **Dijkstra's algorithm**

Dijkstra's algorithm is used to find the shortest path between two points in a directed graph and it is a modification of Prim's algorithm.

- Algorithm setup:
  - Label all vertices with:
    - Distance → initialized to infinity.
    - Predecessor → initialized to null.
  - Set up a min heap.

CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

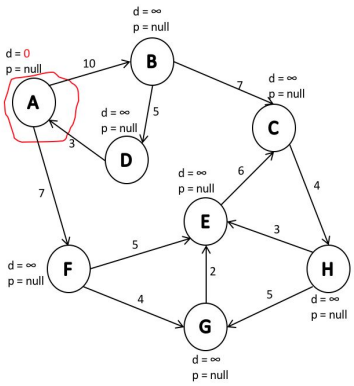
By Wenjie



V	d	p
A	∞	null
B	∞	null
C	∞	null
D	∞	null
E	∞	null
F	∞	null
G	∞	null
H	∞	null

- Algorithm logic:
    - Choose an arbitrary starting point and set its distance to 0.

In our example, we will choose vertex A as a starting point.



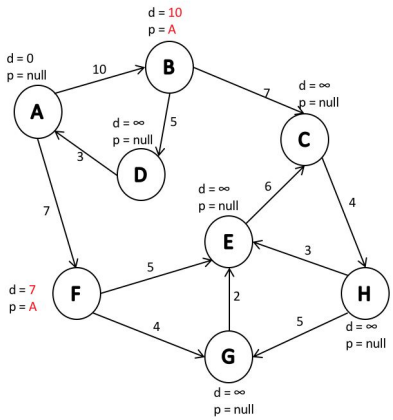
V	d	p
A	0	null
B	∞	null
C	∞	null
D	∞	null
E	∞	null
F	∞	null
G	∞	null
H	∞	null

    - Pop the starting vertex from the heap and update the distance/predecessor of adjacent vertices.

CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

By Wenjie

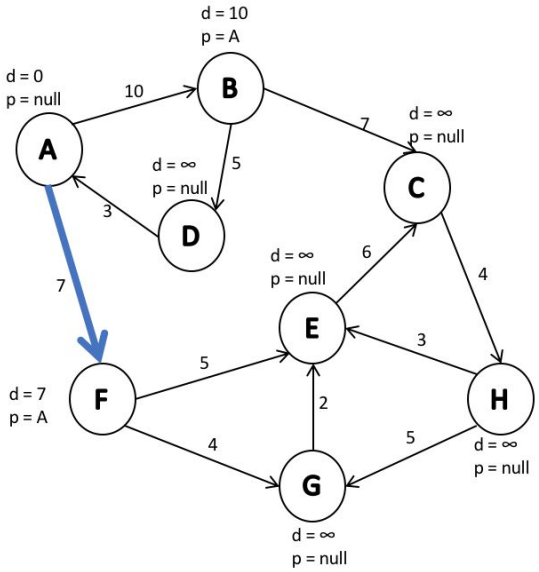
- We pop A and update adjacent vertices B and F



V	d	p
~~A~~	~~0~~	~~null~~
B	10	A
C	∞	null
D	∞	null
E	∞	null
F	7	A
G	∞	null
H	∞	null

- We want to add an edge to the node with the smallest distance.

In the example, an edge with the smallest *d* is the edge from A to F.



V	d	p
~~A~~	~~0~~	~~null~~
B	10	A
C	∞	null
D	∞	null
E	∞	null
F	7	A
G	∞	null
H	∞	null

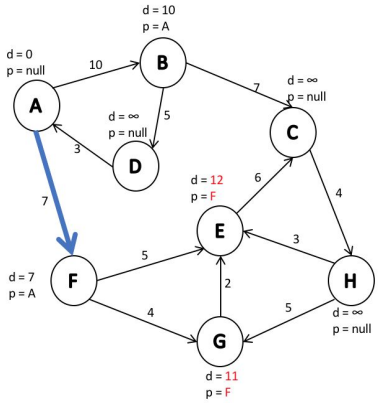
  - Next, we pop a vertex with the smallest distance and update adjacent vertices. However, we update vertices only if the distance **from the start** is smaller than the current d.

CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

By Wenjie

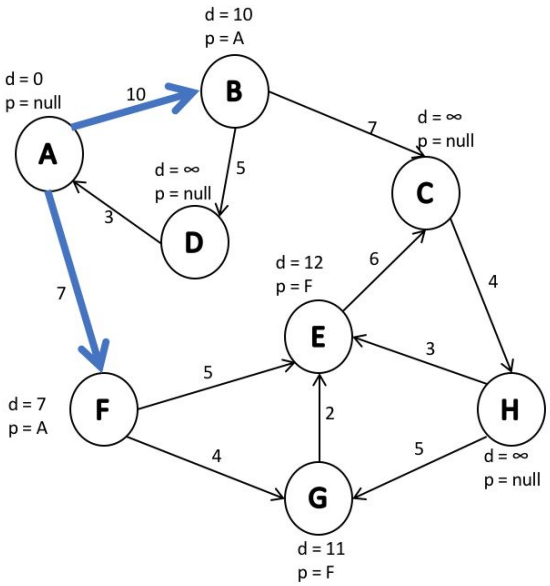
- This is the main distinction from Prim's algorithm. Prim cares about the smallest edge weights, while Dijkstra is concerned with smallest paths.

We pop F and update adjacent vertices: E to $7 + 5 = 12$ and G to $7 + 4 = 11$.



V	d	p
~~A~~	~~0~~	~~null~~
B	10	A
C	∞	null
D	∞	null
E	12	F
~~F~~	~~7~~	~~A~~
G	11	F
H	∞	null

We add an edge to the node with the smallest path.
The node with the smallest path is B, so we add edge AB.



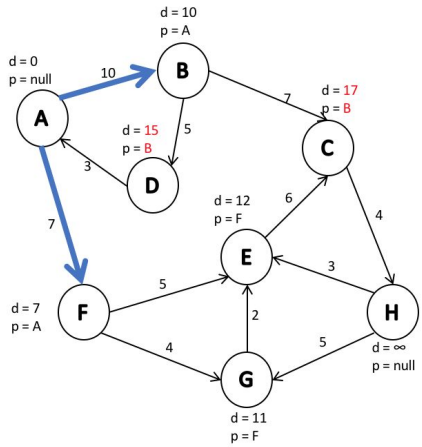
V	d	p
~~A~~	~~0~~	~~null~~
B	10	A
C	∞	null
D	∞	null
E	12	F
~~F~~	~~7~~	~~A~~
G	11	F
H	∞	null

CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

By Wenjie

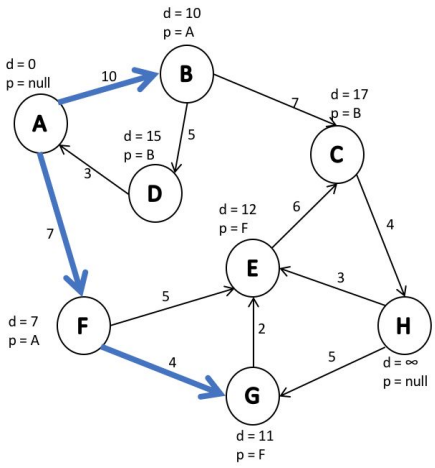
- Again, we pop a vertex with the smallest path, we update adjacent vertices if needed, and we add the edge with the smallest path. These steps are repeated until the heap is empty.

Continuing through the example graph → we pop B and we update its adjacent vertices C and D.



V	d	p
~~A~~	~~0~~	~~null~~
~~B~~	~~10~~	~~A~~
C	17	B
D	15	B
E	12	F
~~F~~	~~7~~	~~A~~
G	11	F
H	∞	null

The next vertex with smallest path is G. Thereby, we add an edge from F to G.

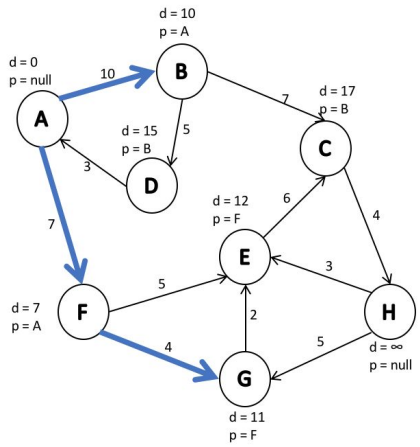


V	d	p
~~A~~	~~0~~	~~null~~
~~B~~	~~10~~	~~A~~
C	17	B
D	15	B
E	12	F
~~F~~	~~7~~	~~A~~
G	11	F
H	∞	null

We further pop G, but we do not update E, because the path to E through G is longer than the path to E through F.

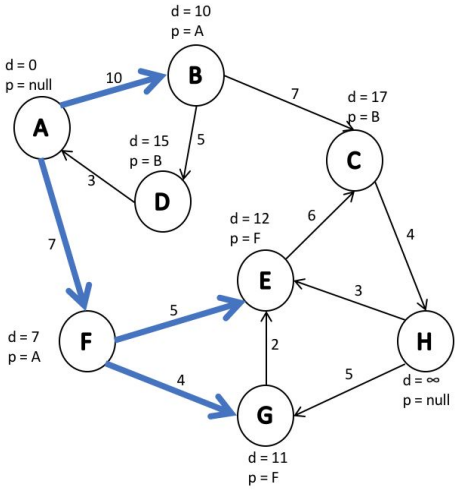
CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

By Wenjie



V	d	p
A	0	null
B	10	A
C	17	B
D	15	B
E	12	F
F	7	A
G	11	F
H	∞	null

We add an edges from E to F because that is currently the path with the smallest distance.

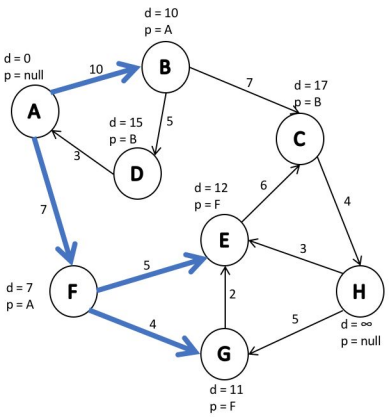


V	d	p
A	0	null
B	10	A
C	17	B
D	15	B
E	12	F
F	7	A
G	11	F
H	∞	null

Next, pop vertex E and we won't update anything again because the path to C through E is longer than path to C through B.

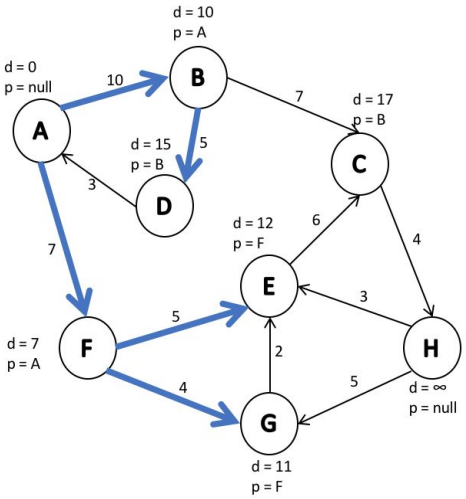
CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

By Wenjie



V	d	p
A	0	null
B	10	A
C	17	B
D	15	B
E	12	F
F	7	A
G	11	F
H	∞	null

Then, we add an edge from B to D and pop D. We do not update anything.

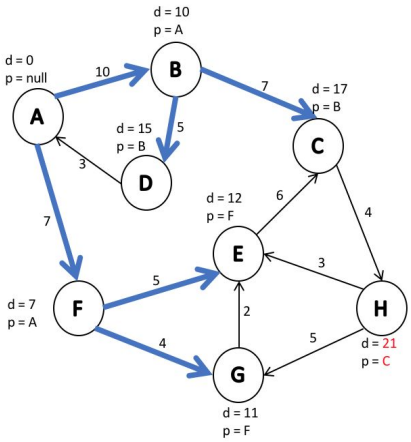


V	d	p
A	0	null
B	10	A
C	17	B
D	15	B
E	12	F
F	7	A
G	11	F
H	∞	null

We add an edge from B to C and pop C. Here we have to update H because path to H through C is less than infinity.

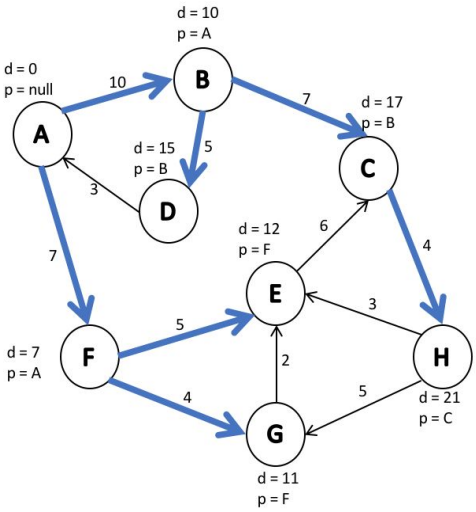
CS 225 Spring 2019 :: TA Lecture Notes 4/24 Dijkstra's Algorithm (SSSP)

By Wenjie



V	d	p
A	0	null
B	10	A
C	17	B
D	15	B
E	12	F
F	7	A
G	11	F
H	21	C

Finally, we add an edge from C to H and pop H. After this step, the heap is empty and we are done.

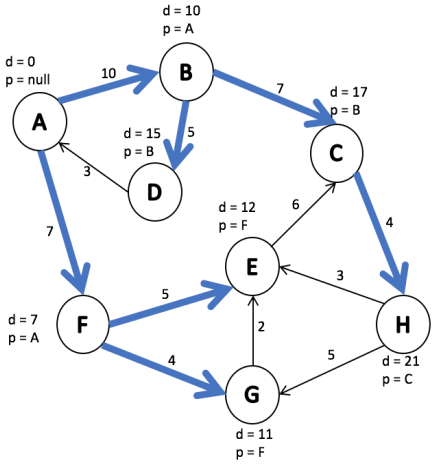


V	d	p
A	0	null
B	10	A
C	17	B
D	15	B
E	12	F
F	7	A
G	11	F
H	21	C

- After running Dijkstra's algorithm, we get an acyclic directed graph. In the example, the starting point is A.
    - This means that we have the shortest path from A to any other vertex in the graph.

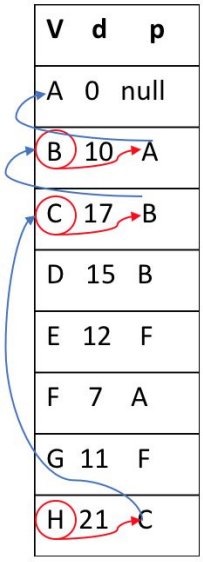
CS 225 Spring 2019 :: TA Lecture Notes
4/24 Dijkstra's Algorithm (SSSP)

By Wenjie



V	d	p
A	0	null
B	10	A
C	17	B
D	15	B
E	12	F
F	7	A
G	11	F
H	21	C

- Now we can ask the following questions:
  - What is the shortest path from A to H?
    - By looking up the table, we can see it is 21.
    - The time to find this information is O(1) because it is just a simple table lookup.
  - What is the path from A to H?
    - Start at H and trace back the predecessor nodes → A-B-C-H



  - If there is no path to a particular vertex, we will have infinity as distance.