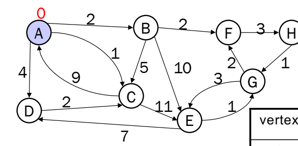
The Algorithm

1. For each node **v**, set **v.cost = ∞** and **v.known = false**
2. Set **source.cost = 0**
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**, if **u** is unknown,

```
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
}
```

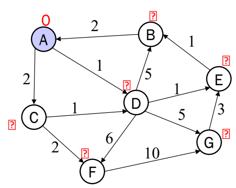

Example #1



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

<u>Order Added to Known Set:</u>

Example #2



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

<u>Order Added to Known Set:</u>

Efficiency, first approach

Use pseudocode to determine asymptotic run-time
- Notice each edge is processed only once

```
dijkstra(Graph G, Node start) {
  for each node: x.cost=infinity, x.known=false
  start.cost = 0
  while(not all nodes are known) {
    b = find unknown node with smallest cost
    b.known = true
    for each edge (b,a) in G
     if(!a.known)
       if(b.cost + weight((b,a)) < a.cost){
         a.cost = b.cost + weight((b,a))
         a.path = b
       }
}
```

Efficiency, second approach

Use pseudocode to determine asymptotic run-time

```
dijkstra(Graph G, Node start) {
  for each node: x.cost=infinity, x.known=false
  start.cost = 0
  build-heap with all nodes
  while(heap is not empty) {
    b = deleteMin()
    b.known = true
    for each edge (b,a) in G
     if(!a.known)
      if(b.cost + weight((b,a)) < a.cost){
        decreaseKey(a,"new cost – old cost")
        a.path = b
      }
}
```