Revised: 8/11/2024

Graphs III

Michael C. Hackett

Assistant Professor, Computer Science

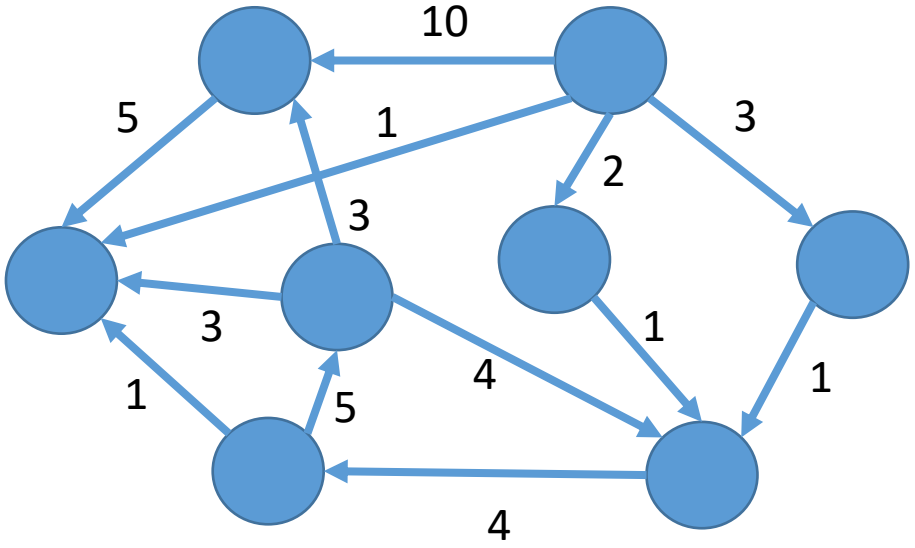
COMMUNITY COLLEGE OF PHILADELPHIA

Lecture Topics

- Weighted Graphs
  - Dijkstra's Algorithm

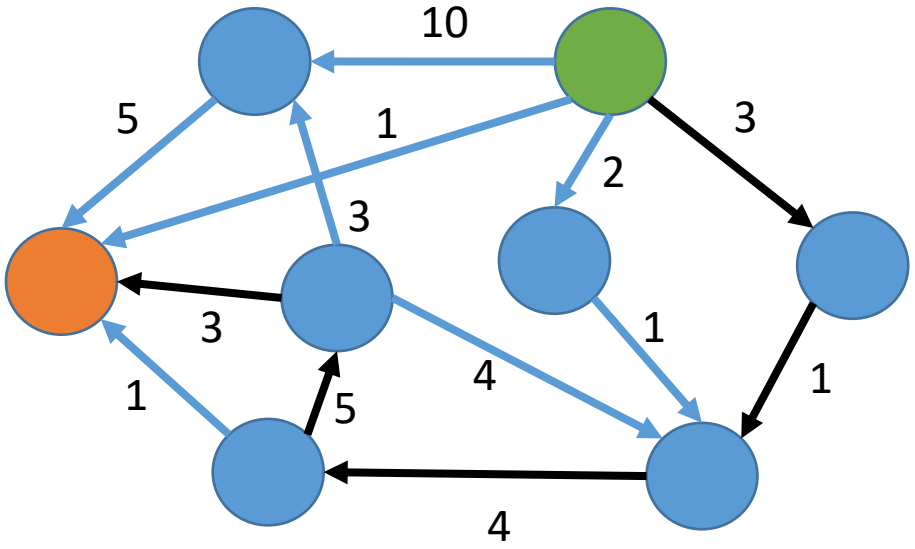
Weighted Graphs

- A **weighted graph** is a graph where each edge has a weight or cost.
  - Weighted graphs can be undirected or directed



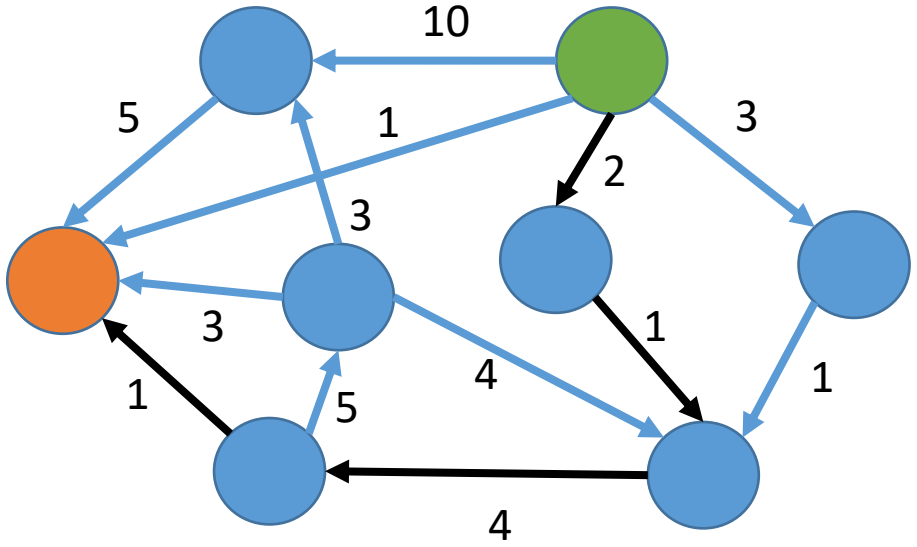
Weighted Graphs

- The path length of a weighted graph is the sum of the edge costs.
  - 3+1+4+5+3 = **16**

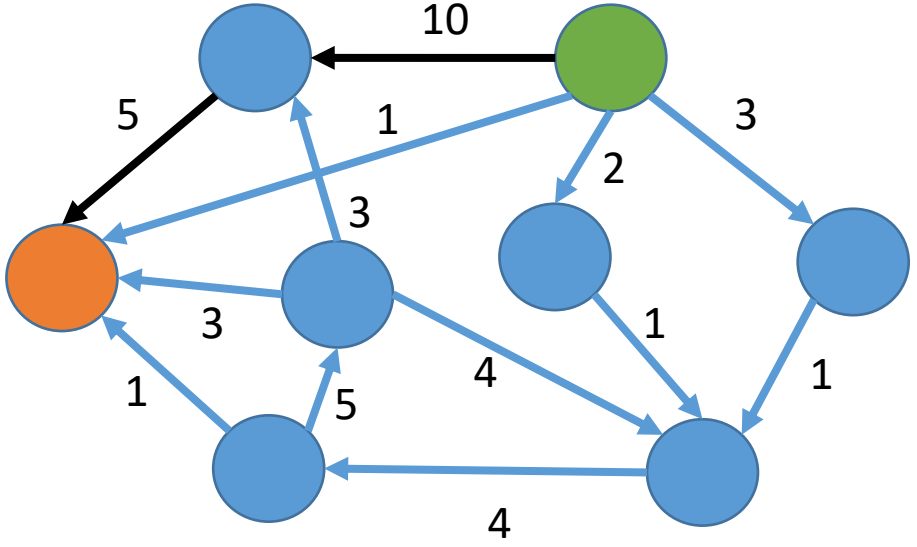


Weighted Graphs

Unweighted Length = 4
Weighted Length = 8



Unweighted Length = 2
Weighted Length = 15



- The first path is less costly than the second path, despite it being twice as long

Weighted Graphs

- A breadth-first traversal would not be useful for finding the path with the least cost
- Other algorithms are used to find the path with the least cost between two vertices
- The most well known is Djikstra's Algorithm

Dijkstra's Algorithm

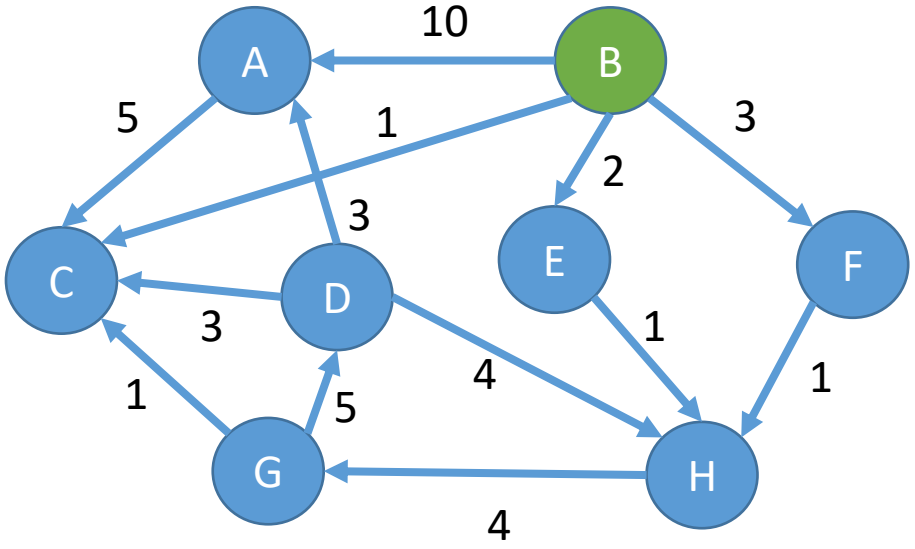
- This algorithm finds the shortest path between one vertex and **every other vertex** in a graph.

- *For each vertex*
  - The algorithm determines the vertex's distance (shortest/least costly path) from the starting vertex
  - The algorithm determines the vertex's predecessor pointer- the previous vertex with the shortest (or least costly) path from the starting vertex

- Can be used on:
  - Bi-directional and digraphs
  - Weighted graphs and unweighted graphs

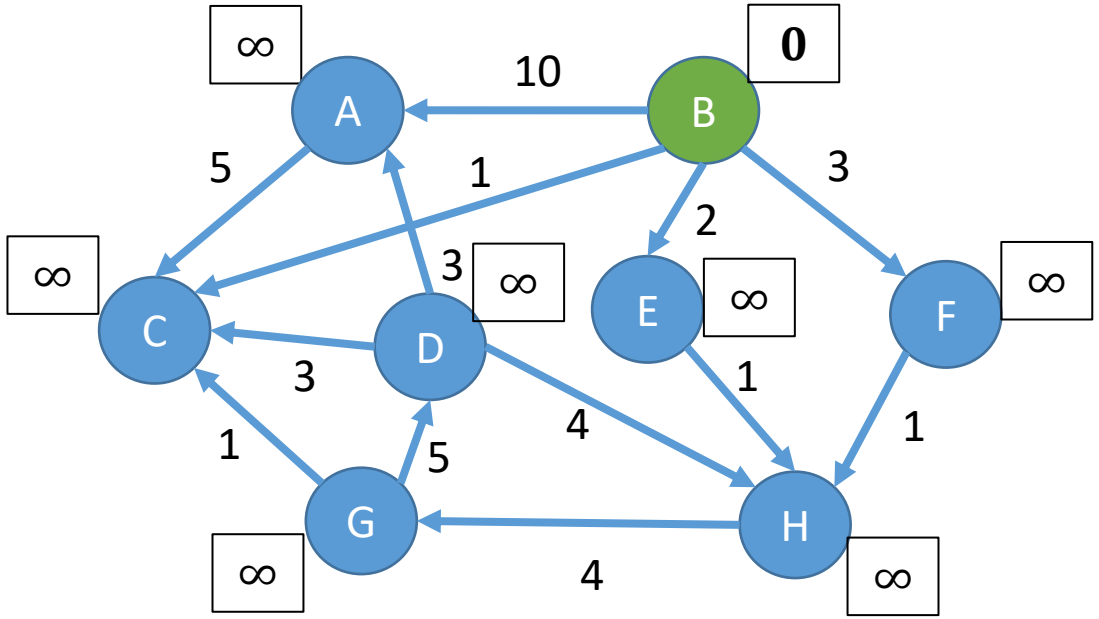
Dijkstra's Algorithm

- We can start with any vertex, but we'll start with vertex B since a path exists from B to all other vertices



Dijkstra's Algorithm

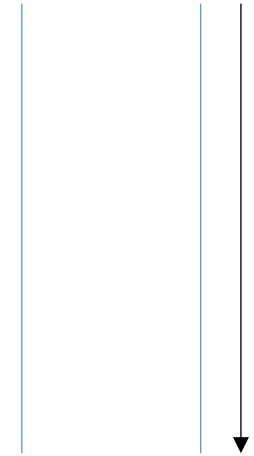
- We'll remember the cost from each vertex back to vertex B.
- B has a cost of 0; All other vertices are assumed to have a cost of infinity
  - Ensures the path found will be less than that



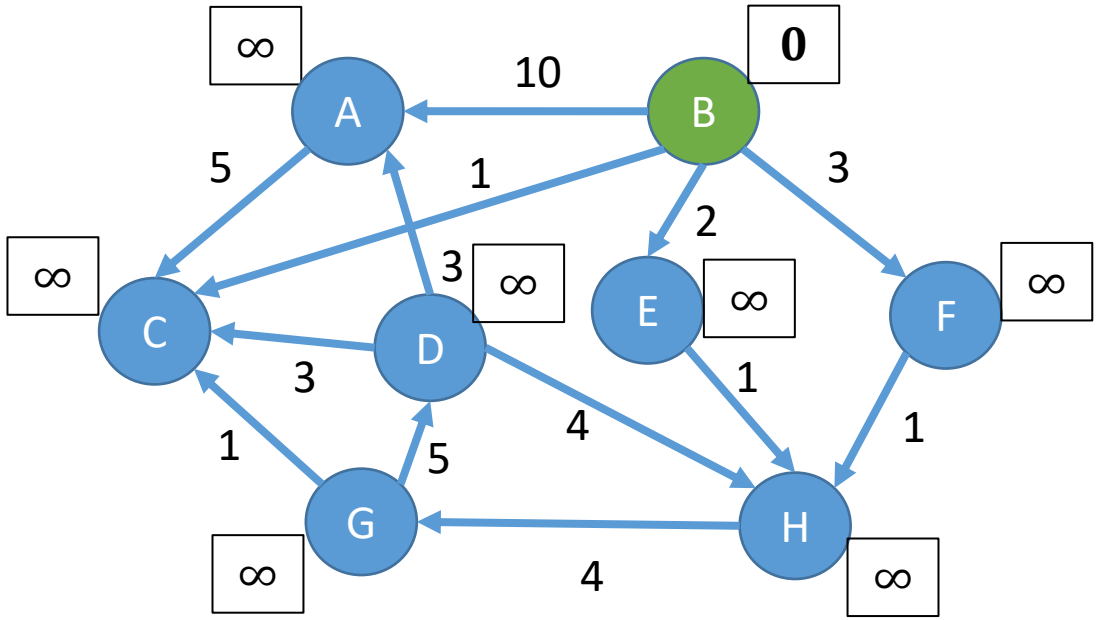
A cost = ∞
B cost = 0
C cost = ∞
D cost = ∞
E cost = ∞
F cost = ∞
G cost = ∞
H cost = ∞

Dijkstra's Algorithm

Priority Queue/Min-heap

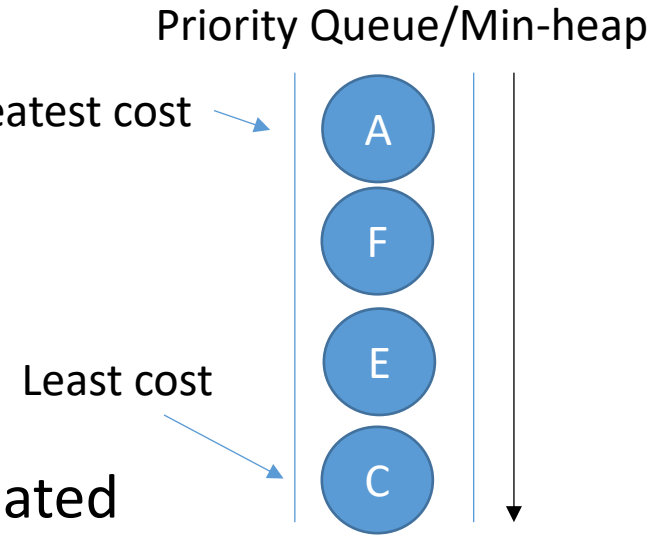


- A priority queue or min-heap is used to prioritize vertices with lower costs to be visited first

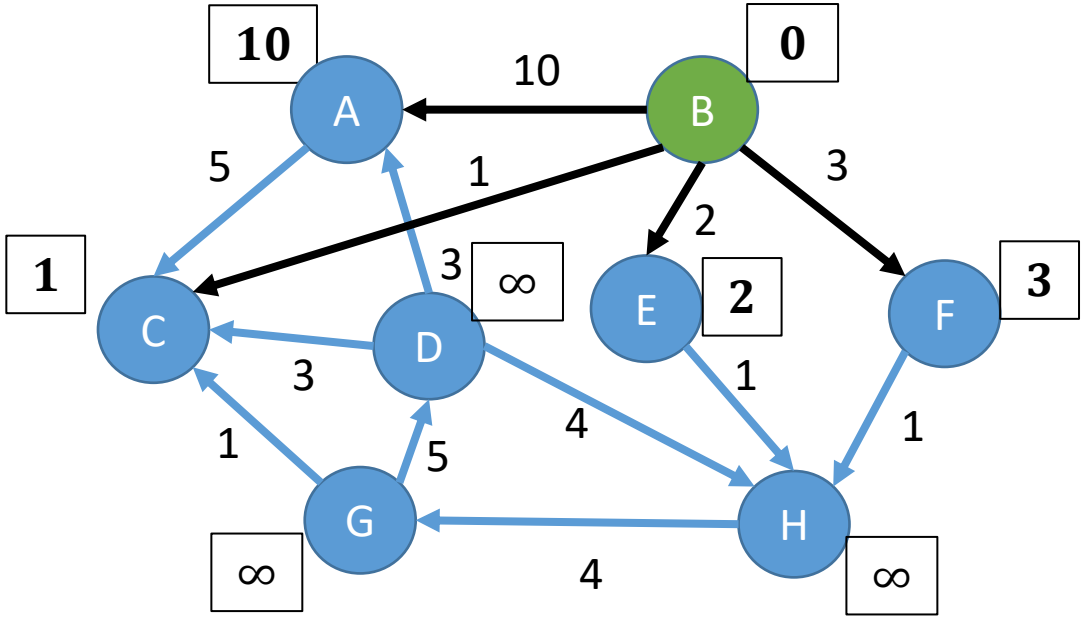


A cost = ∞
B cost = 0
C cost = ∞
D cost = ∞
E cost = ∞
F cost = ∞
G cost = ∞
H cost = ∞

Dijkstra's Algorithm



- We'll now look at the vertices adjacent to B.
- Cost of each is B's cost + cost of the edge
  - All of which are less than infinity so the costs are updated

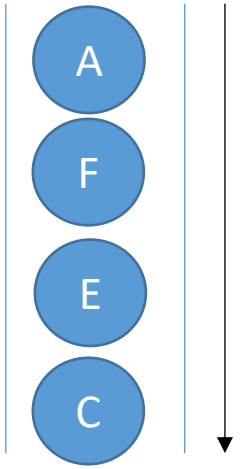


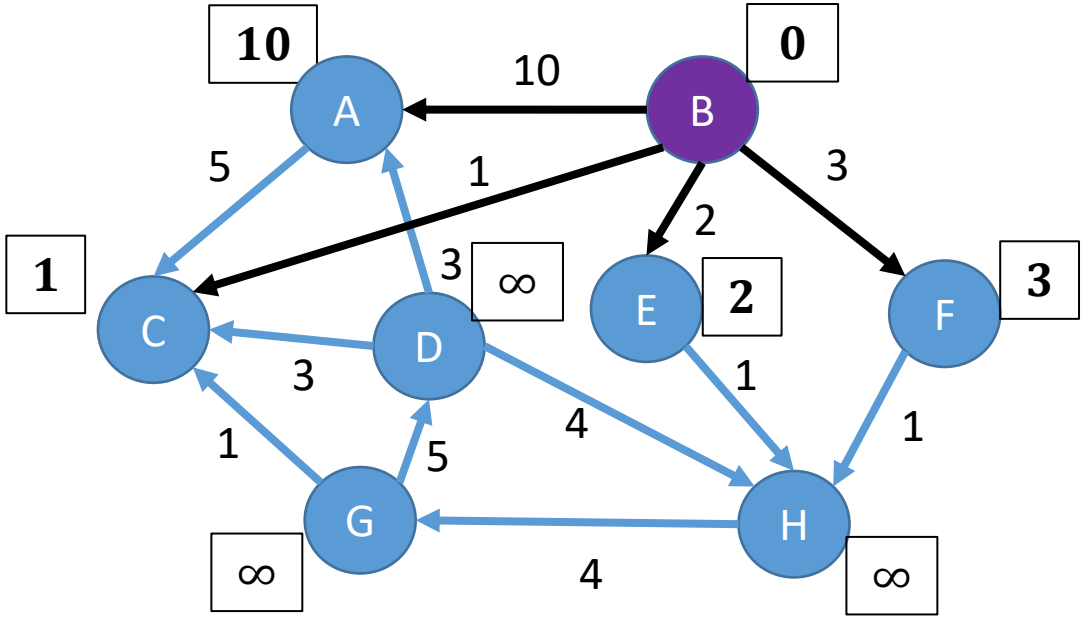
A cost = $0 + 10 = 10$
B cost = 0
C cost = $0 + 1 = 1$
D cost = ∞
E cost = $0 + 2 = 2$
F cost = $0 + 3 = 3$
G cost = ∞
H cost = ∞

Dijkstra's Algorithm

- We are finished with B

Priority Queue/Min-heap

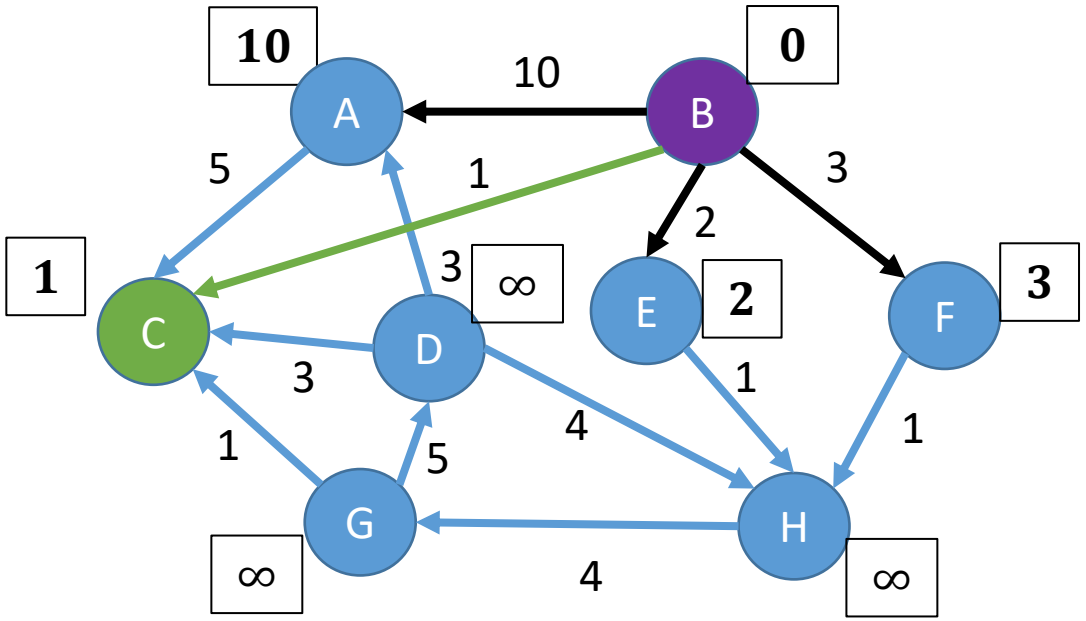




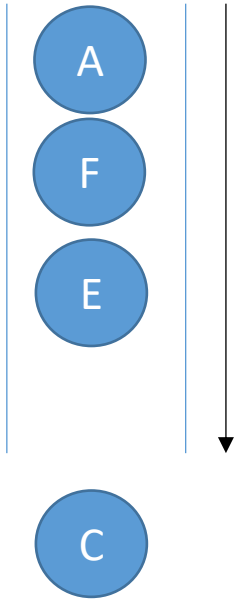
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = ∞

Dijkstra's Algorithm

- We move on to the next vertex in the priority queue



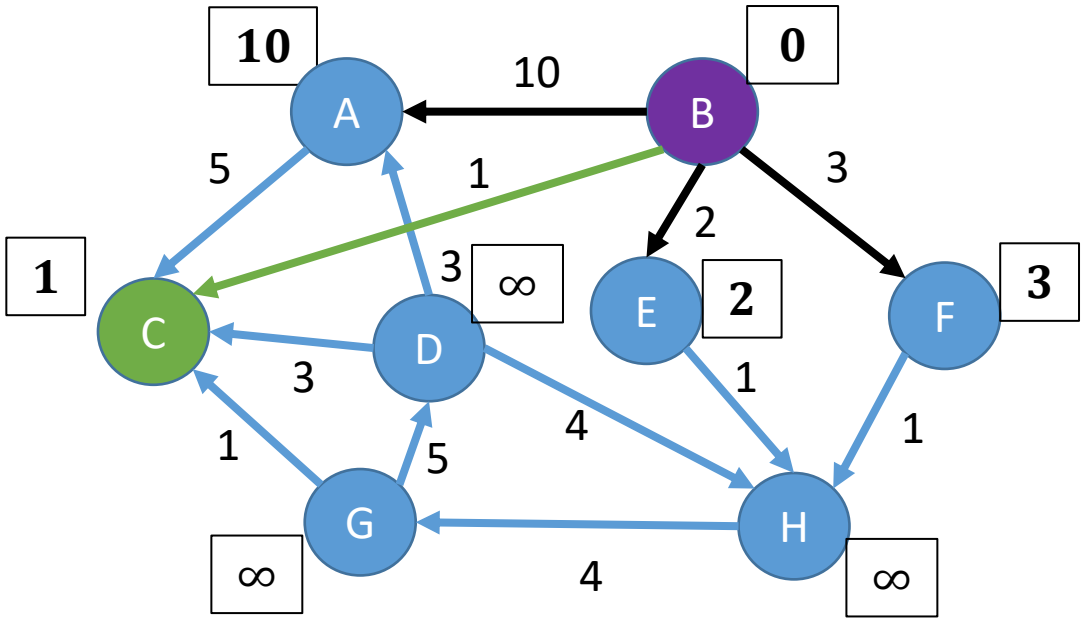
Priority Queue/Min-heap



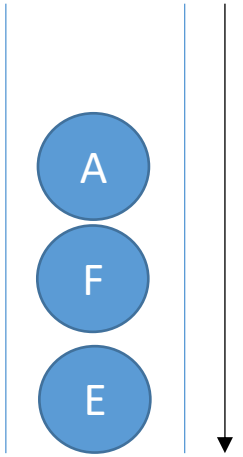
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = ∞

Dijkstra's Algorithm

- We'll now look at the vertices adjacent to C.
  - There are none



Priority Queue/Min-heap

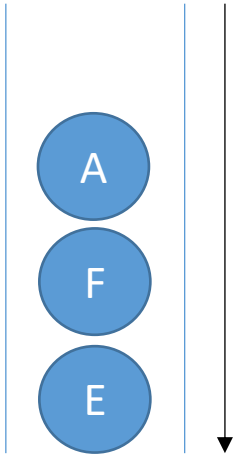


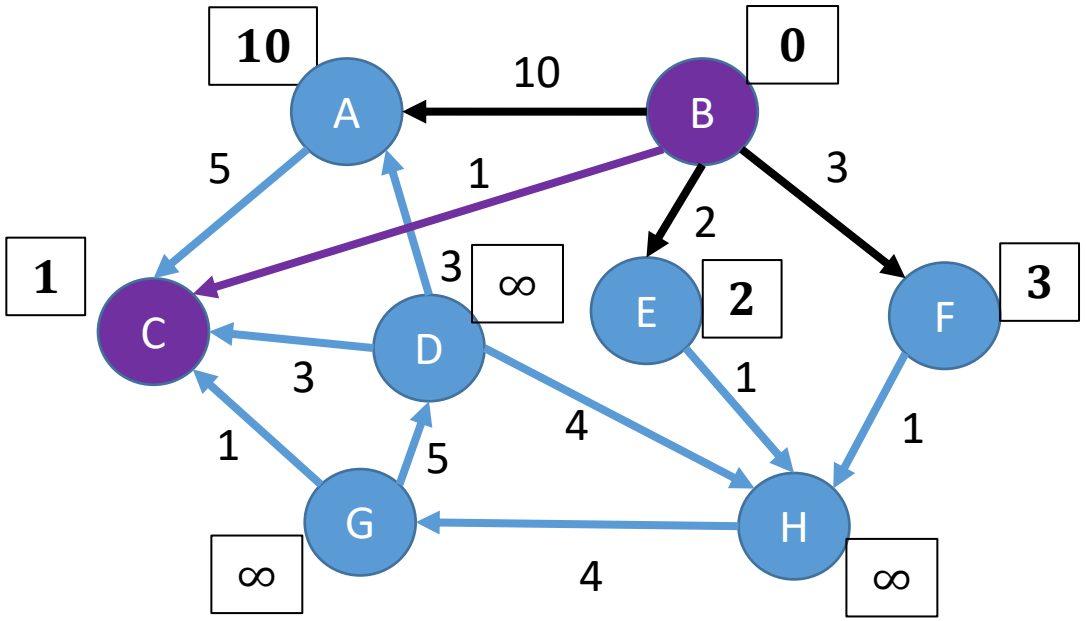
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = ∞

Dijkstra's Algorithm

- We are finished with C

Priority Queue/Min-heap

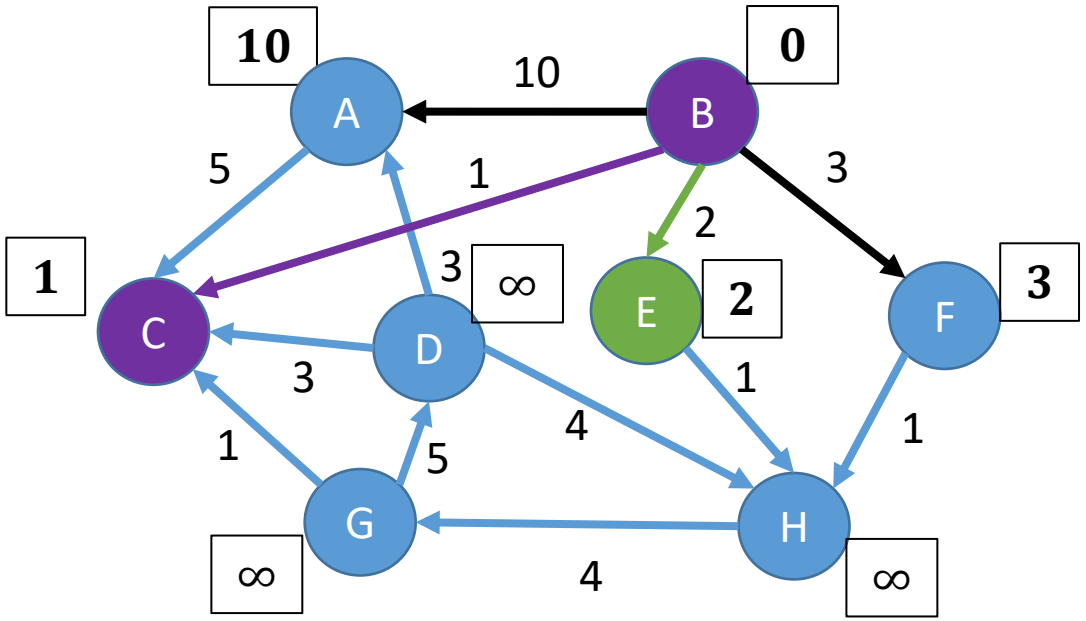




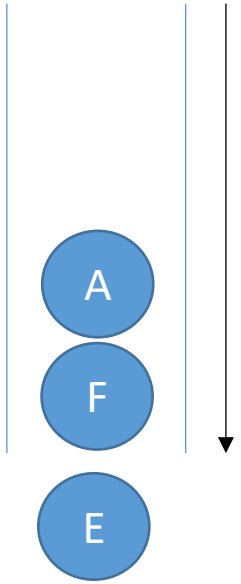
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = ∞

Dijkstra's Algorithm

- We move on to the next vertex in the priority queue



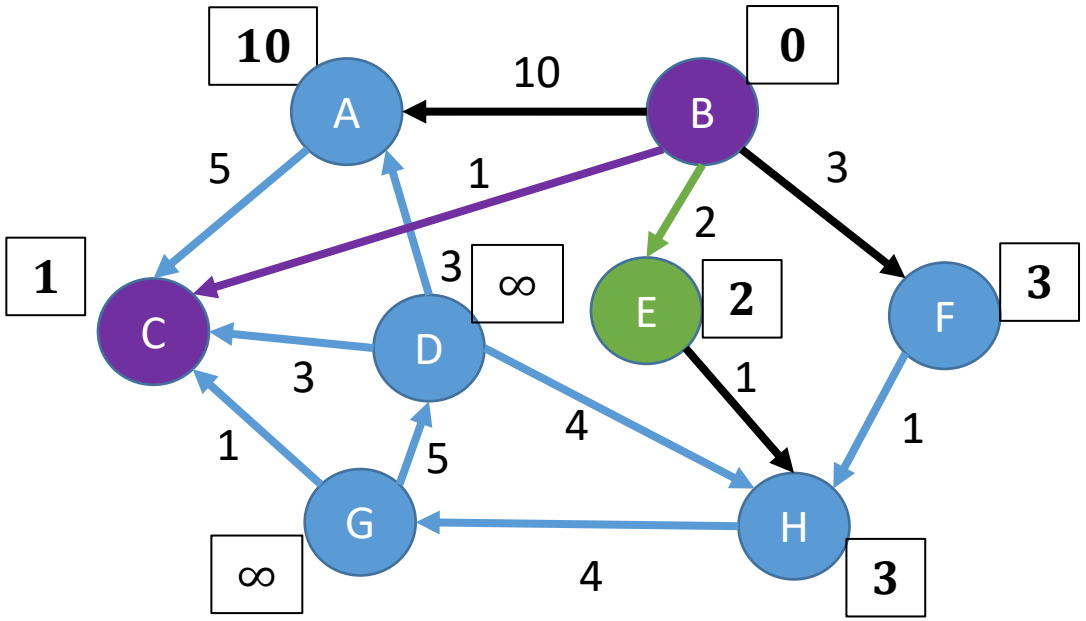
Priority Queue/Min-heap



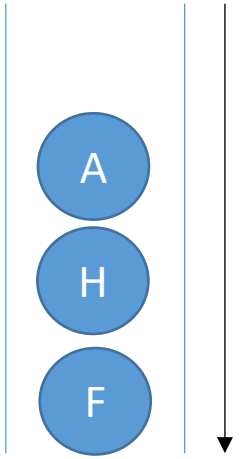
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = ∞

Dijkstra's Algorithm

- We'll now look at the vertices adjacent to E.
  - Cost of each is E's cost + cost of the edge



Priority Queue/Min-heap

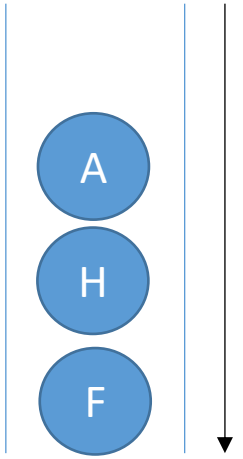


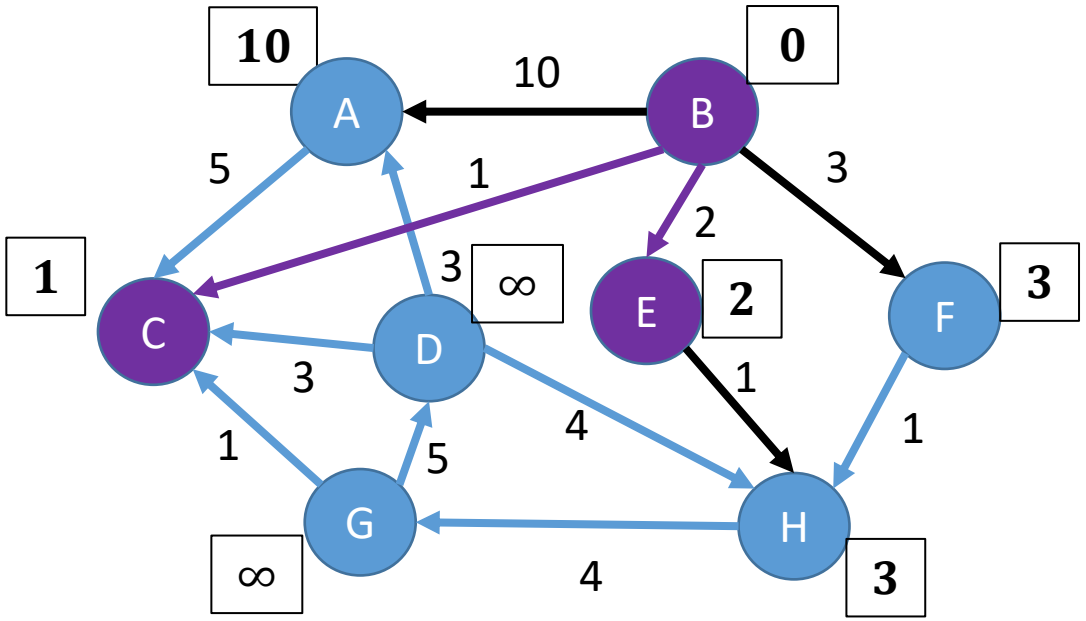
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = 2 + 1 = 3

Dijkstra's Algorithm

- We are finished with E.

Priority Queue/Min-heap

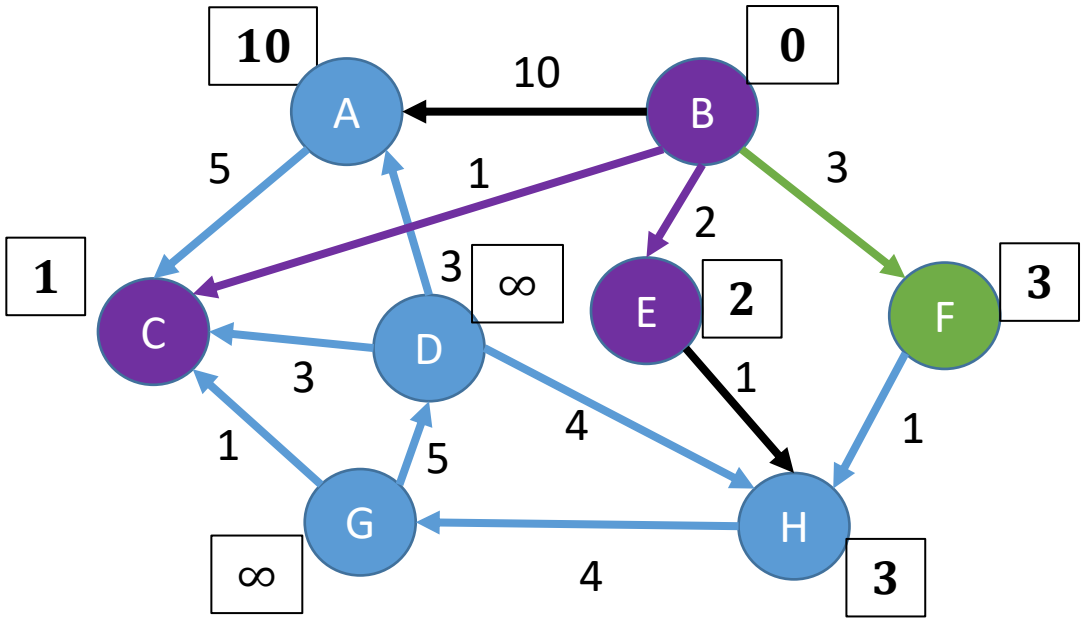




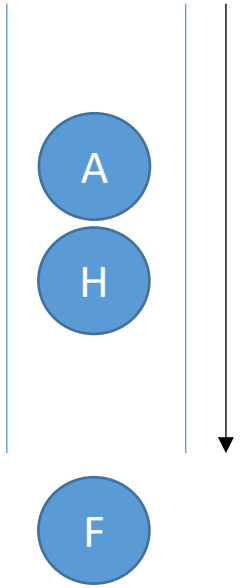
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = 3

Dijkstra's Algorithm

- We move on to the next vertex in the priority queue



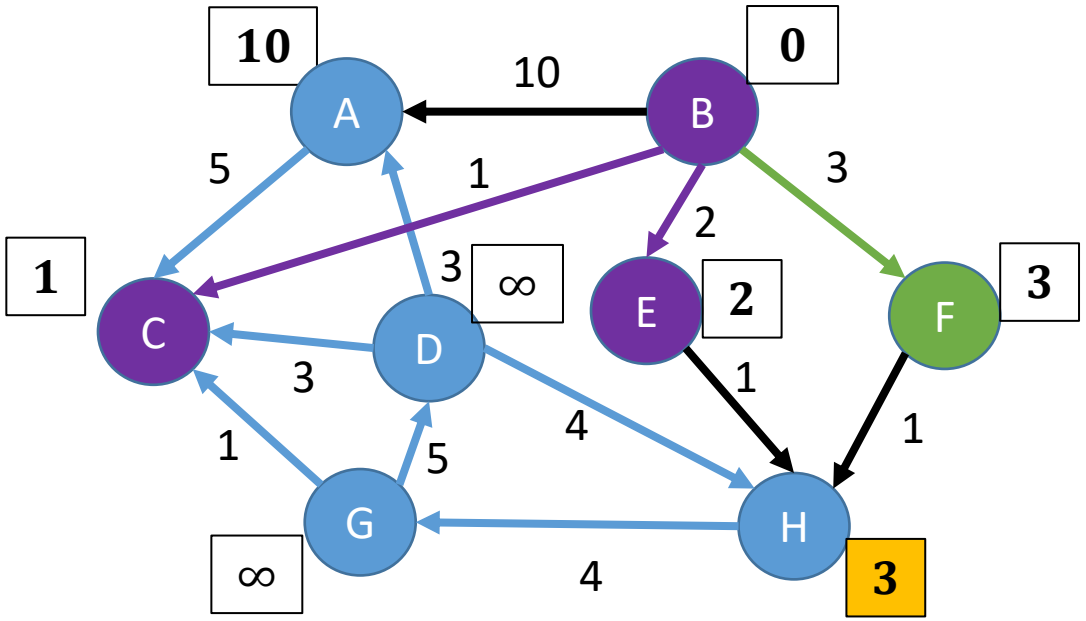
Priority Queue/Min-heap



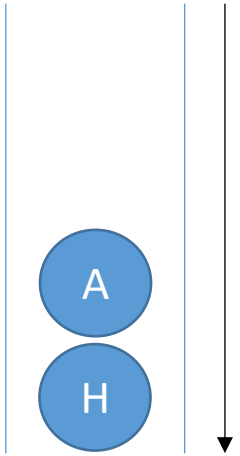
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = 3

Dijkstra's Algorithm

- We'll now look at the vertices adjacent to F.
  - Cost of each is F's cost + cost of the edge
  - 3 + 1 = **4** (NOT LESS THAN THE CURRENT COST OF H)



Priority Queue/Min-heap

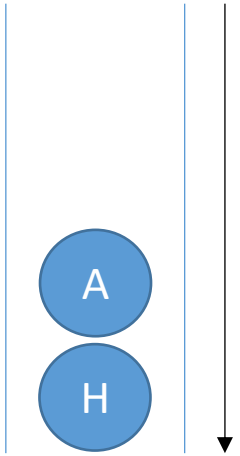


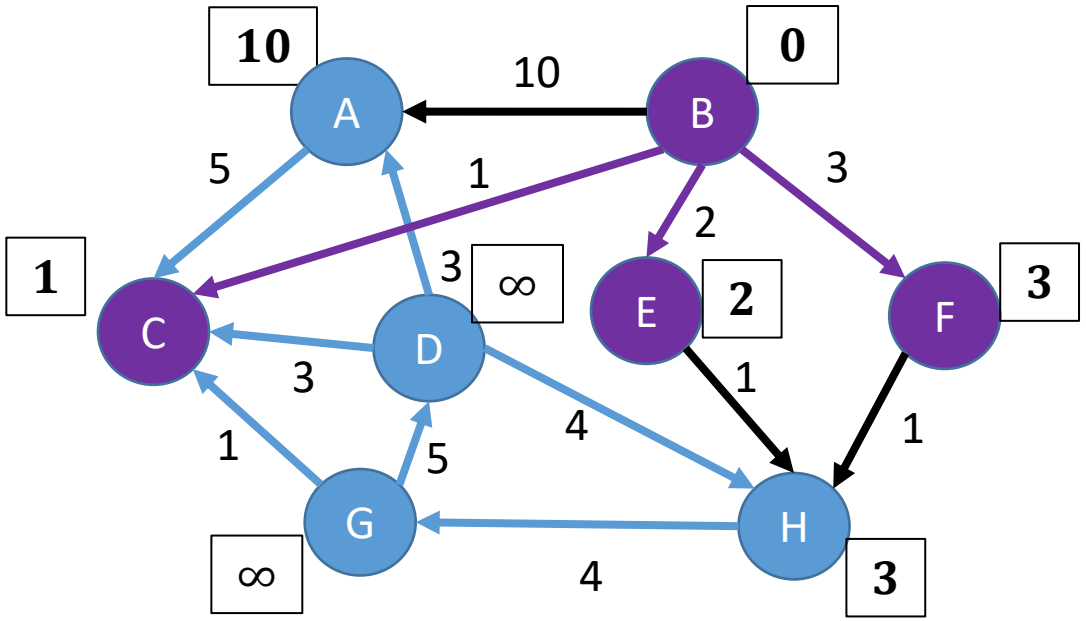
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = **3**

Dijkstra's Algorithm

- We are finished with F

Priority Queue/Min-heap

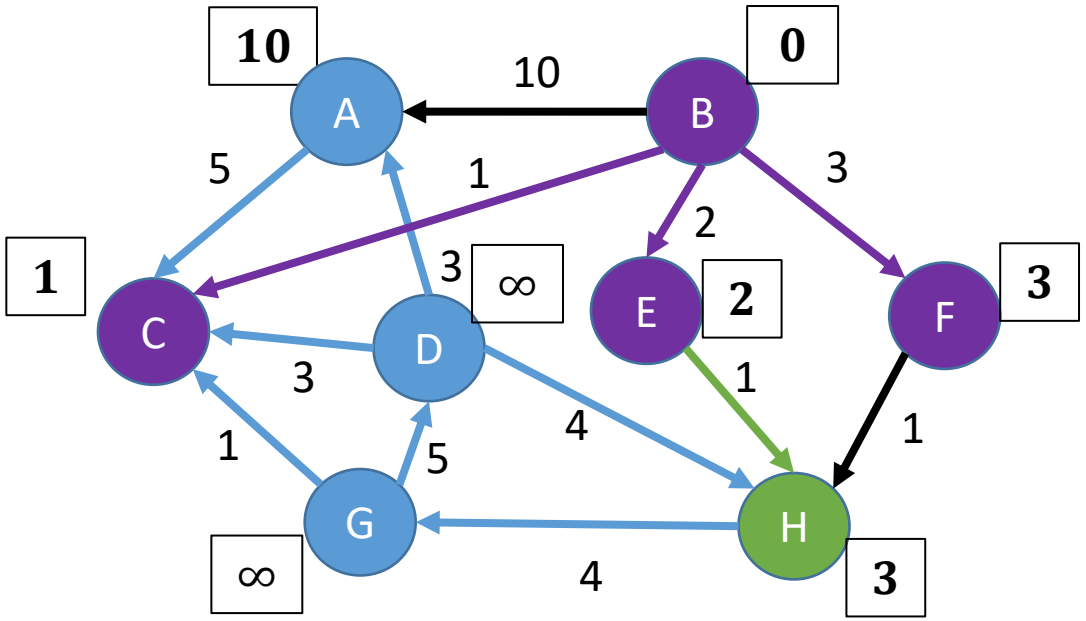




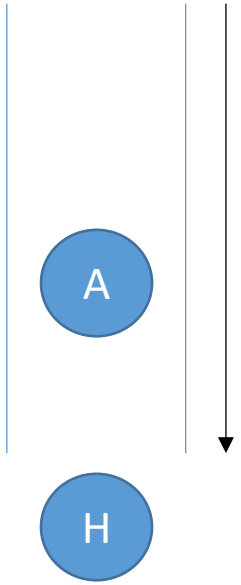
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = 3

Dijkstra's Algorithm

- We move on to the next vertex in the priority queue



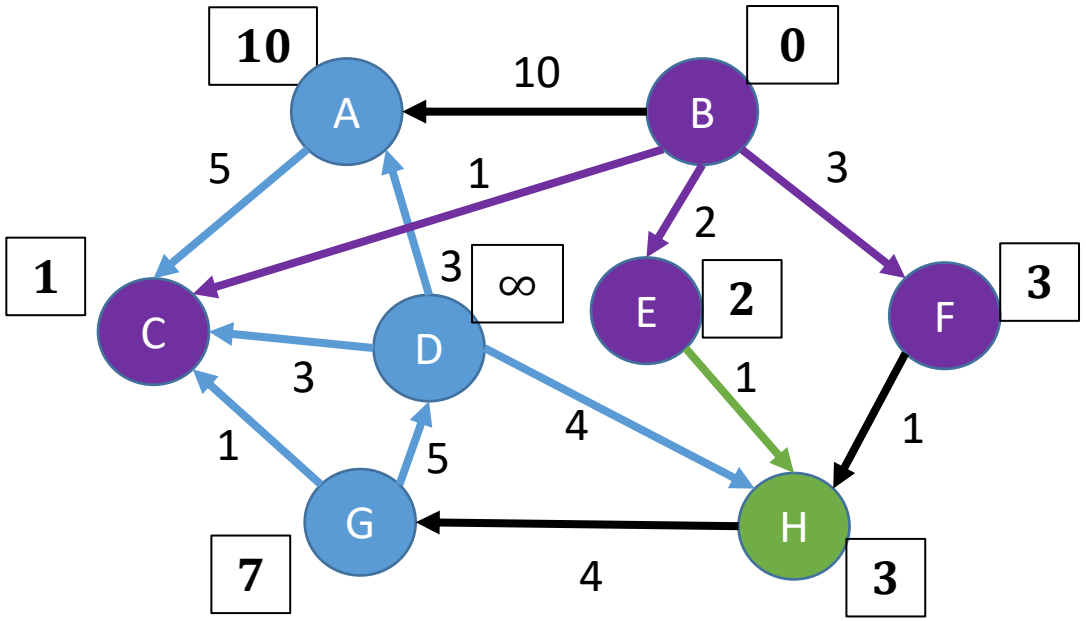
Priority Queue/Min-heap



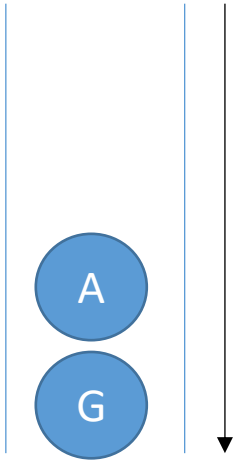
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = ∞
H cost = 3

Dijkstra's Algorithm

- We'll now look at the vertices adjacent to H.
  - Cost of each is H's cost + cost of the edge



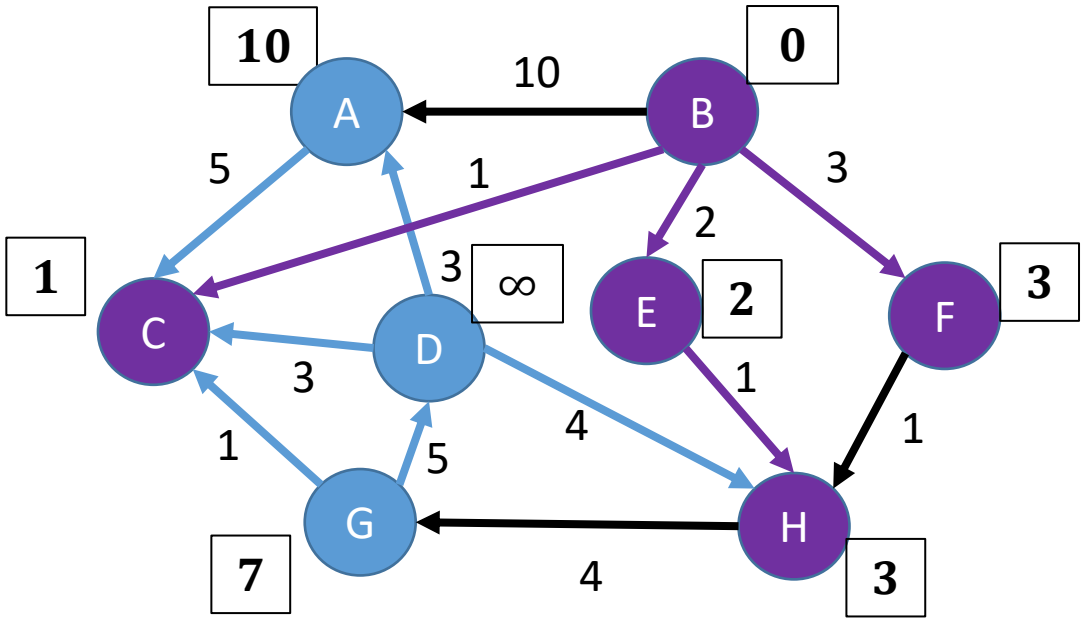
Priority Queue/Min-heap



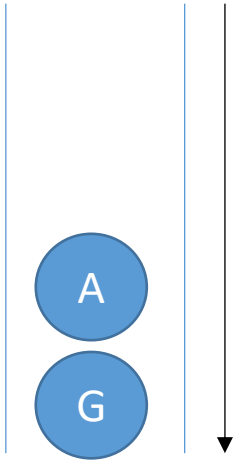
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = 3 + 4 = 7
H cost = 3

Dijkstra's Algorithm

- We are finished with H.



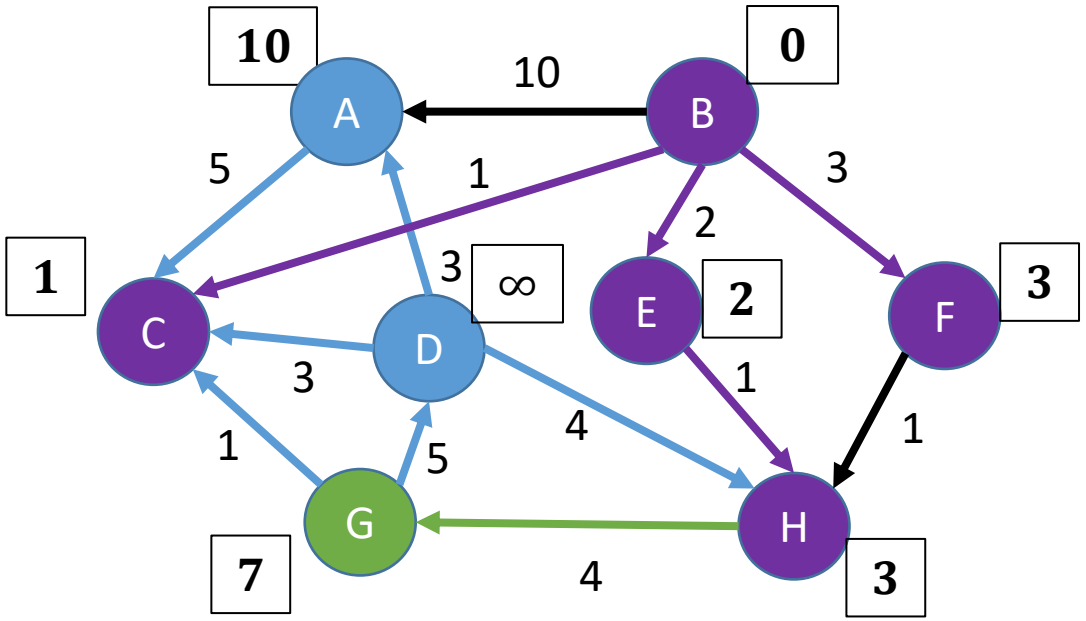
Priority Queue/Min-heap



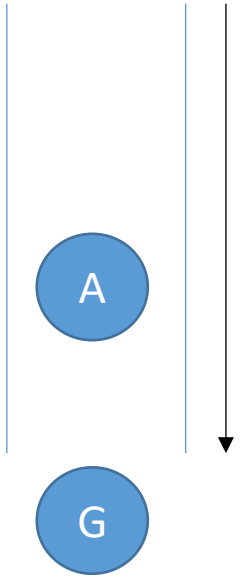
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We move on to the next vertex in the priority queue



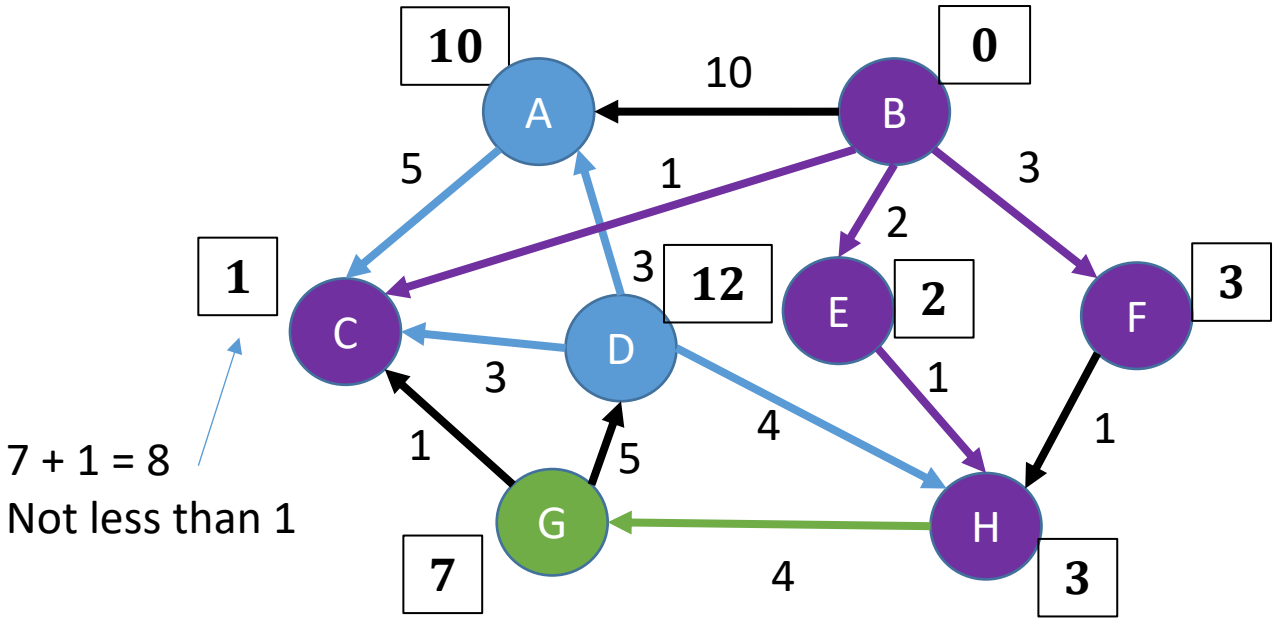
Priority Queue/Min-heap



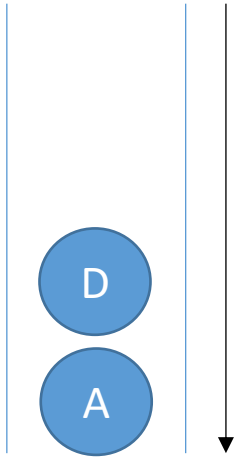
A cost = 10
B cost = 0
C cost = 1
D cost = ∞
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We'll now look at the vertices adjacent to G.
  - Cost of each is G's cost + cost of the edge



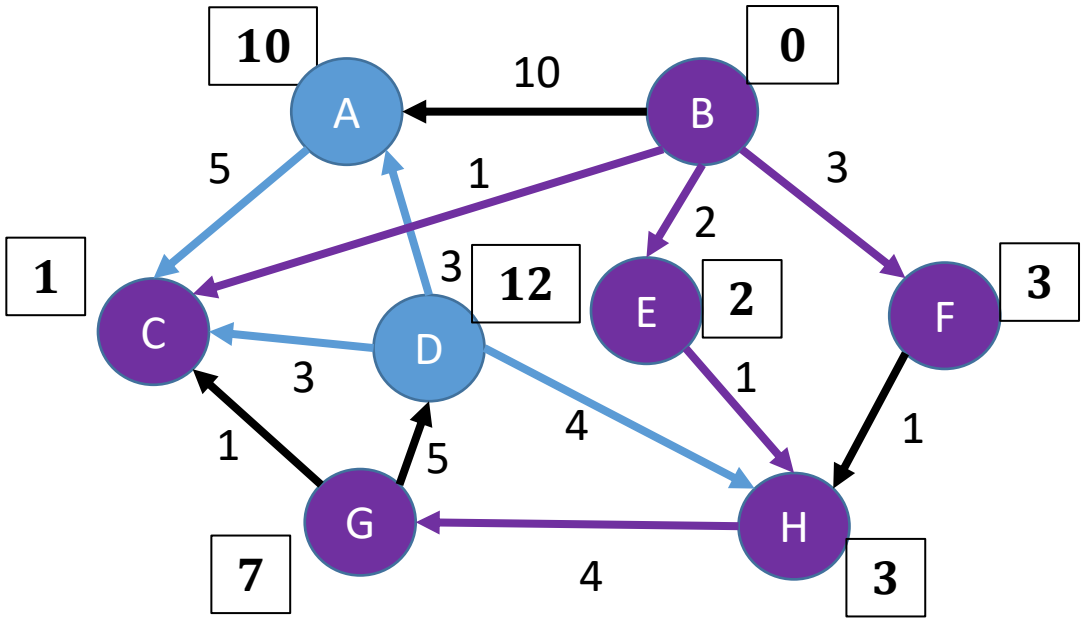
Priority Queue/Min-heap



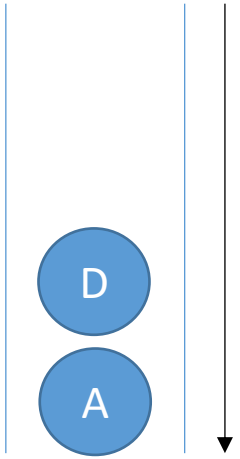
A cost = 10
B cost = 0
C cost = 1
D cost = 7 + 5 = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We are finished with G



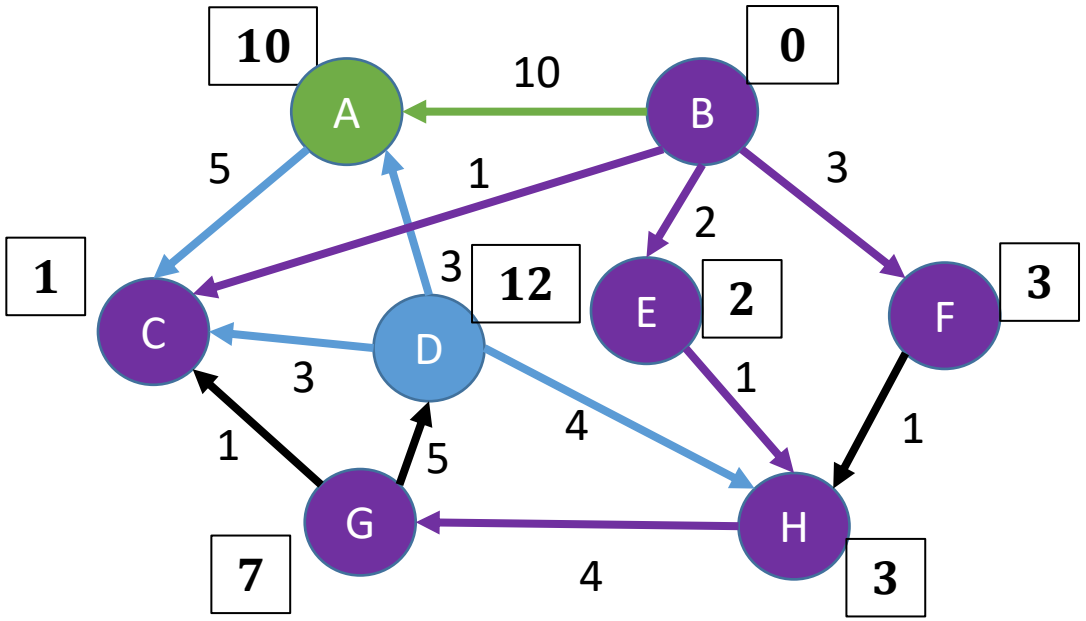
Priority Queue/Min-heap



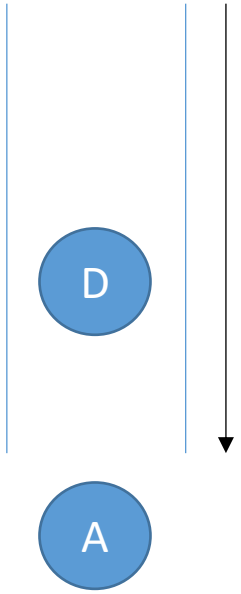
A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We move on to the next node in the priority queue



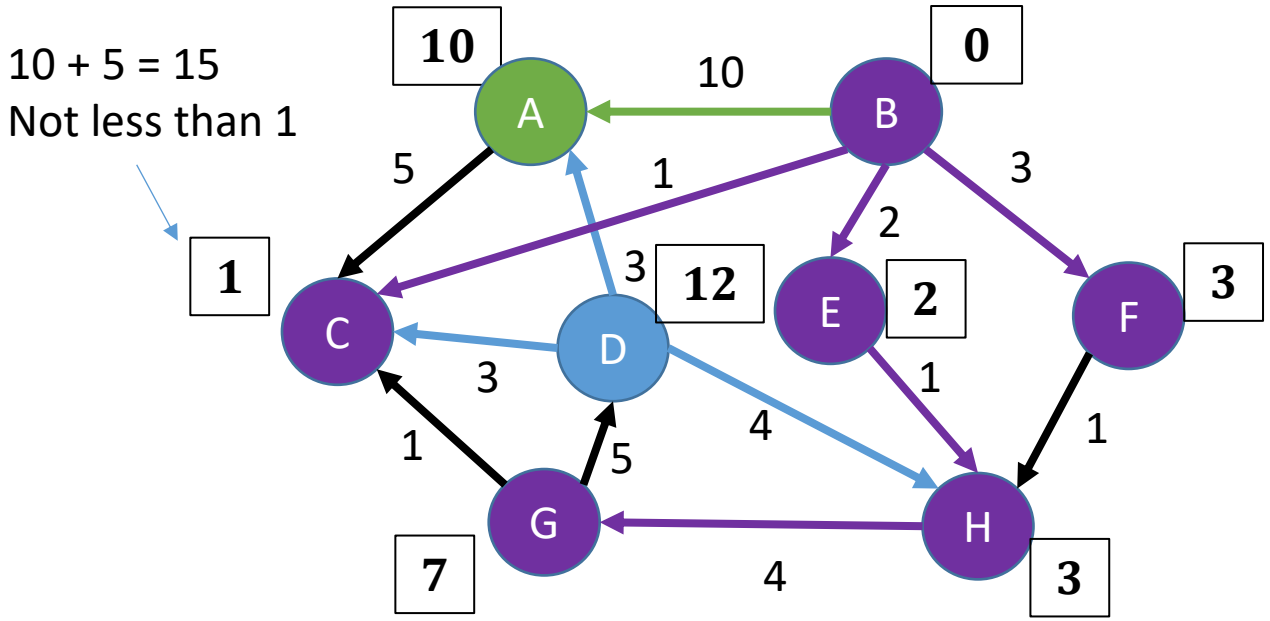
Priority Queue/Min-heap



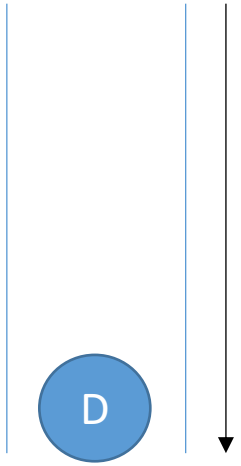
A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We'll now look at the vertices adjacent to A.
  - Cost of each is A's cost + cost of the edge



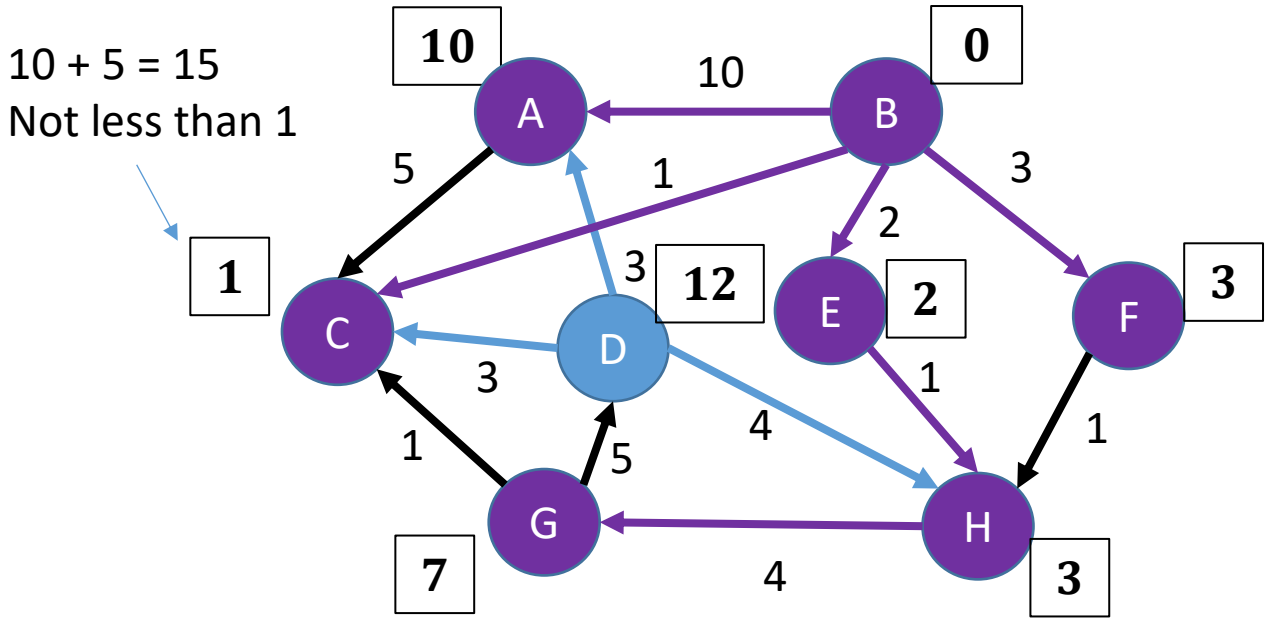
Priority Queue/Min-heap



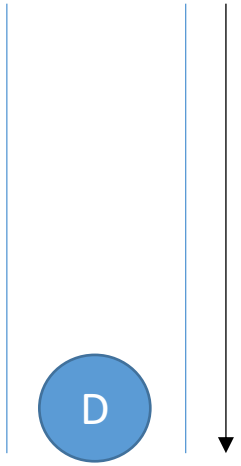
A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We are finished with A



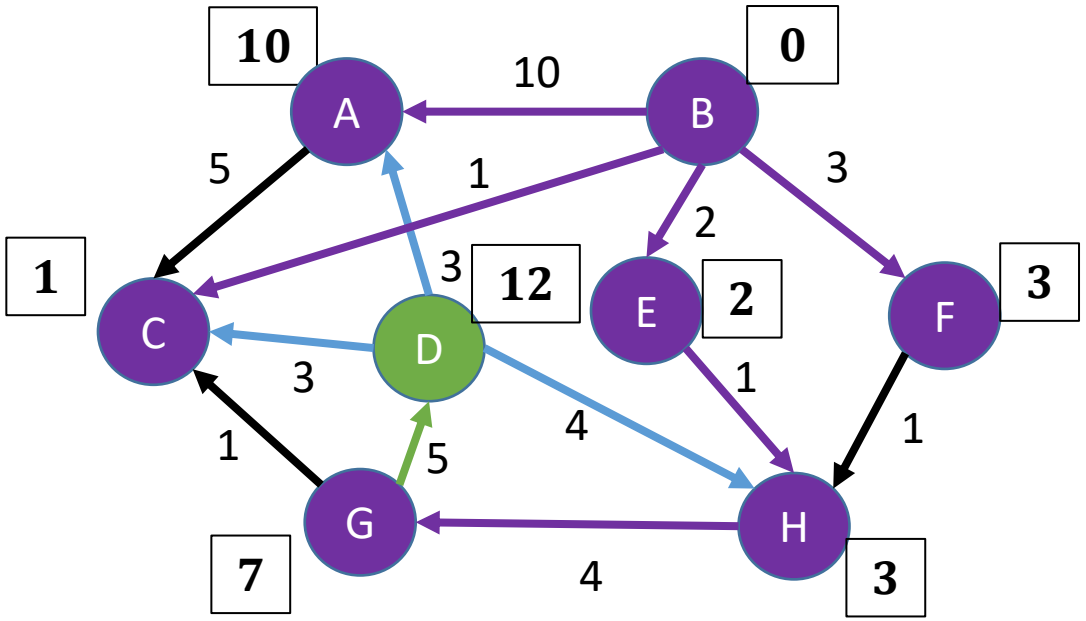
Priority Queue/Min-heap



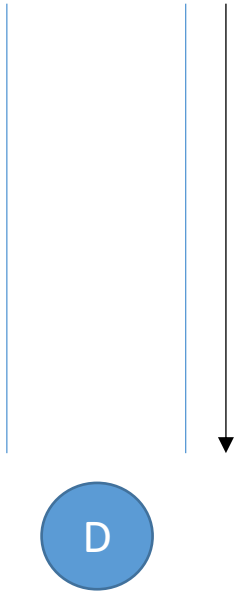
A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We move on to the next node in the priority queue



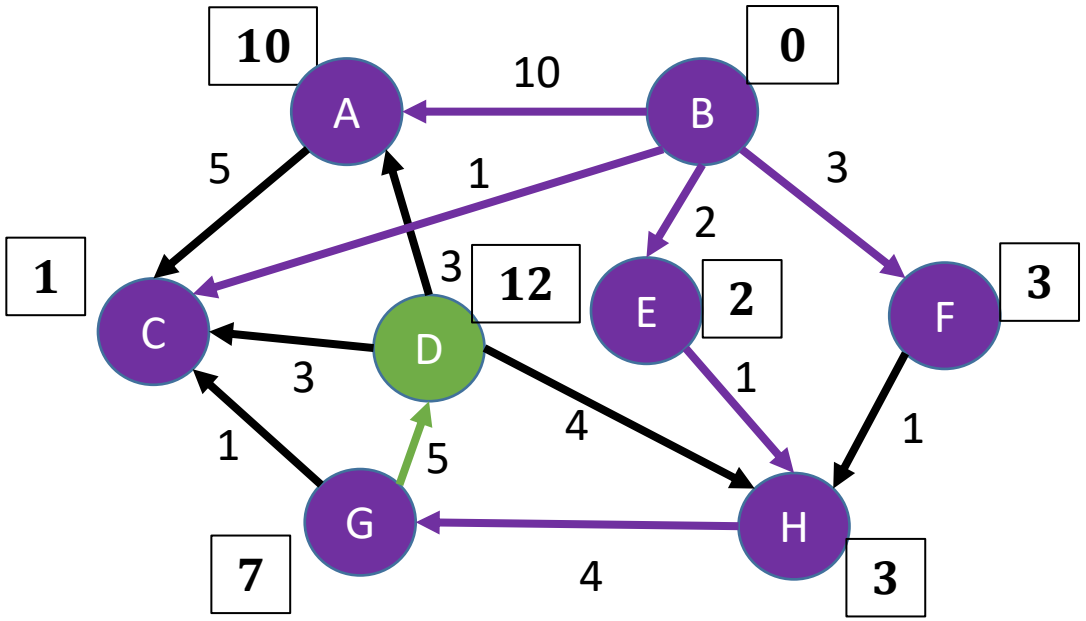
Priority Queue/Min-heap



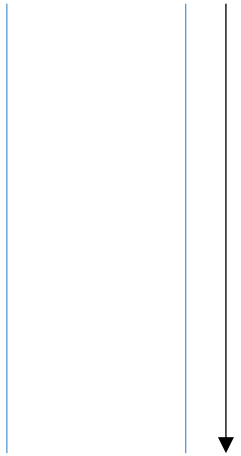
A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We'll now look at the vertices adjacent to D.
  - Cost of each is D's cost + cost of the edge



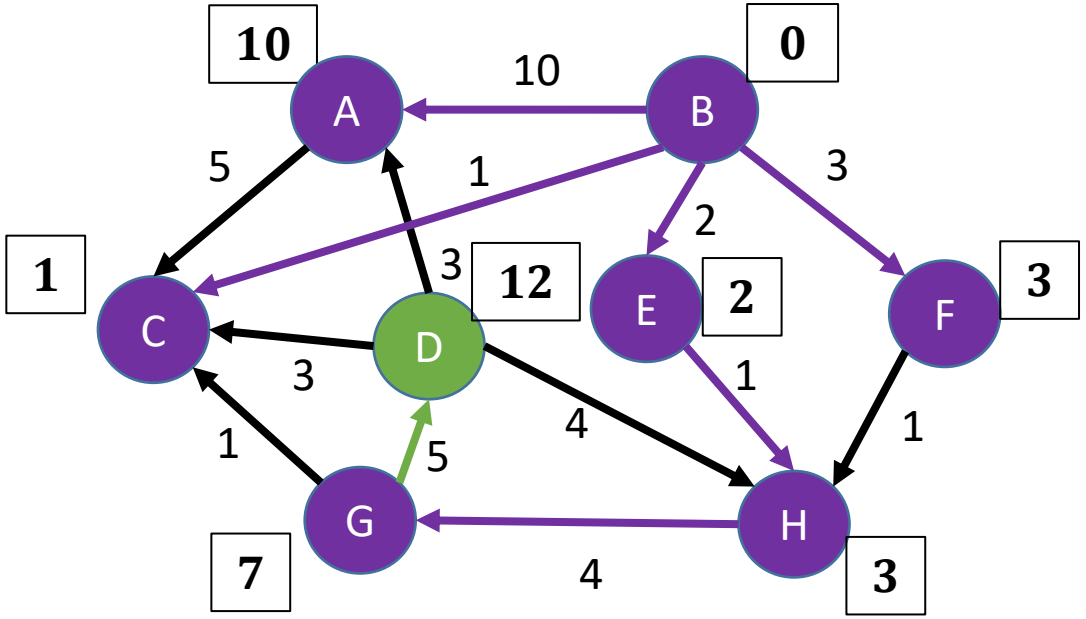
Priority Queue/Min-heap



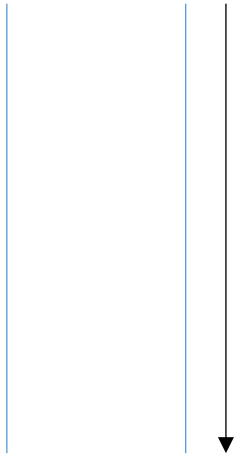
A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- D -> A = 12 + 3 = 15 (Not less than 10)
- D -> C = 12 + 3 = 15 (Not less than 1)
- D -> H = 12 + 4 = 16 (Not less than 3)



Priority Queue/Min-heap

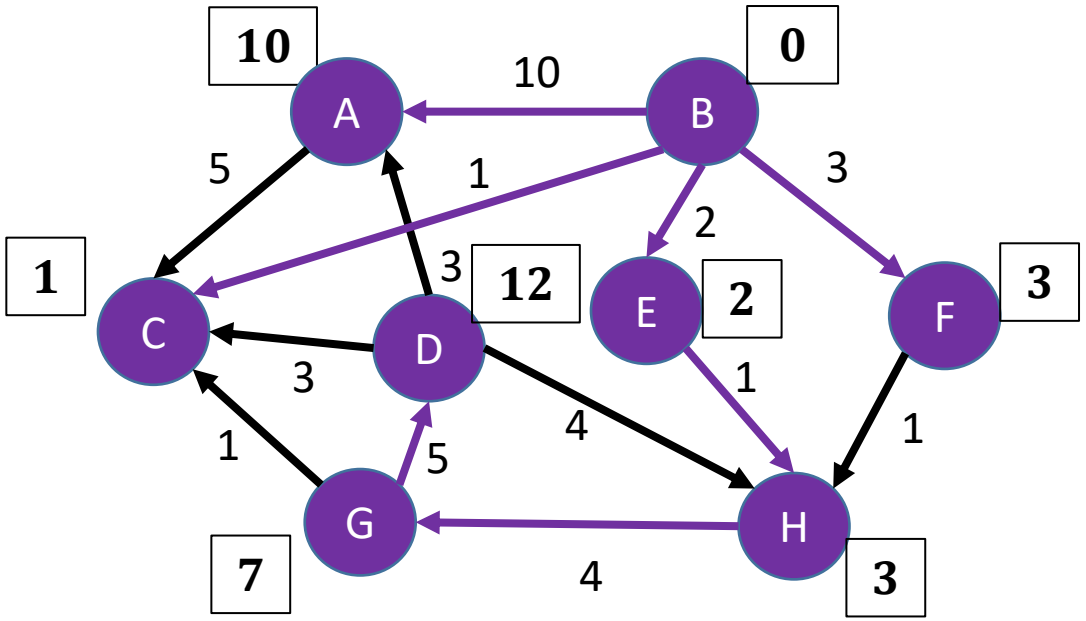


A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- We are finished with D

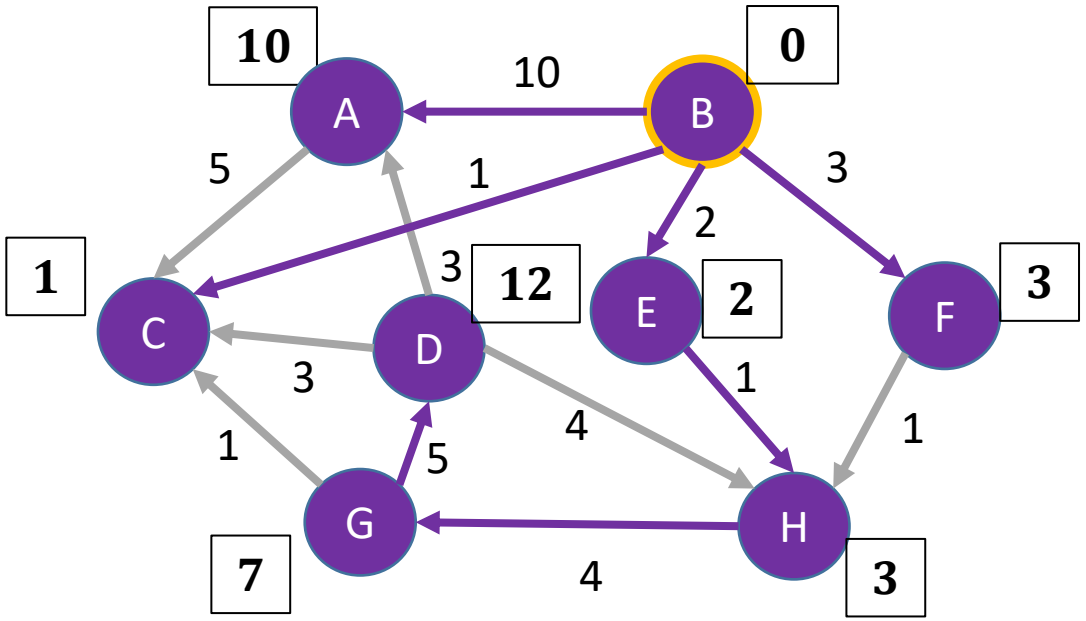
Priority Queue/Min-heap



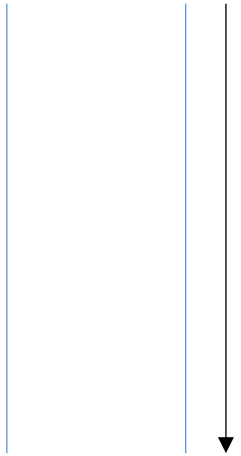
A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3

Dijkstra's Algorithm

- Priority Queue is empty
  - Algorithm is complete



Priority Queue/Min-heap



Least cost from B to all other vertices:

A cost = 10
B cost = 0
C cost = 1
D cost = 12
E cost = 2
F cost = 3
G cost = 7
H cost = 3