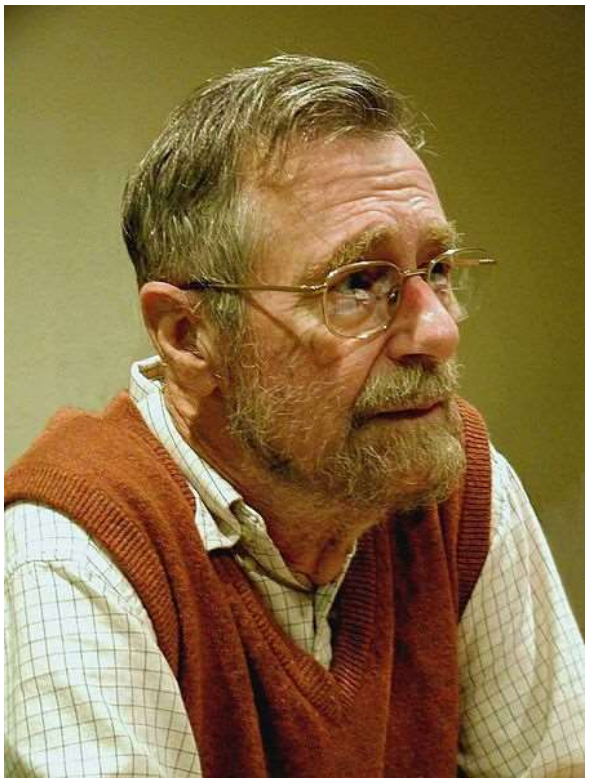
Graphs and Graph Traversals

Mahdi Ebrahimi

Summer 2020

Dijkstra's Algorithm—The Man



Named after its inventor Edsger Dijkstra (1930-2002)

Truly one of the "founders" of computer science

This is just one of his many contributions

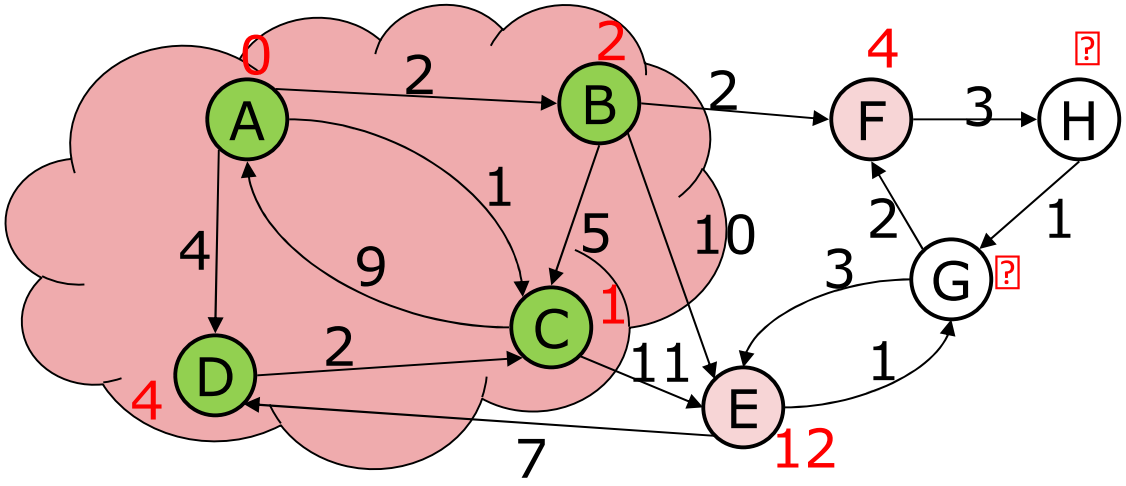
"Computer science is no more about computers than astronomy is about telescopes"

Dijkstra's Algorithm—The Idea

His algorithm is similar to BFS, but adapted to handle weights

- A priority queue will prove useful for efficiency
- Grow set of nodes whose shortest distance has been computed
- Nodes not in the set will have a "best distance so far"

Dijkstra's Algorithm—The Cloud



Initial State:

- Start node has cost 0
- All other nodes have cost ∞

At each step:

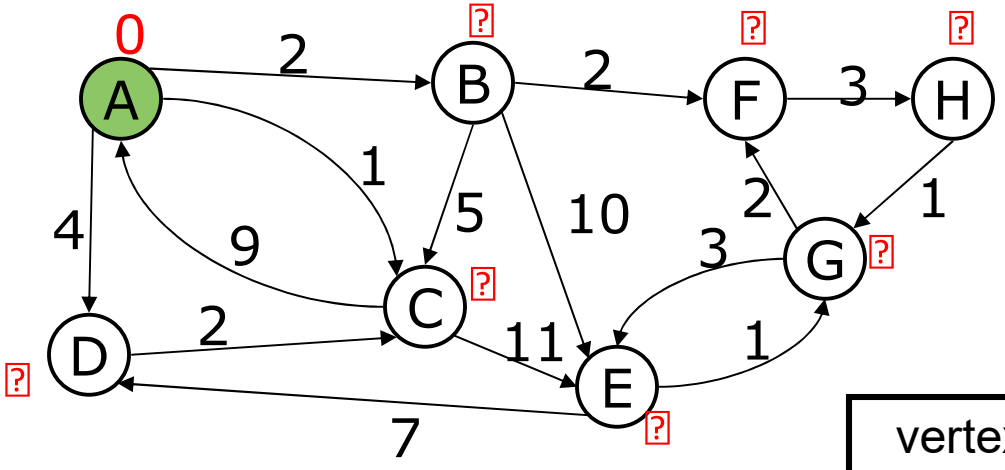
- Pick closest unknown vertex v
- Add it to the "cloud" of known vertices
- Update distances for nodes with edges from v

The Algorithm

1. For each node **v**≠**source**,
   Set **v**.*cost* = ∞ and **v**.*known* = false
2. Set **source**.*cost* = 0 and **source**.*known* = true
3. While there are unknown nodes in the graph
   a) Select the unknown node **v** with lowest cost
   b) Mark **v** as known
   c) For each edge (**v**, **u**) with weight **w**,

```
c1 = v.cost + w       // cost of best path through v to u
c2 = u.cost   // cost of best path to u previously known
if(c1 < c2)                  // if the path through v is better
    u.cost = c1
    u.path = v               // for computing actual paths
```

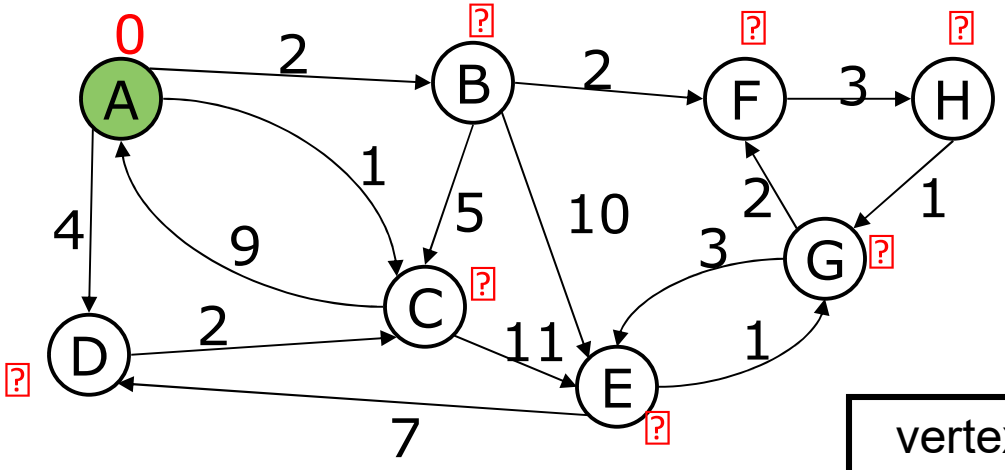
Example #1



Order Added to Known Set:

vertex	known?	cost	path
A			
B			
C			
D			
E			
F			
G			
H			

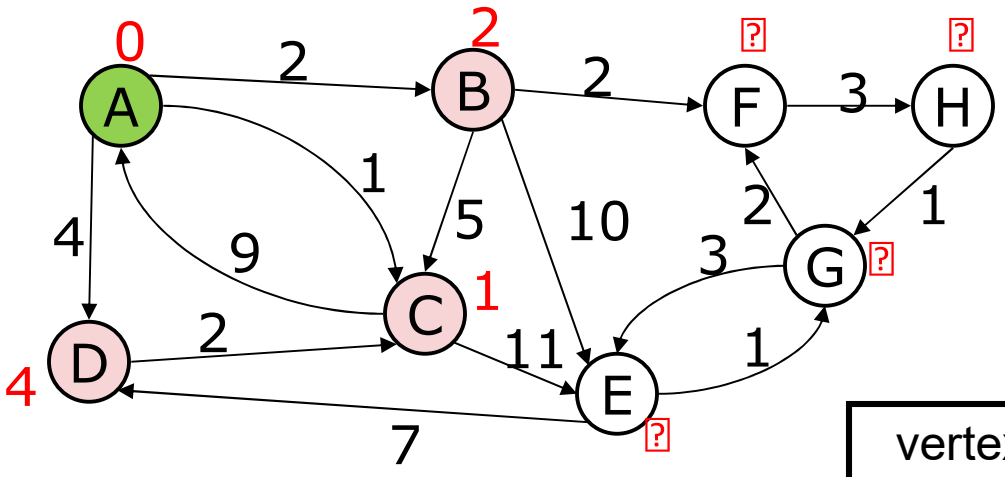
Example #1



Order Added to Known Set:

vertex	known?	cost	path
A		0	
B		??	
C		??	
D		??	
E		??	
F		??	
G		??	
H		??	

Example #1

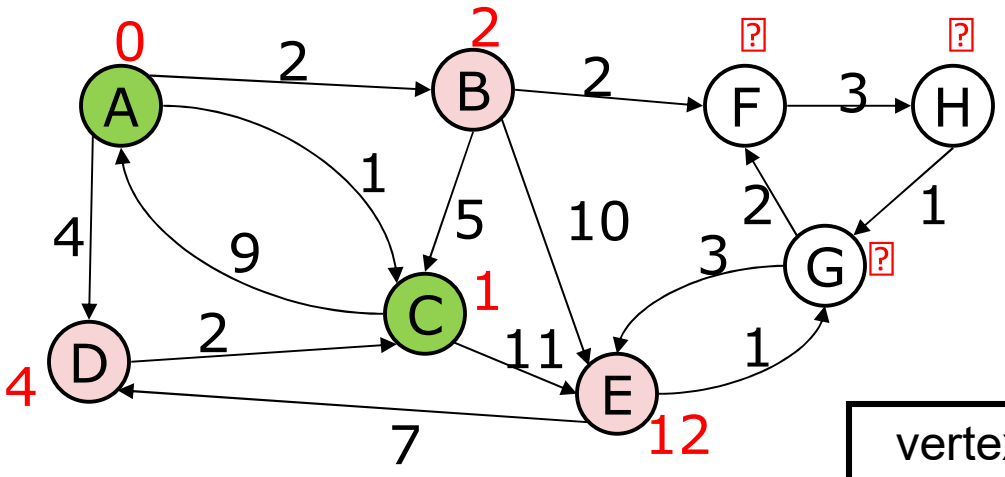


vertex	known?	cost	path
A	Y	0	
B		≤ 2	A
C		≤ 1	A
D		≤ 4	A
E		??	
F		??	
G		??	
H		??	

Order Added to Known Set:

A

Example #1

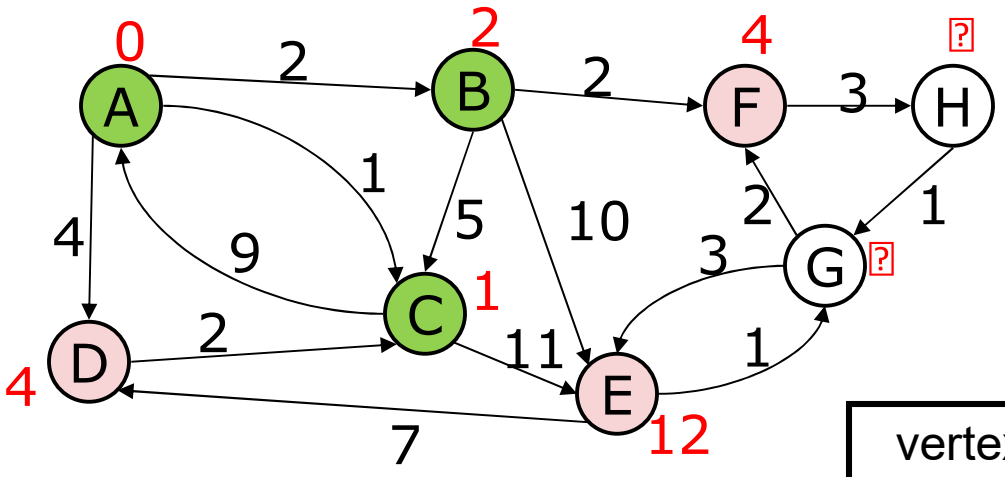


vertex	known?	cost	path
A	Y	0	
B		≤ 2	A
C	Y	1	A
D		≤ 4	A
E		≤ 12	C
F		??	
G		??	
H		??	

Order Added to Known Set:

A, C

Example #1

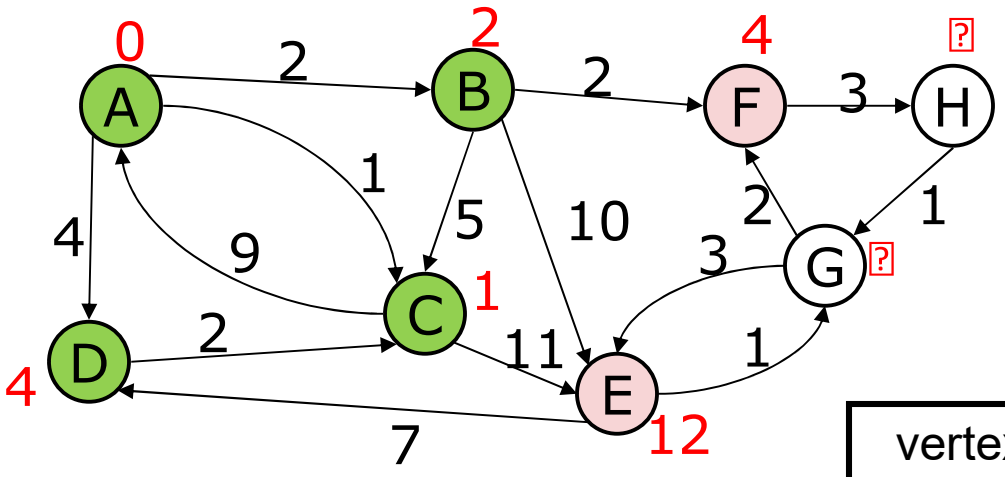


Order Added to Known Set:

A, C, B

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D		≤ 4	A
E		≤ 12	C
F		≤ 4	B
G		??	
H		??	

Example #1

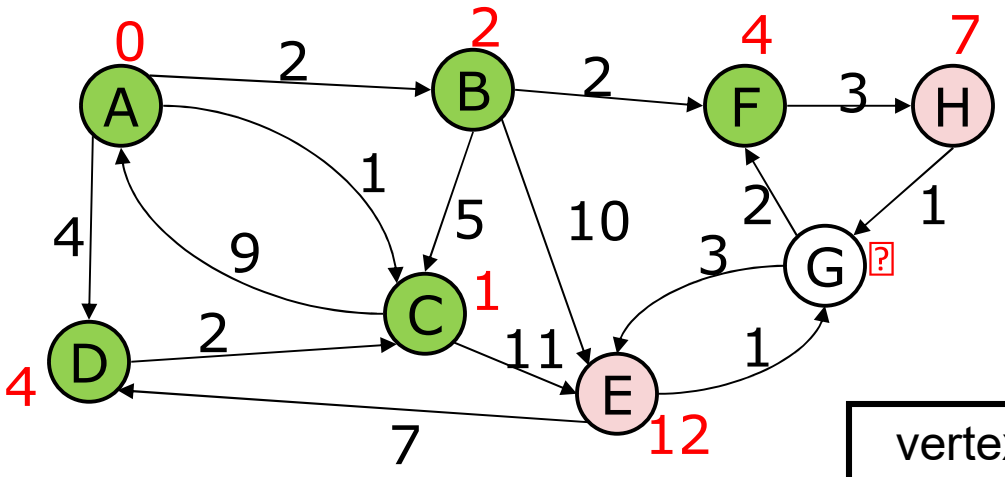


Order Added to Known Set:

A, C, B, D

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D	Y	4	A
E		≤ 12	C
F		≤ 4	B
G		??	
H		??	

Example #1

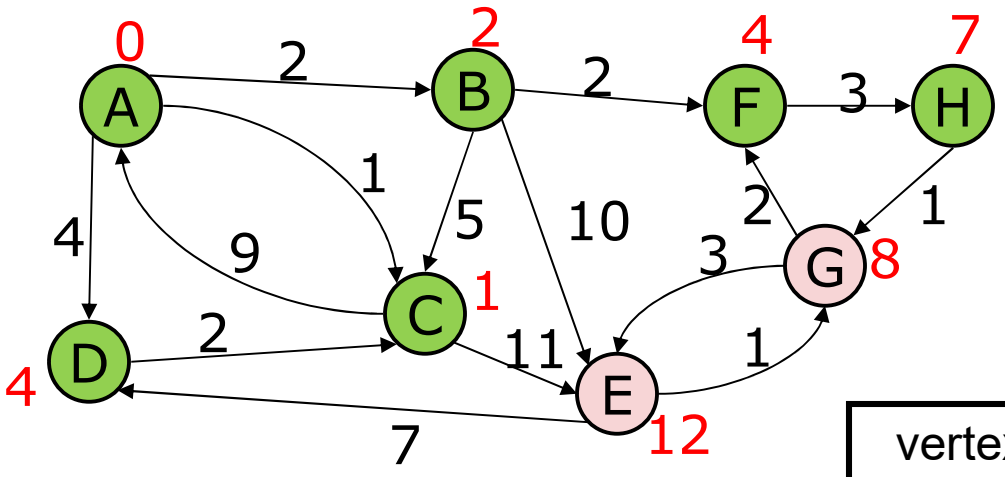


Order Added to Known Set:

A, C, B, D, F

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D	Y	4	A
E		≤ 12	C
F	Y	4	B
G		??	
H		≤ 7	F

Example #1

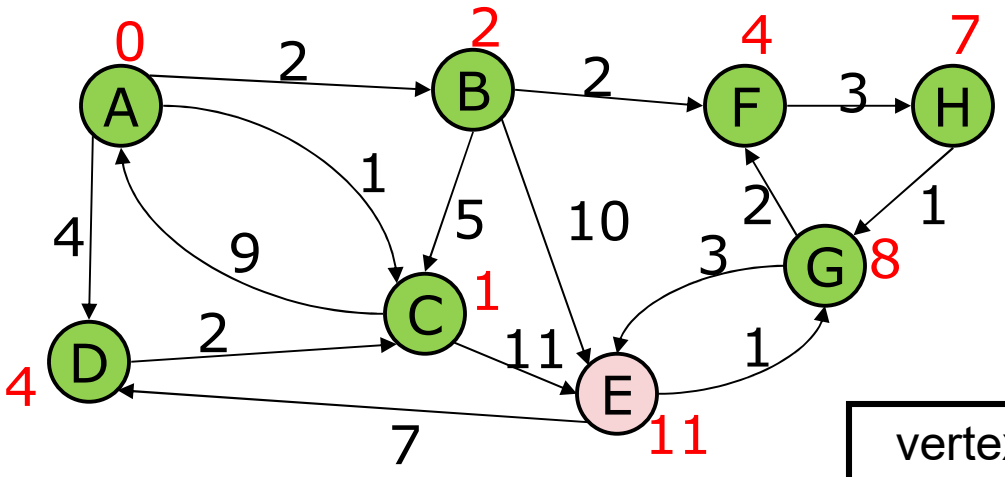


Order Added to Known Set:

A, C, B, D, F, H

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D	Y	4	A
E		≤ 12	C
F	Y	4	B
G		≤ 8	H
H	Y	7	F

Example #1

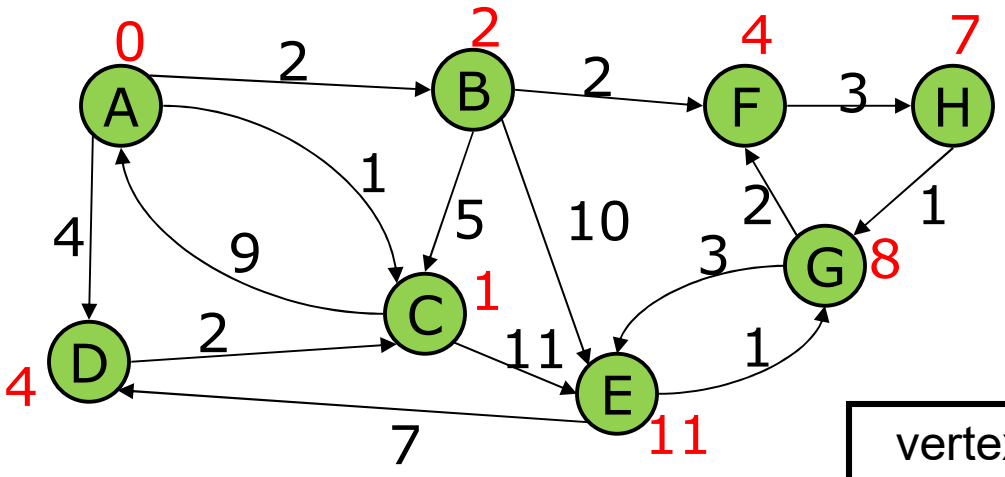


Order Added to Known Set:

A, C, B, D, F, H, G

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D	Y	4	A
E		≤ 11	G
F	Y	4	B
G	Y	8	H
H	Y	7	F

Example #1



Order Added to Known Set:

A, C, B, D, F, H, G, E

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D	Y	4	A
E	Y	11	G
F	Y	4	B
G	Y	8	H
H	Y	7	F

Important Features

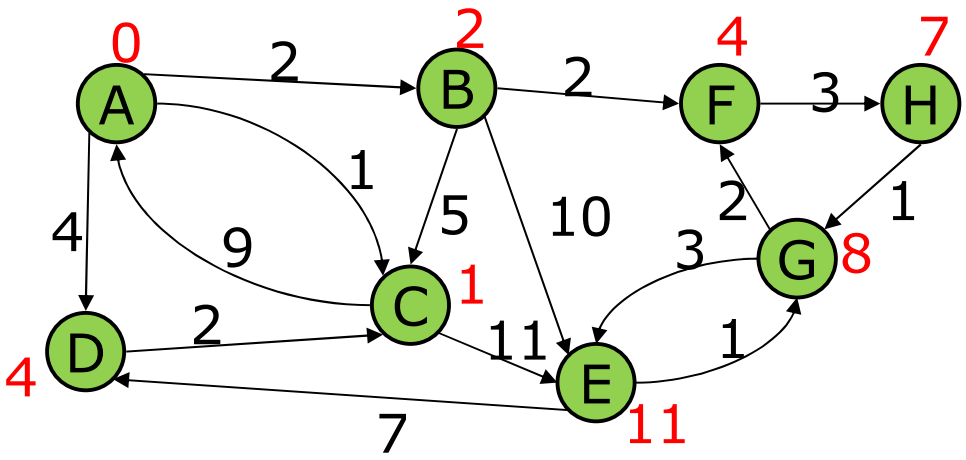
When a vertex is marked known, the cost of the shortest path to that node is known

- The path is also known by following back-pointers

While a vertex is still not known, another shorter path to it might still be found

Interpreting the Results

Now that we're done, how do we get the path from, say, A to E?



Order Added to Known Set:

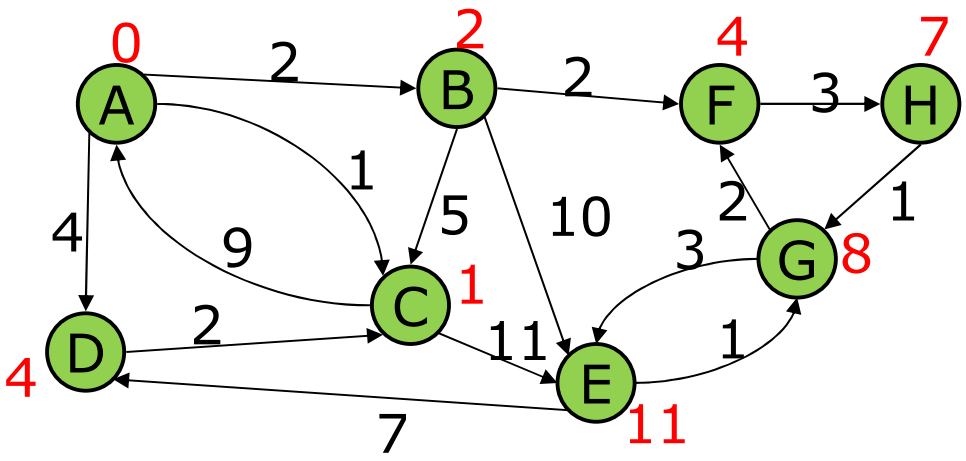
A, C, B, D, F, H, G, E

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D	Y	4	A
E	Y	11	G
F	Y	4	B
G	Y	8	H
H	Y	7	F

Stopping Short

How would this have worked differently if we were only interested in:

- the path from A to G?
- the path from A to E?

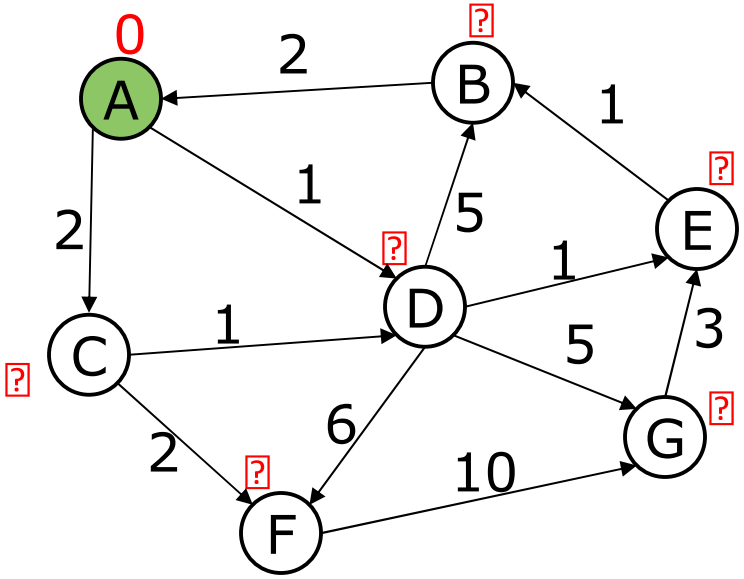


Order Added to Known Set:

A, C, B, D, F, H, G, E

vertex	known?	cost	path
A	Y	0	
B	Y	2	A
C	Y	1	A
D	Y	4	A
E	Y	11	G
F	Y	4	B
G	Y	8	H
H	Y	7	F

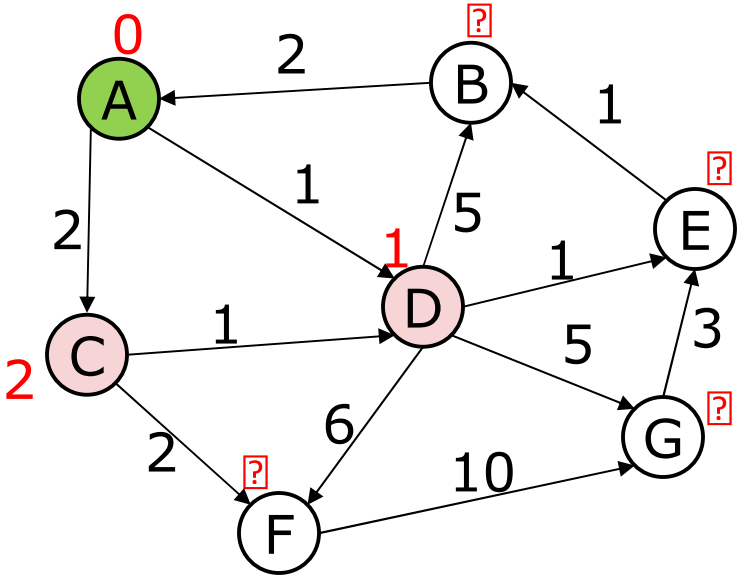
Example #2



Order Added to Known Set:

vertex	known?	cost	path
A		0	
B		??	
C		??	
D		??	
E		??	
F		??	
G		??	

Example #2

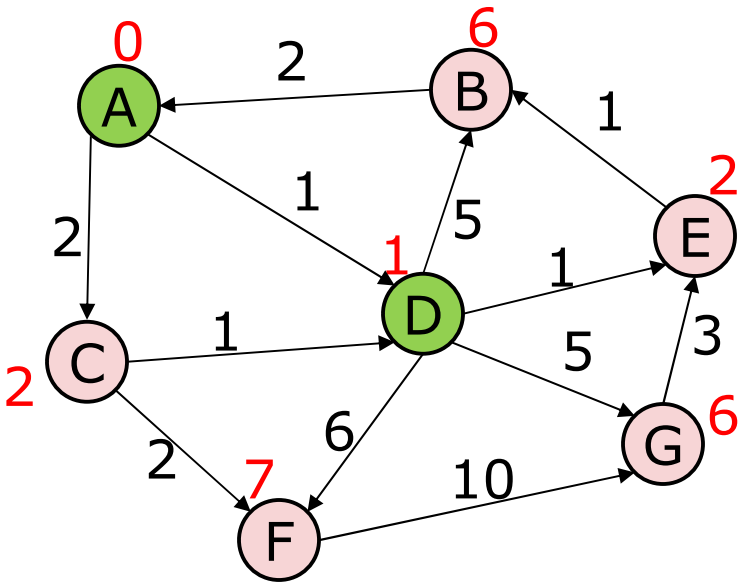


Order Added to Known Set:

A

vertex	known?	cost	path
A	Y	0	
B		??	
C		≤ 2	A
D		≤ 1	A
E		??	
F		??	
G		??	

Example #2

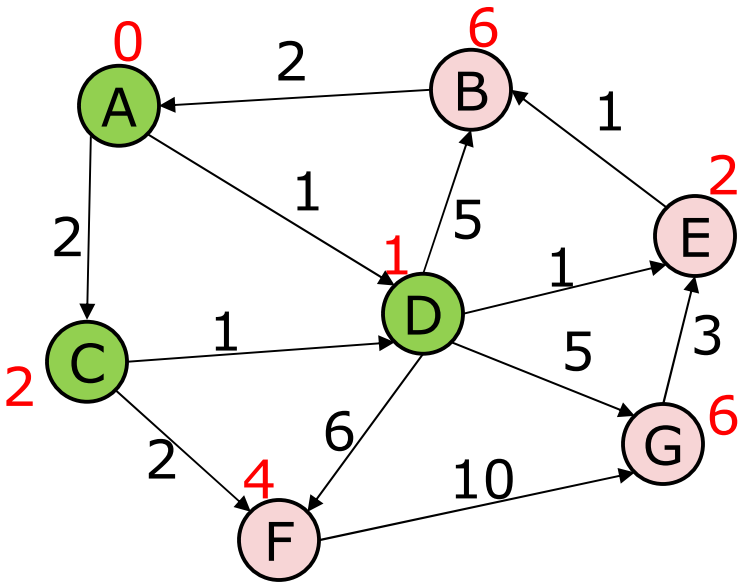


Order Added to Known Set:

A, D

vertex	known?	cost	path
A	Y	0	
B		≤ 6	D
C		≤ 2	A
D	Y	1	A
E		≤ 2	D
F		≤ 7	D
G		≤ 6	D

Example #2

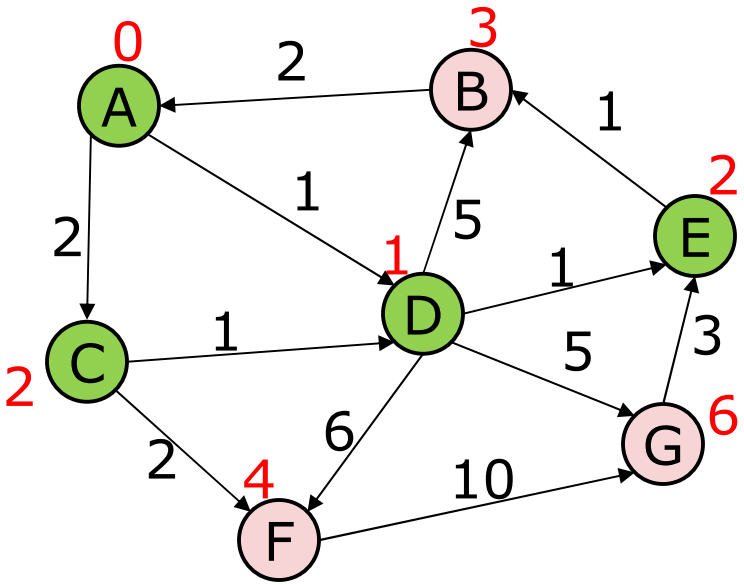


Order Added to Known Set:

A, D, C

vertex	known?	cost	path
A	Y	0	
B		≤ 6	D
C	Y	2	A
D	Y	1	A
E		≤ 2	D
F		≤ 4	C
G		≤ 6	D

Example #2

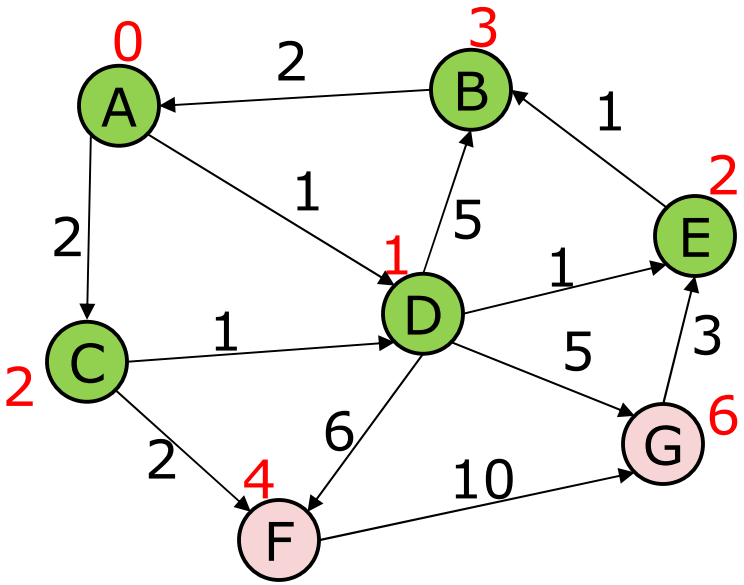


Order Added to Known Set:

A, D, C, E

vertex	known?	cost	path
A	Y	0	
B		≤ 3	E
C	Y	2	A
D	Y	1	A
E	Y	2	D
F		≤ 4	C
G		≤ 6	D

Example #2

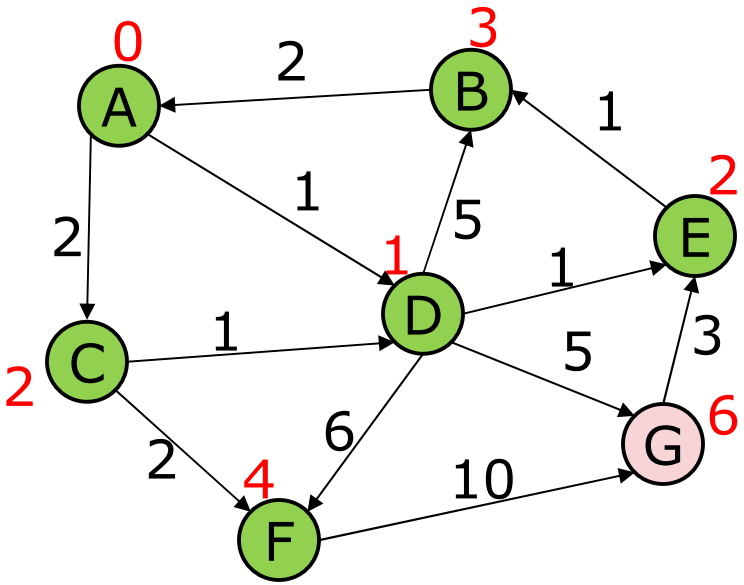


Order Added to Known Set:

A, D, C, E, B

vertex	known?	cost	path
A	Y	0	
B	Y	3	E
C	Y	2	A
D	Y	1	A
E	Y	2	D
F		≤ 4	C
G		≤ 6	D

Example #2

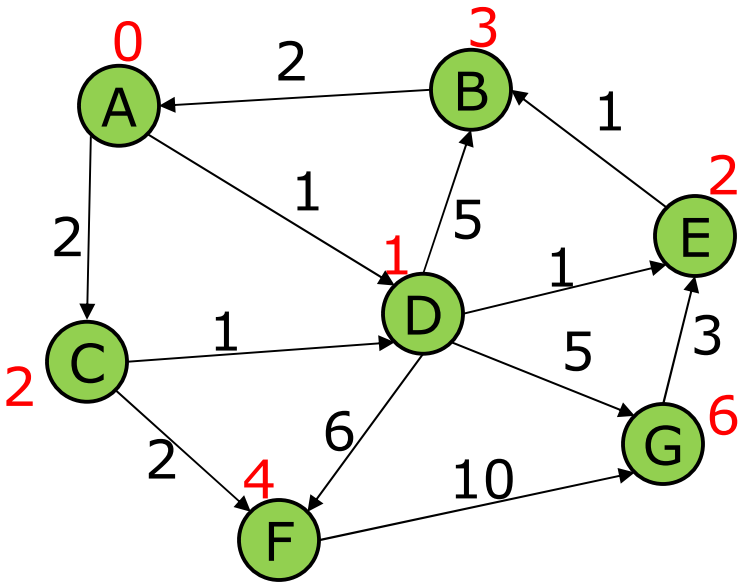


Order Added to Known Set:

A, D, C, E, B, F

vertex	known?	cost	path
A	Y	0	
B	Y	3	E
C	Y	2	A
D	Y	1	A
E	Y	2	D
F	Y	4	C
G		≤ 6	D

Example #2

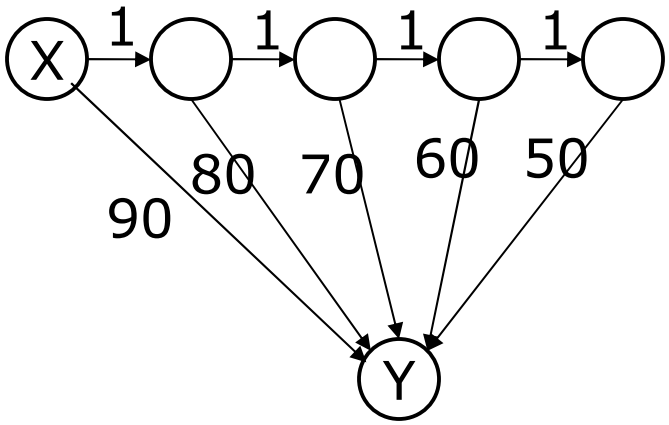


Order Added to Known Set:

A, D, C, E, B, F, G

vertex	known?	cost	path
A	Y	0	
B	Y	3	E
C	Y	2	A
D	Y	1	A
E	Y	2	D
F	Y	4	C
G	Y	6	D

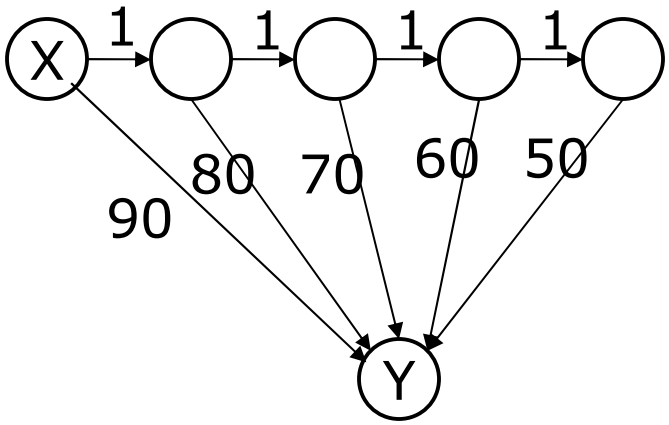
Example #3



How will the best-cost-so-far for Y proceed?

Is this expensive?

Example #3



How will the best-cost-so-far for Y proceed?
90, 81, 72, 63, 54

Is this expensive?
No, each *edge* is processed only once

A Greedy Algorithm

Dijkstra's algorithm is an example of a greedy algorithm:

- At each step, irrevocably does what seems best at that step
  - Once a vertex is in the known set, does not go back and readjust its decision
- Locally optimal
  - Does not always mean globally optimal

Have described Dijkstra's algorithm

- For single-source shortest paths in a weighted graph (directed or undirected) with no negative-weight edges