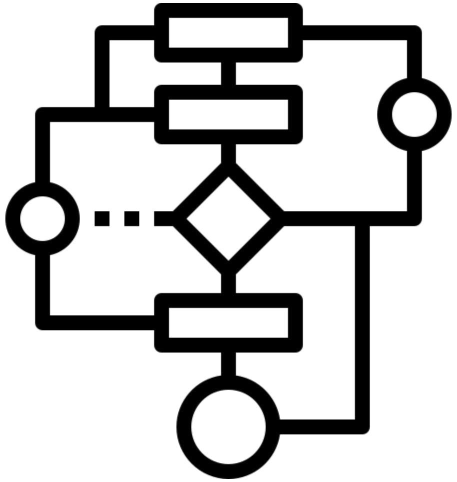
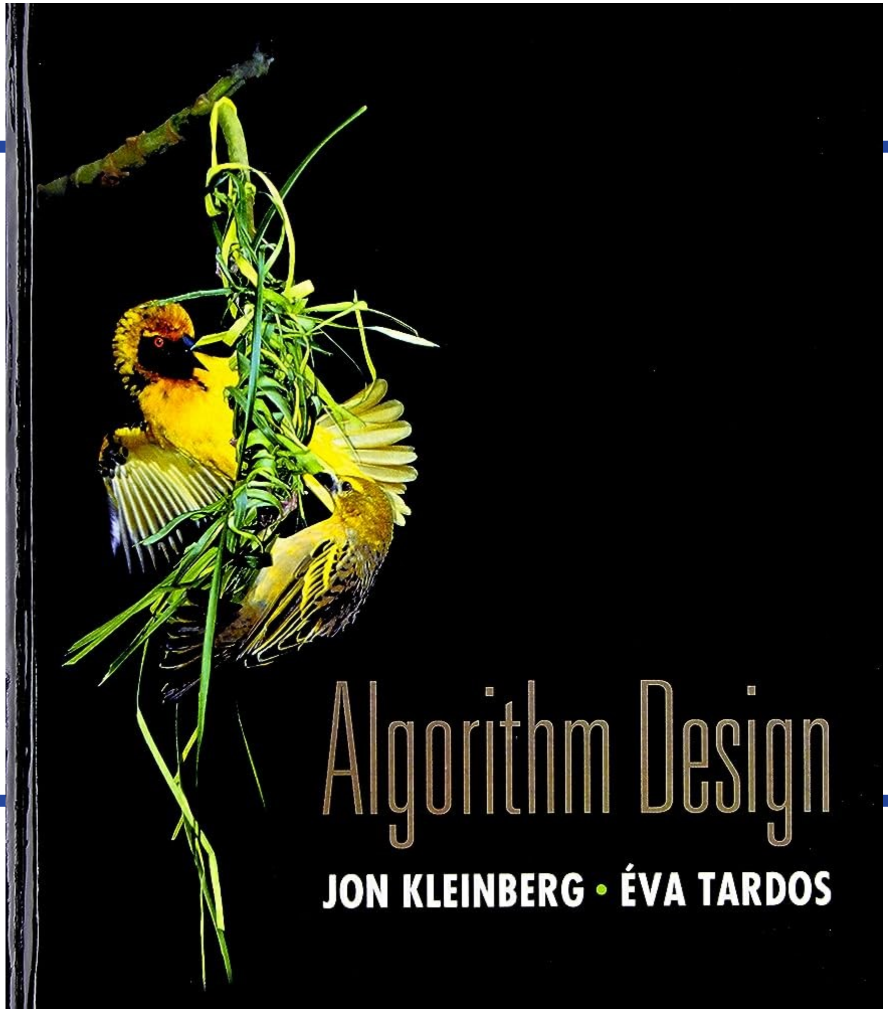


CS 310: Algorithms

Lecture 15

Instructor: Naveed Anwar Bhatti



Chapter 4:
Greedy Algorithms

Section 4.4: Shortest Path Algorithm
Dijkstra Algorithm

Administrivia

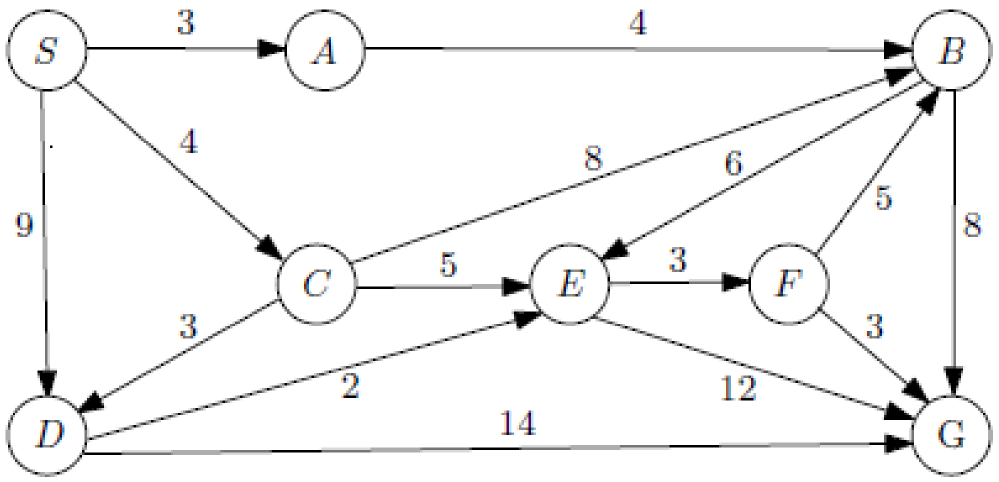
- Midterm marking not complete but **scene bad hai**
- Expect **next three quizzes** on following dates:

Quiz 4	Quiz 5	Quiz 6
6th November	20th November	4th December

- **Assignment 3** will be released soon

Weighted (Directed) Graph

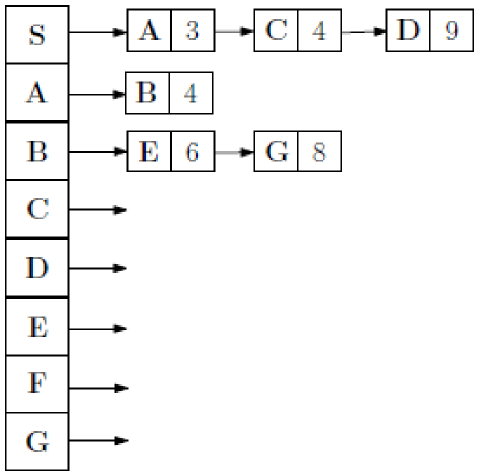
- V : Set of vertices
- E : Set of edges (directed edges)
- w : cost/weight function: $w : E \to \mathbb{R}$
- weights could be lengths, airfare, toll, energy
- Denoted by $G = (V, E, w)$



Weighted Adjacency Matrix

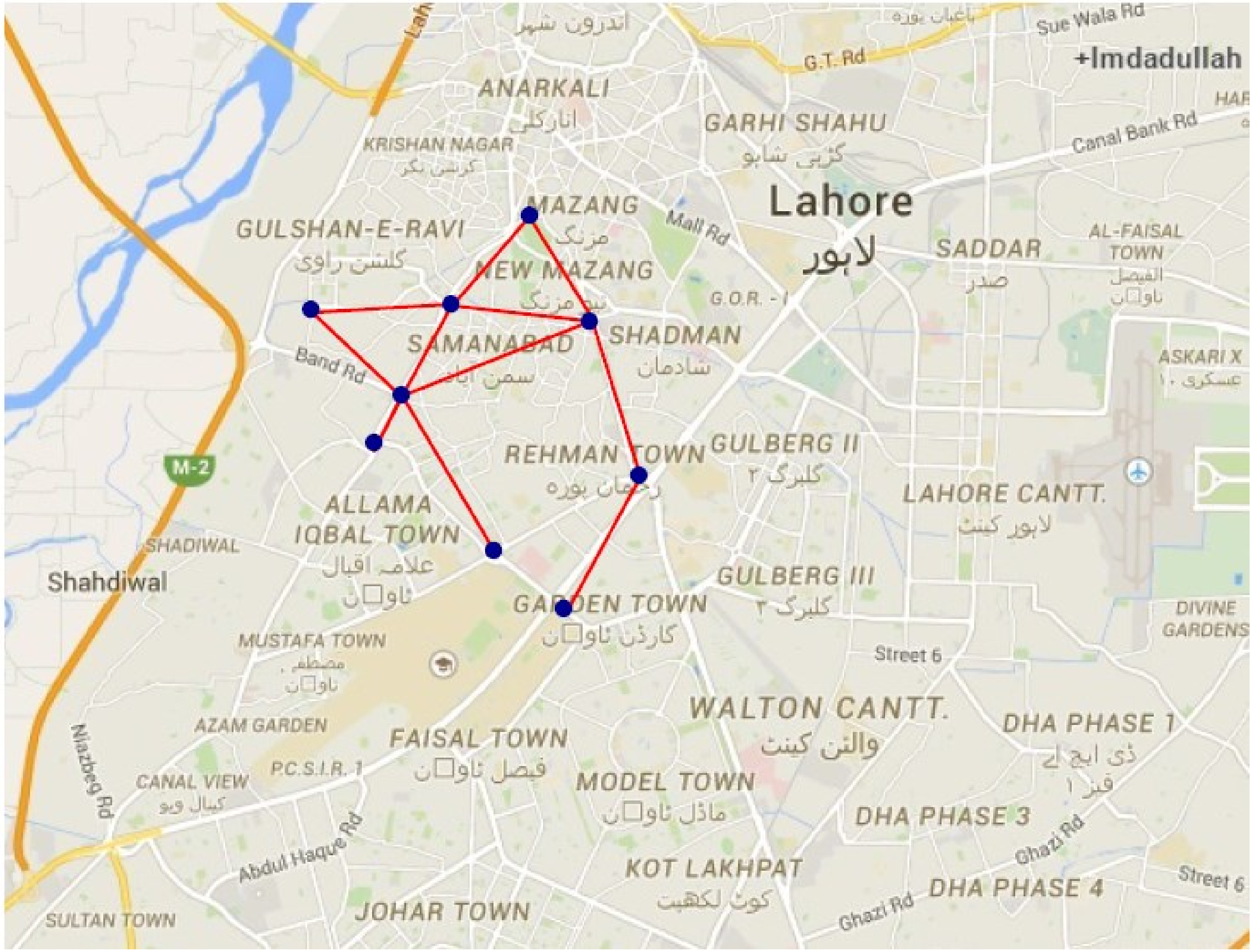
	S	A	B	C	D	E	F	G
S	0	3	0	4	9	0	0	0
A	0	0	4	0	0	0	0	0
B	0	0	0	0	0	6	0	8
C	⋮							
D								
E								
F								
G								

Weighted Adjacency Lists

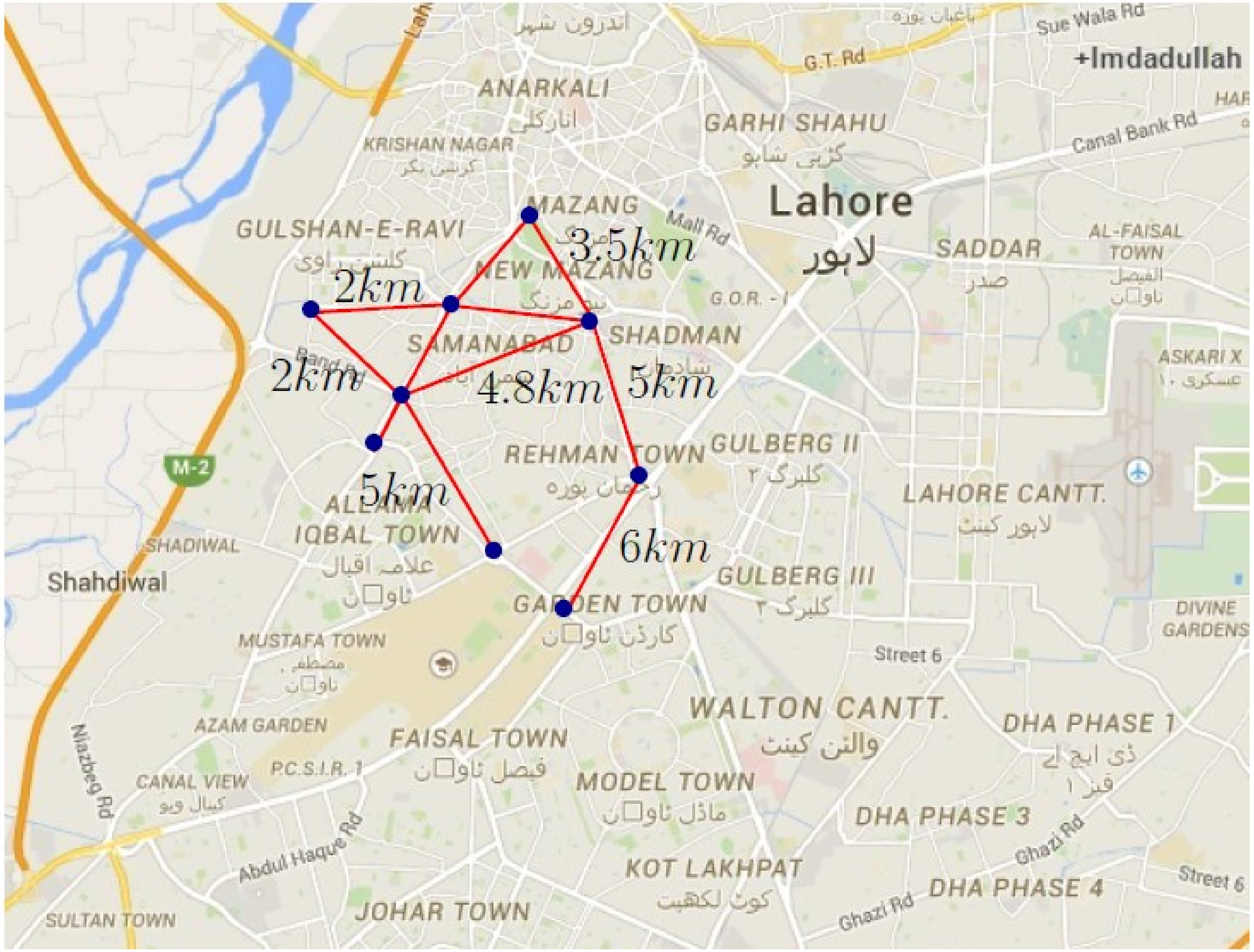


Weighted (Directed) Graph

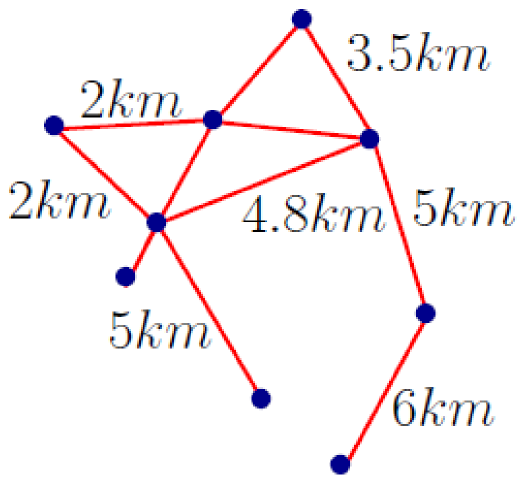
Weighted (Directed) Graph



Weighted (Directed) Graph

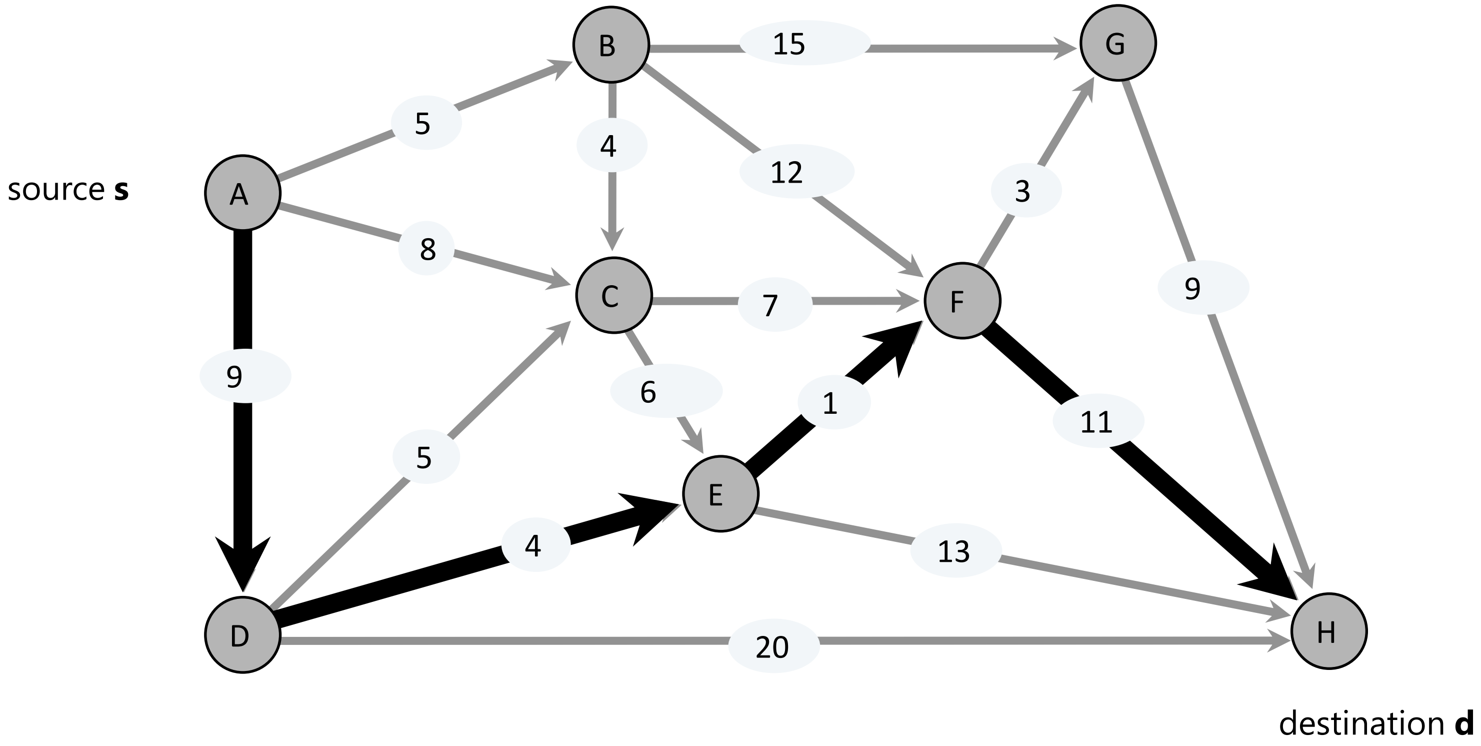


Weighted (Directed) Graph



Single-pair shortest path problem

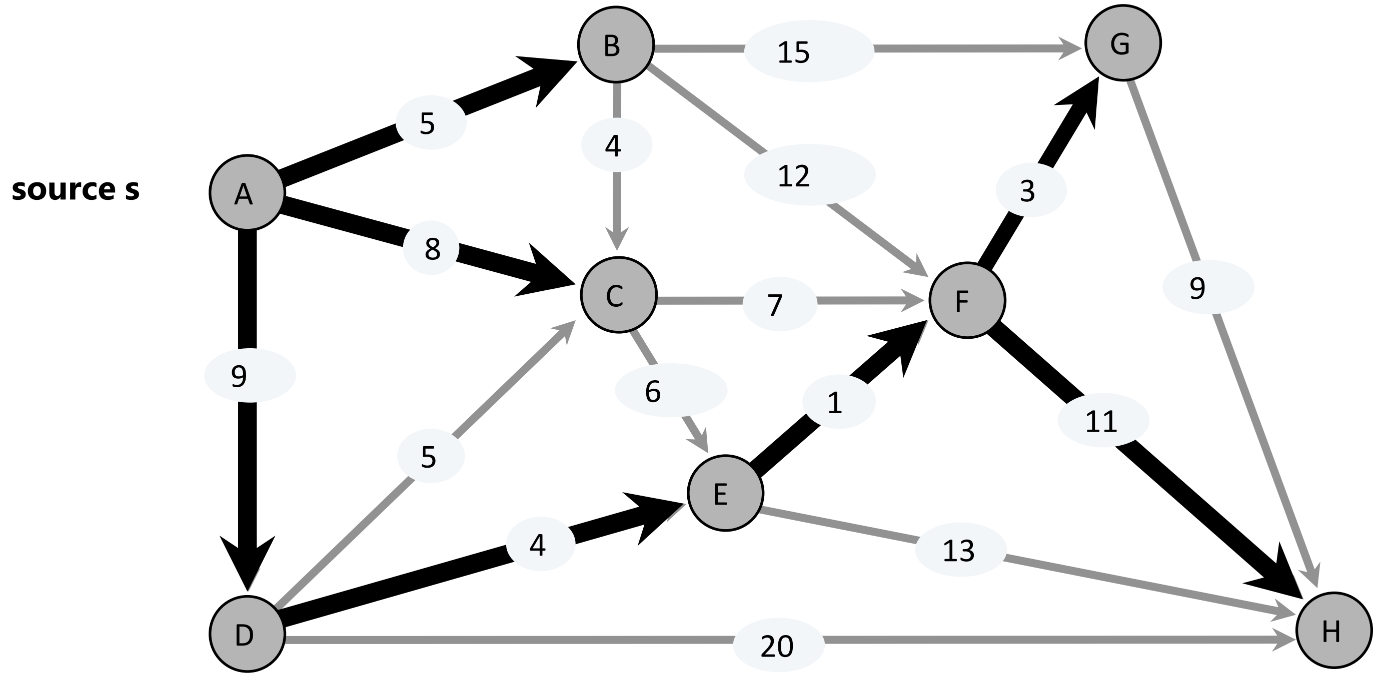
- **Problem:** Given a digraph $\boldsymbol{G = (V, E, w)}$, source $\boldsymbol{s} \in V$, and destination $\boldsymbol{d} \in V$, find a shortest directed path from $\boldsymbol{s}$ to $\boldsymbol{d}$.



length of path = 9 + 4 + 1 + 11 = 25

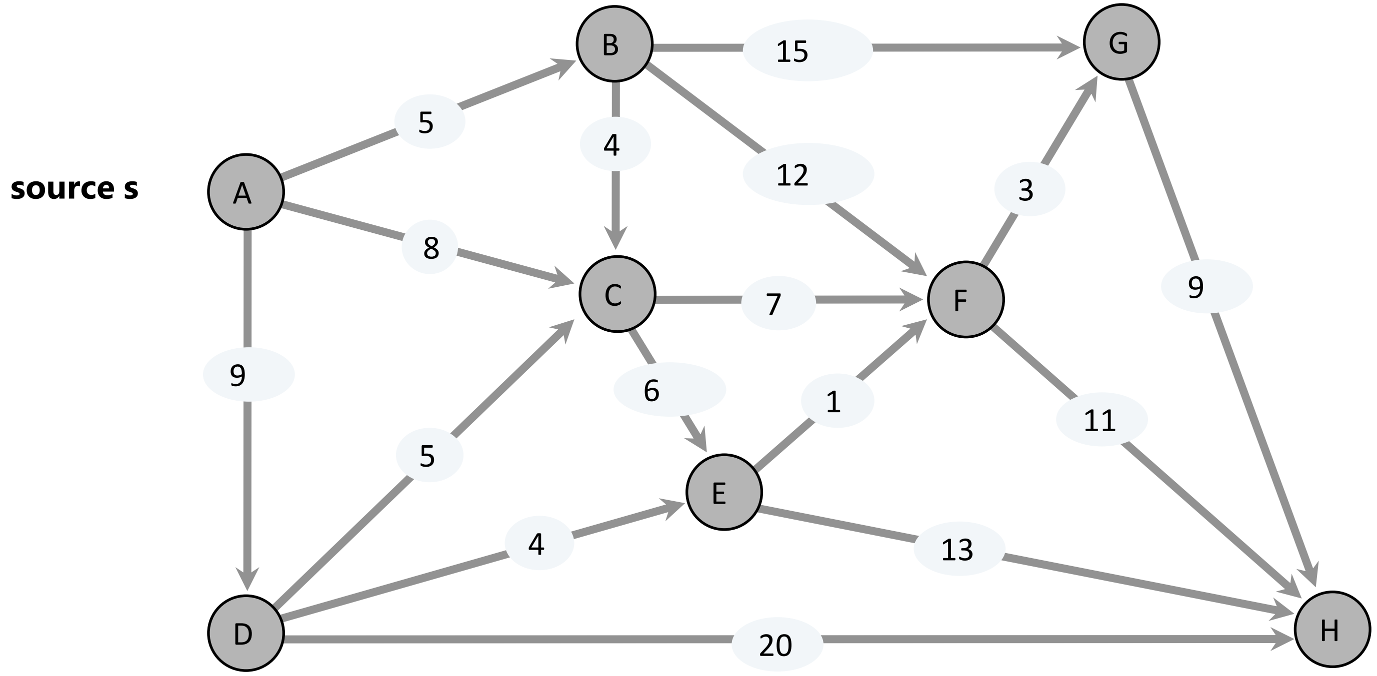
Single-source shortest paths problem

- **Problem**: Given a digraph $G = (V, E, w)$, source $s \in V$, find a shortest directed path from s to every node.



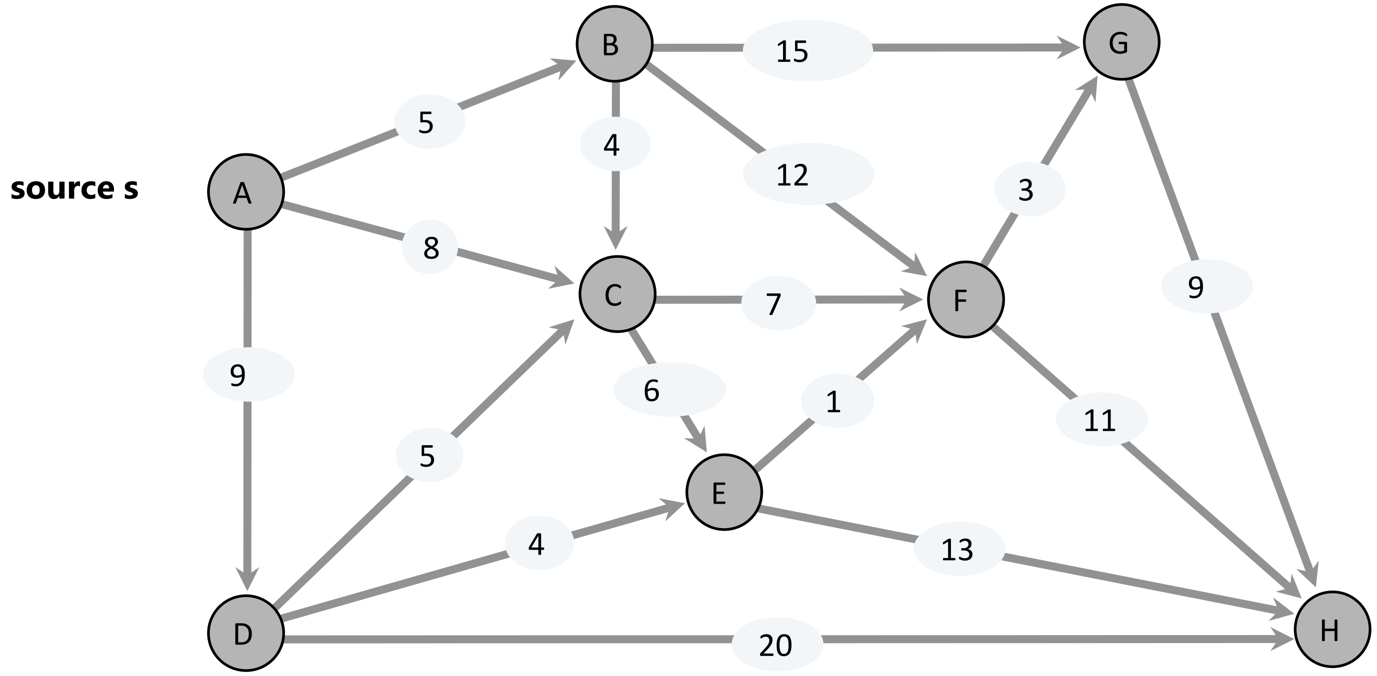
shortest-paths tree

Single-source shortest paths problem



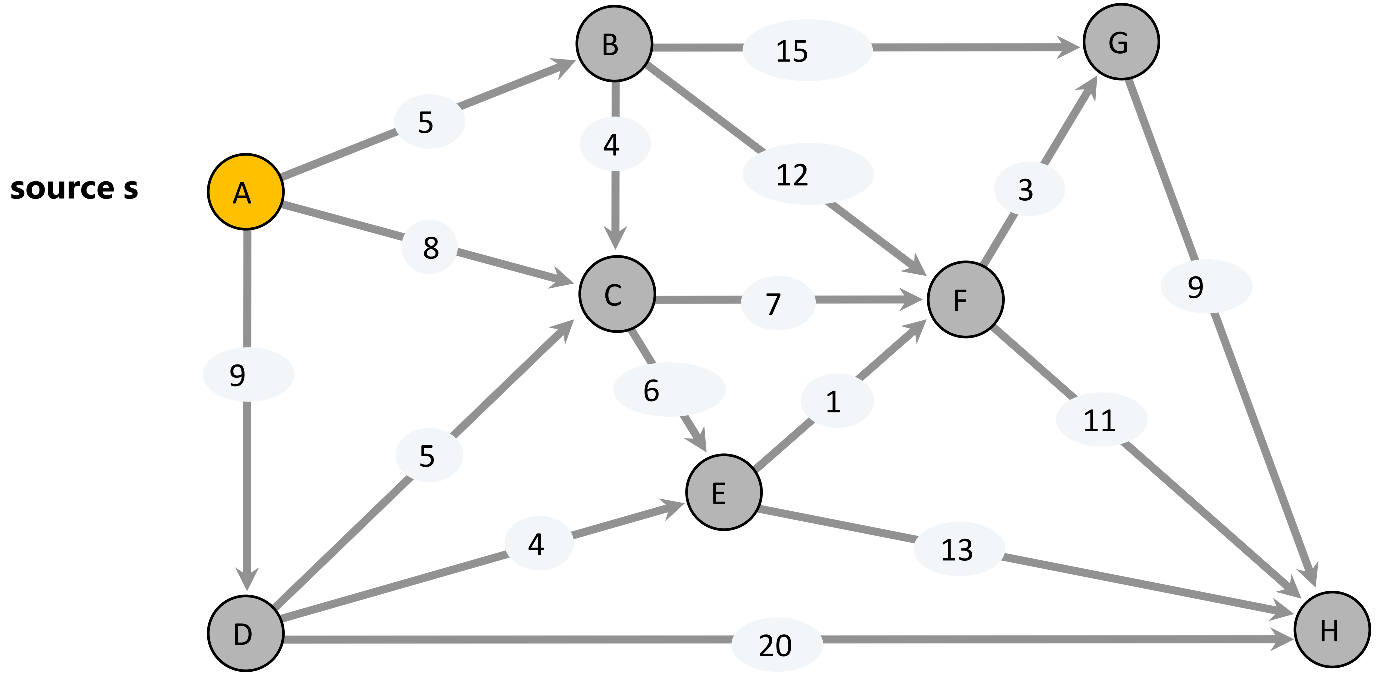
Single-source shortest paths problem

Why can't we use BFS?



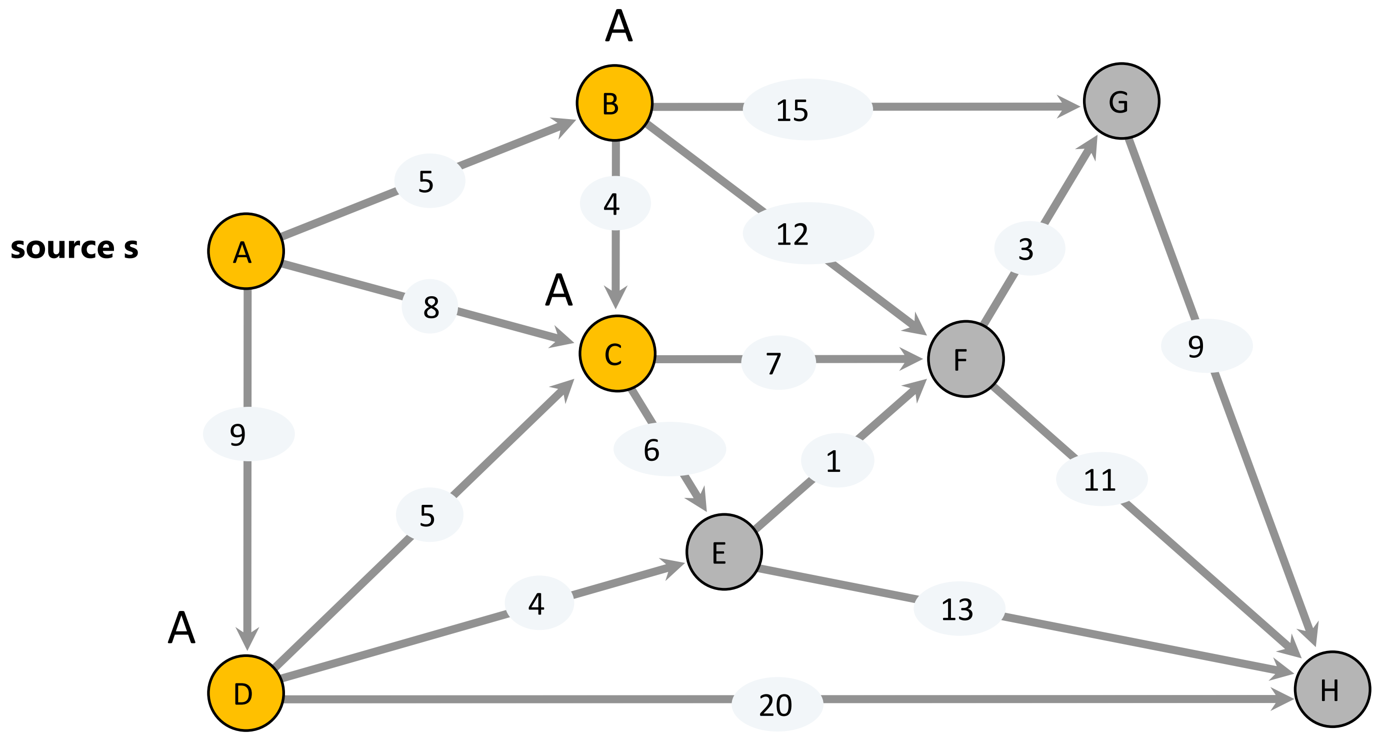
Single-source shortest paths problem

Why can't we use BFS?



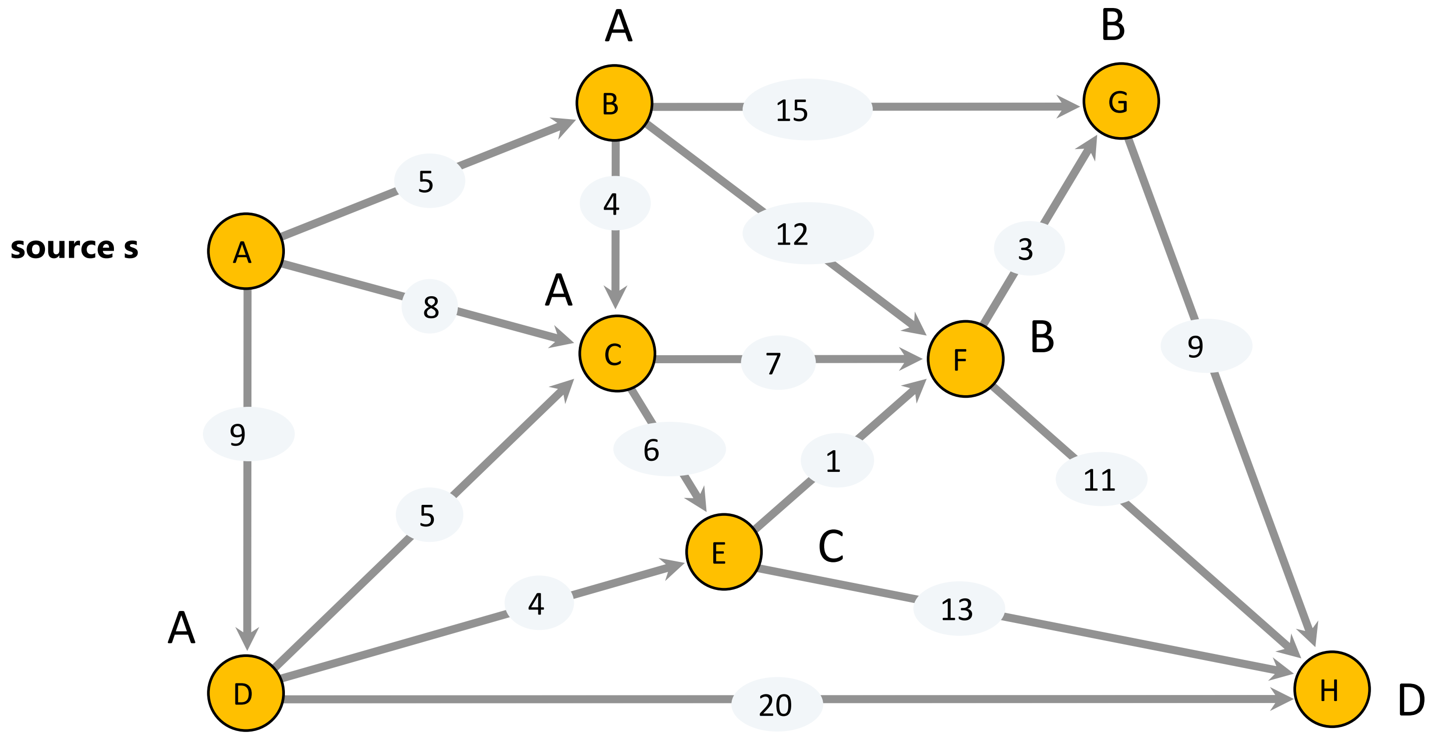
Single-source shortest paths problem

Why can't we use BFS?



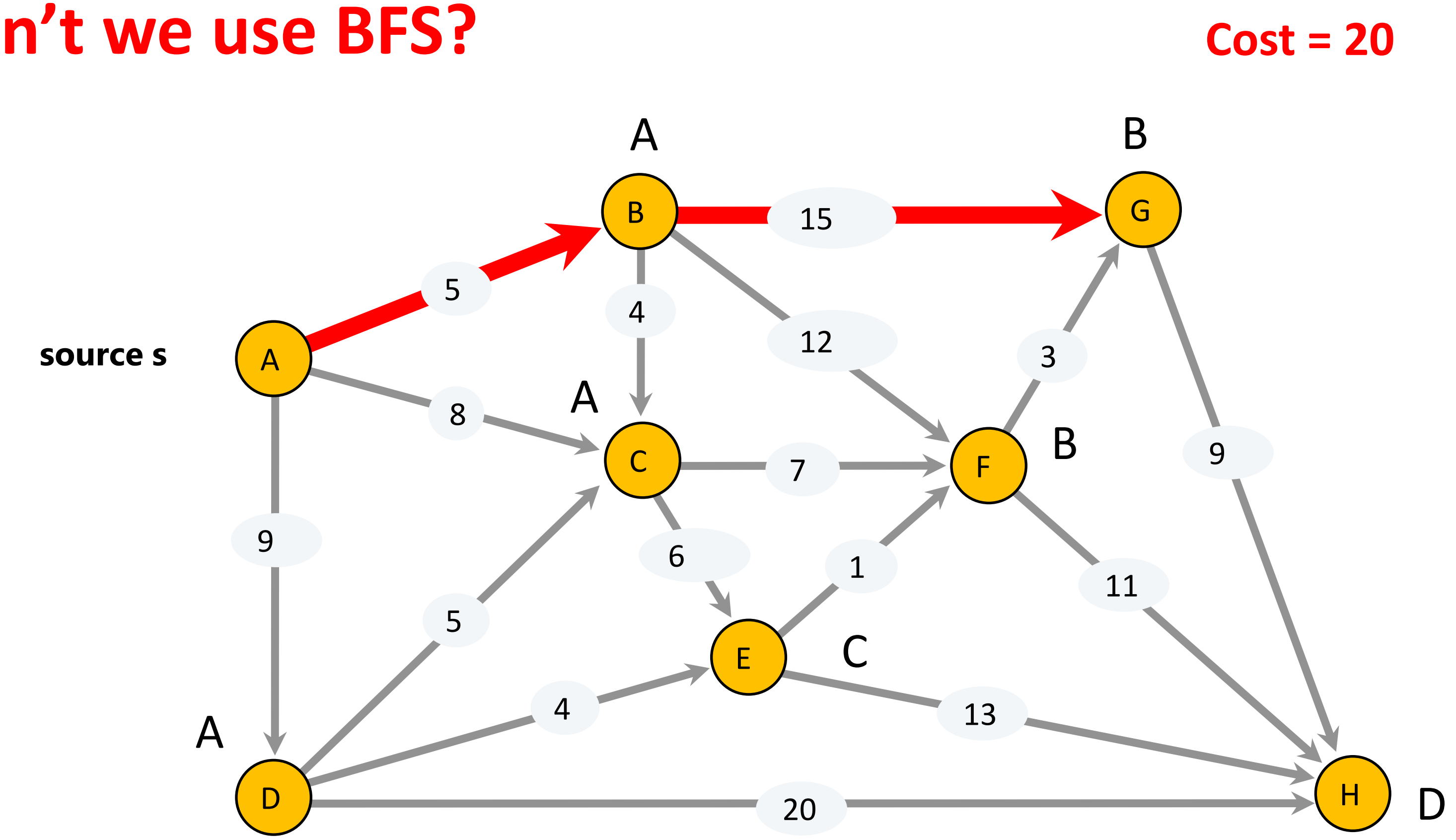
Single-source shortest paths problem

Why can't we use BFS?



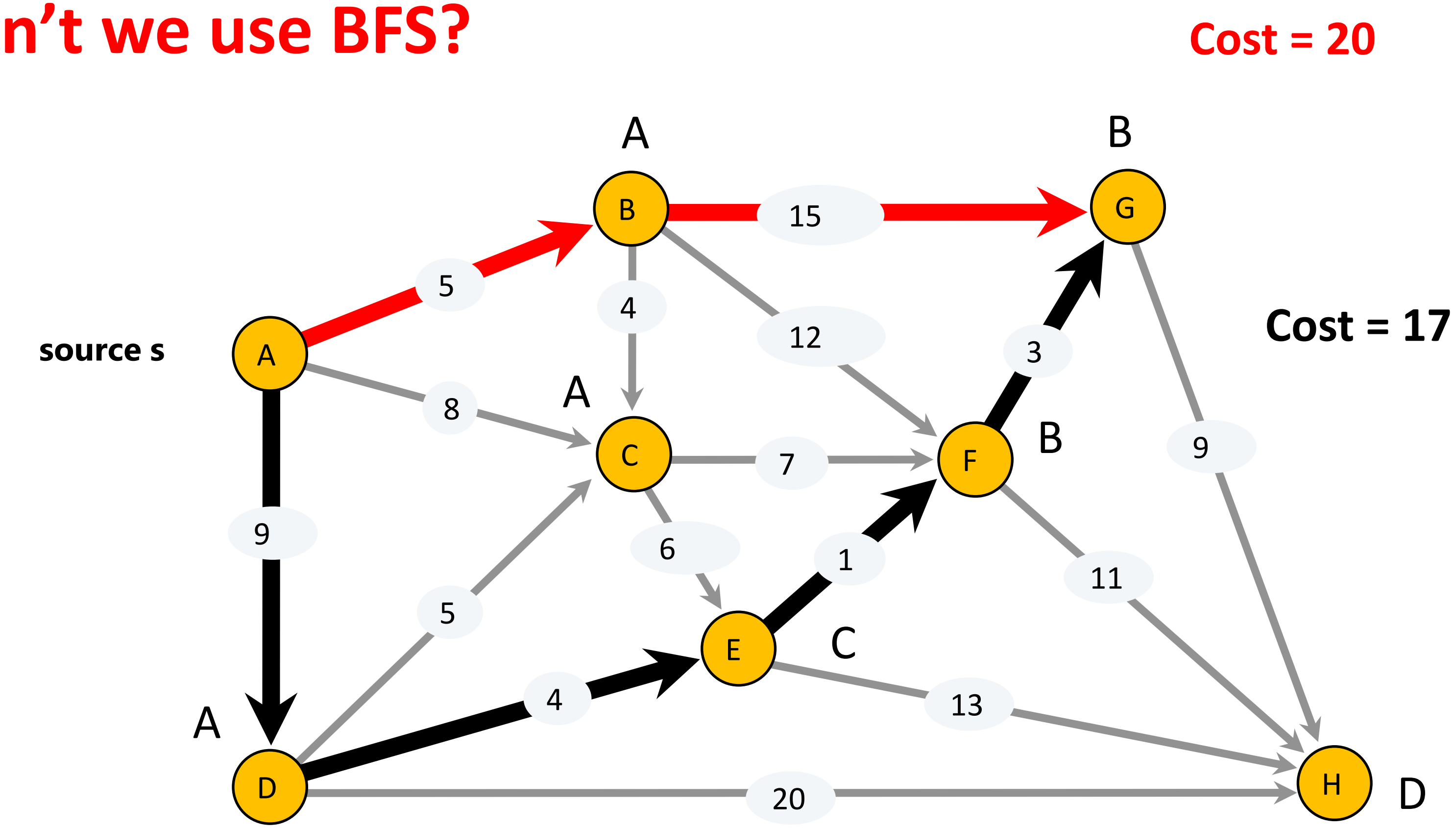
Single-source shortest paths problem

Why can't we use BFS?

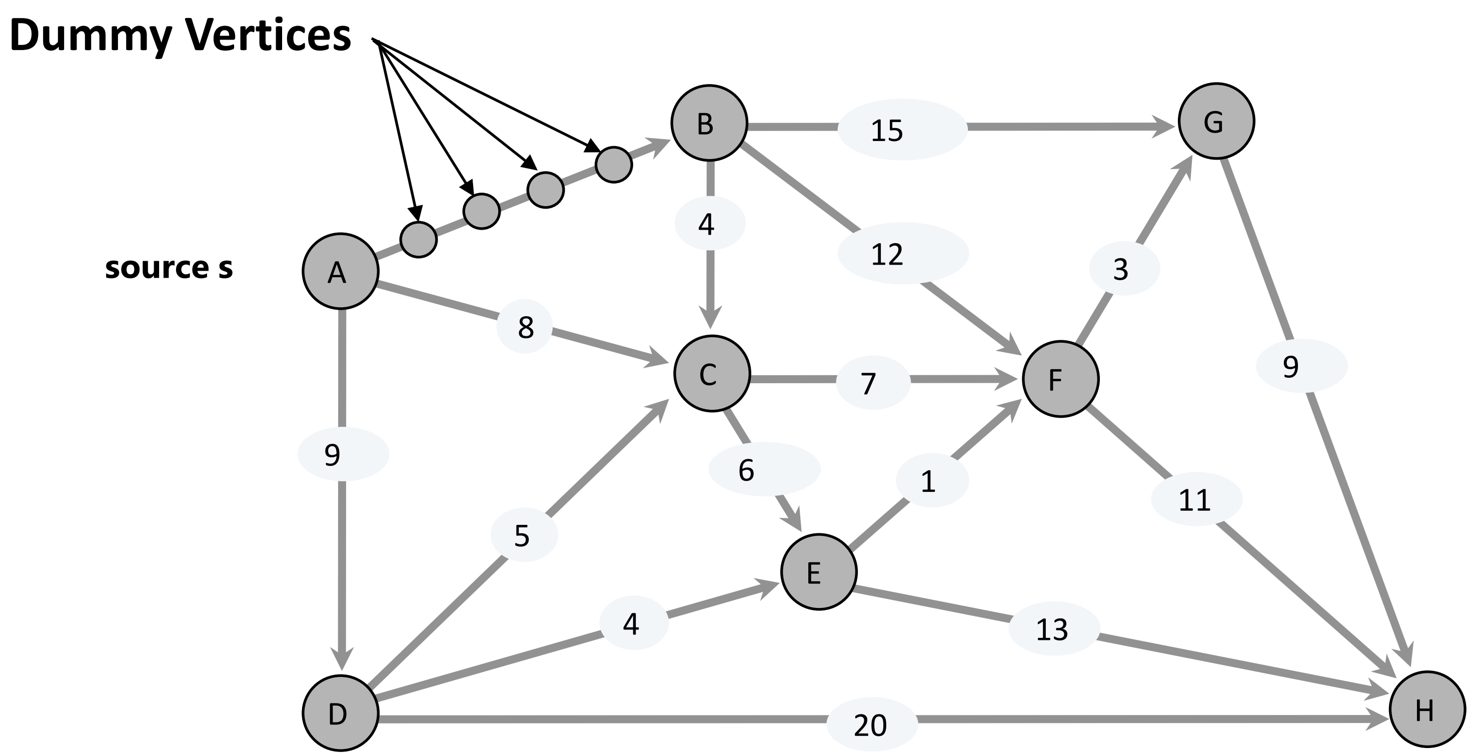


Single-source shortest paths problem

Why can't we use BFS?



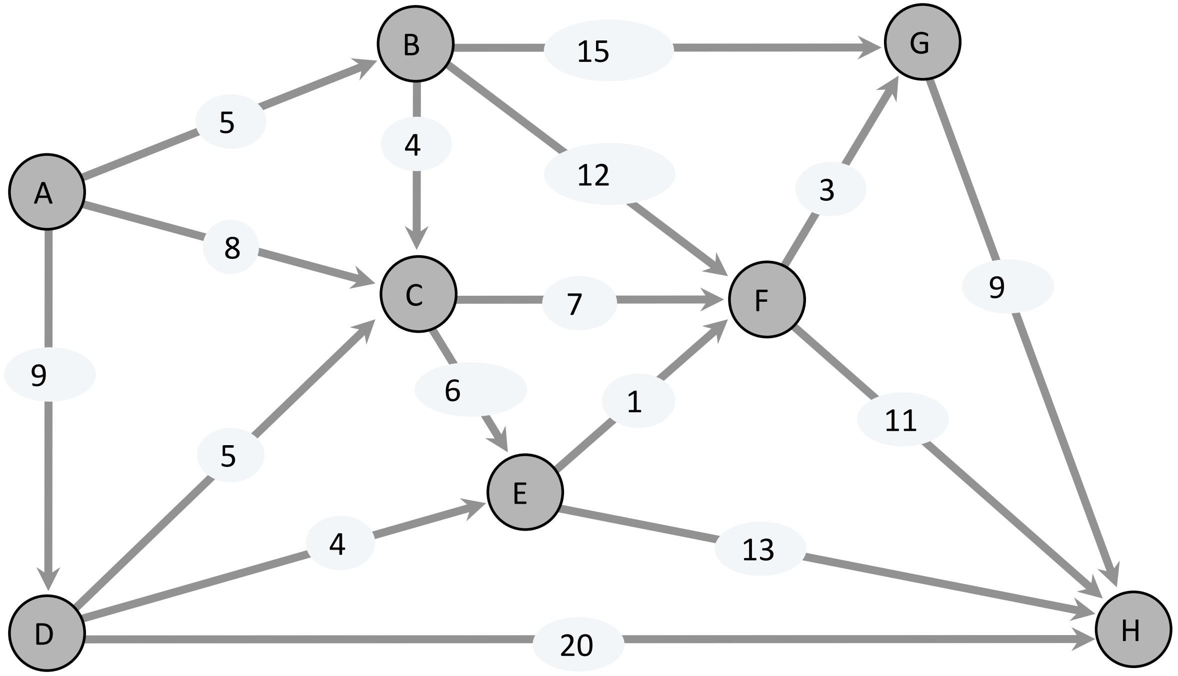
Single-source shortest paths problem



Edge splitting: Transforming a weighted graph into an unweighted one

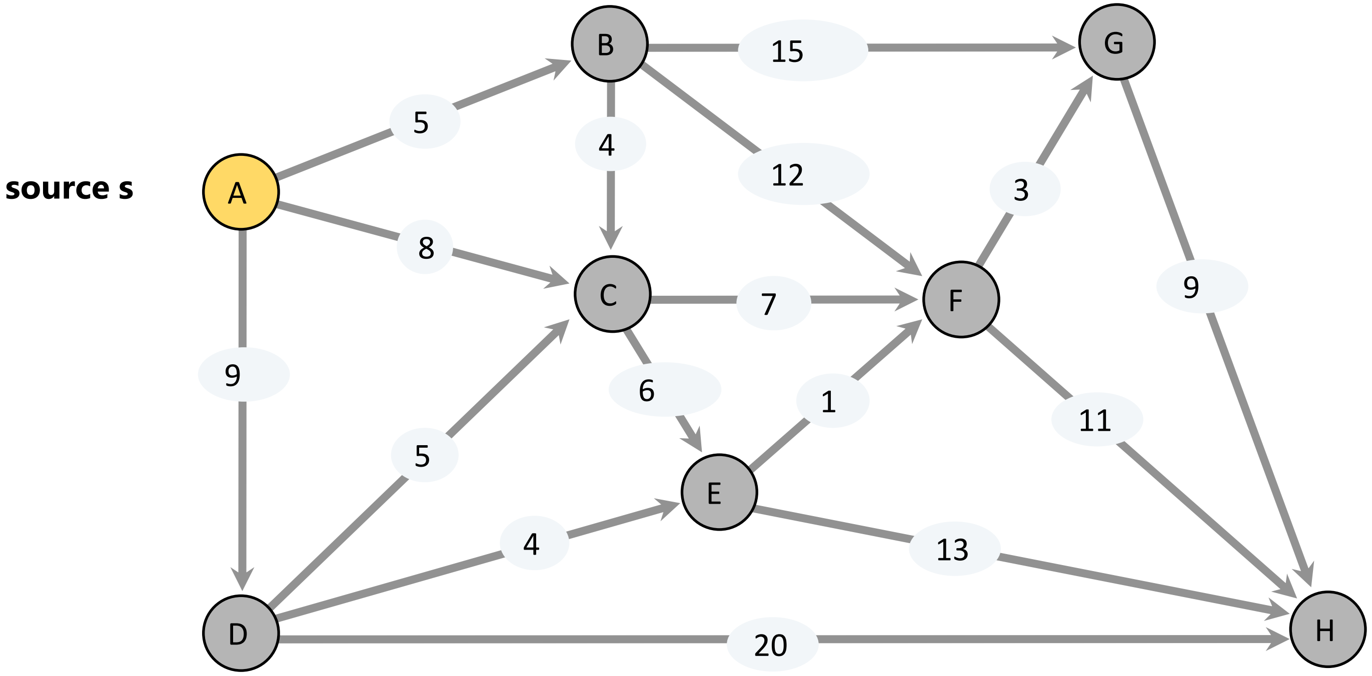
Single-source shortest paths problem

source s



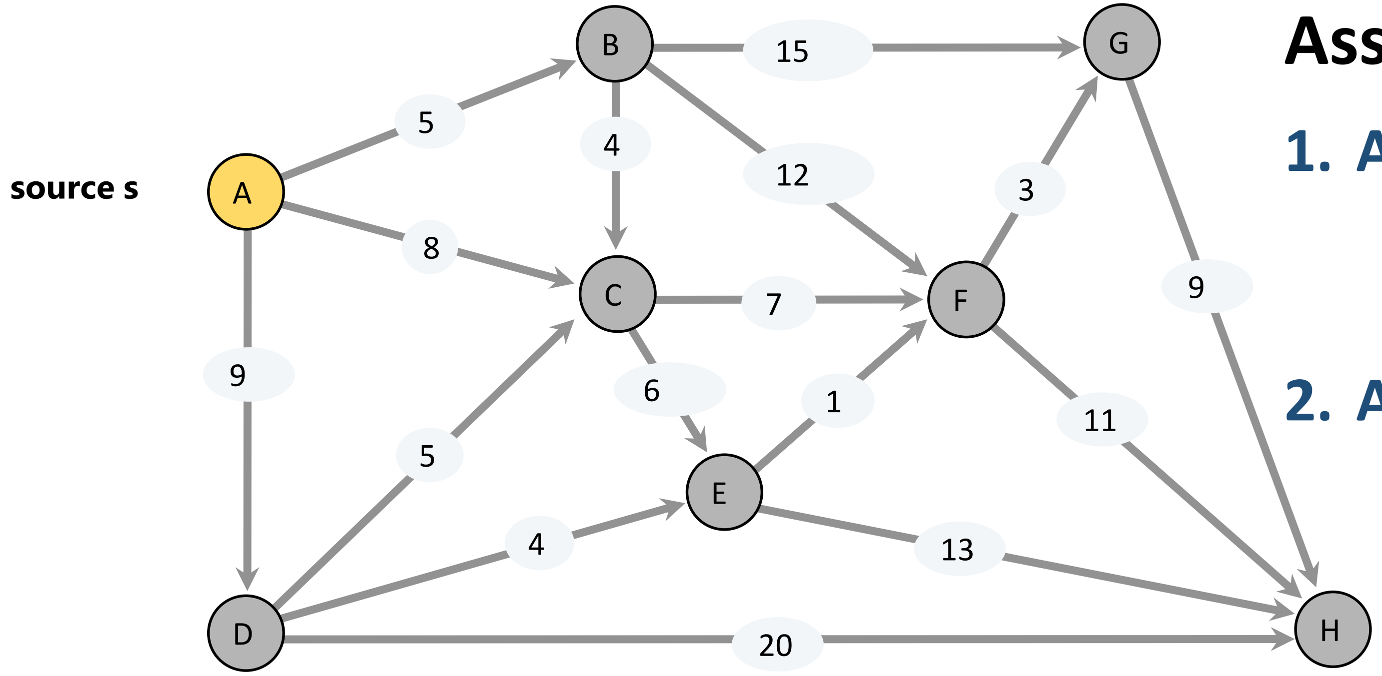
Dijkstra's algorithm for shortest path

Single-source shortest paths problem



Dijkstra's algorithm solves this problem for both **directed** and **undirected** graphs

Single-source shortest paths problem

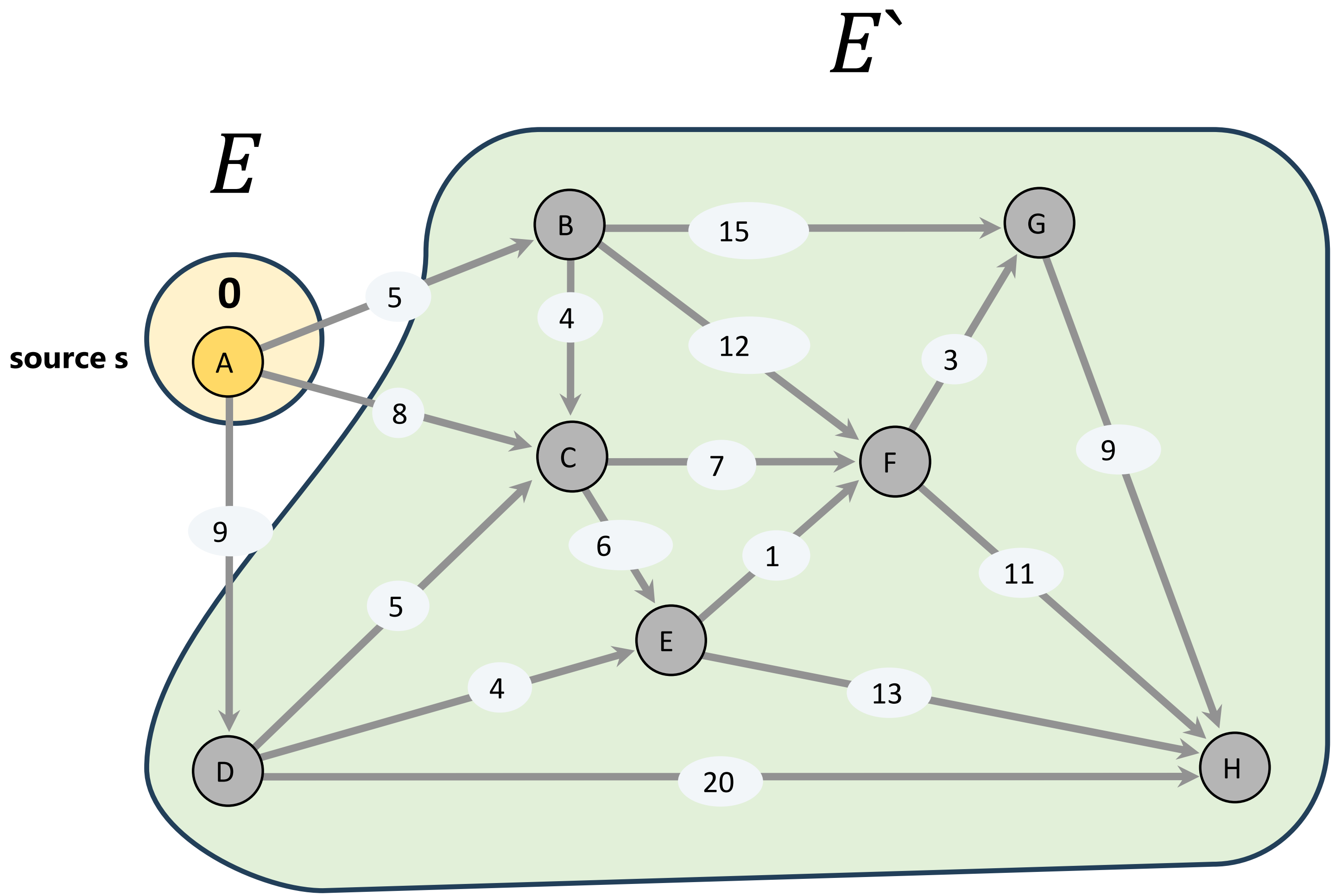


Assumption:

1. **All vertices are reachable from s**
   - Otherwise, there is no shortest path
   - Easy to get R(s) in preprocessing (e.g., BFS or DFS)

2. **All edge weights are non-negative**

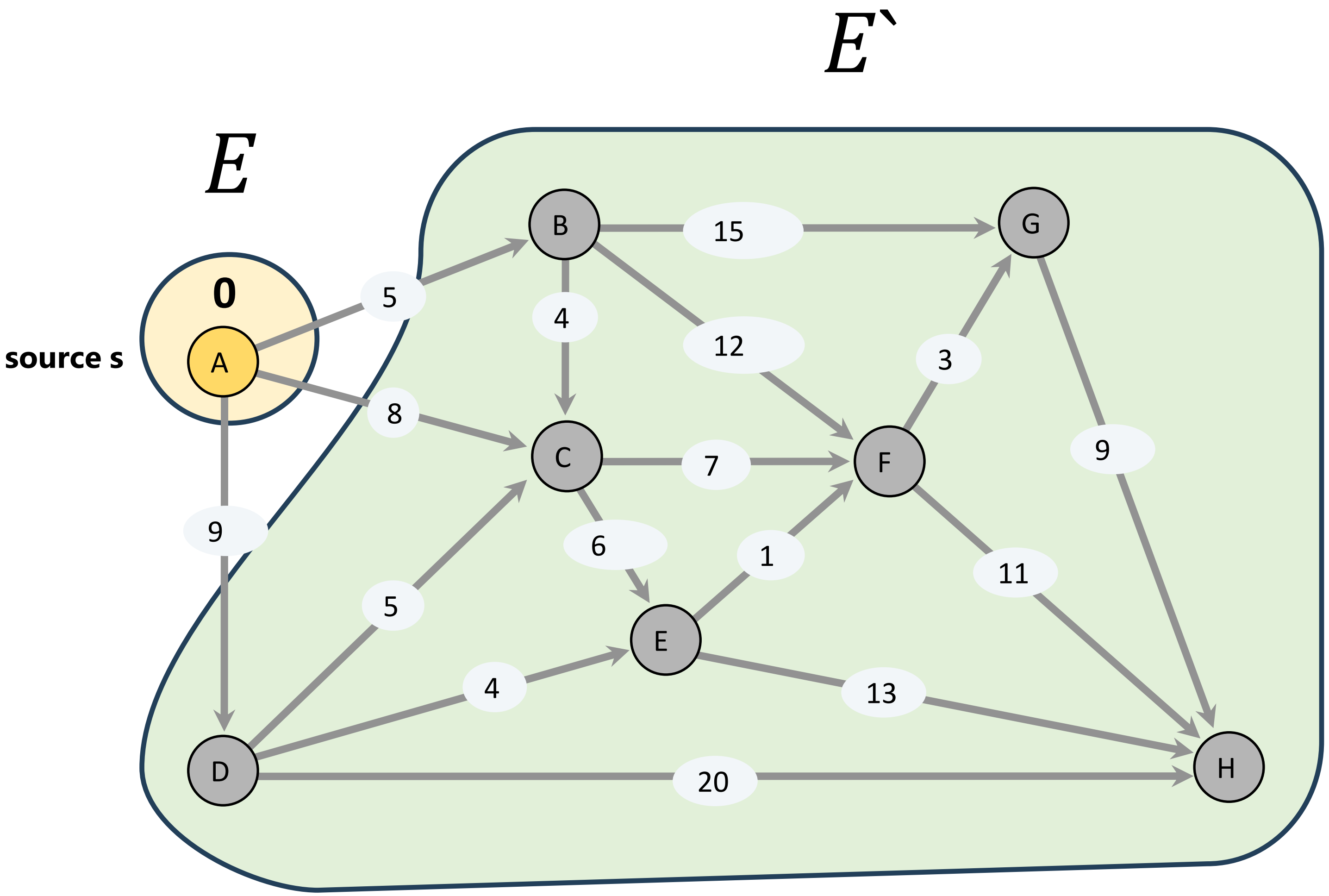
Single-source shortest paths problem



	distance[ ]	prev[ ]
A	**0**	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	
G	∞	
H	∞	

Initially ***E*** = {A} and iteratively add one vertex to ***E`***

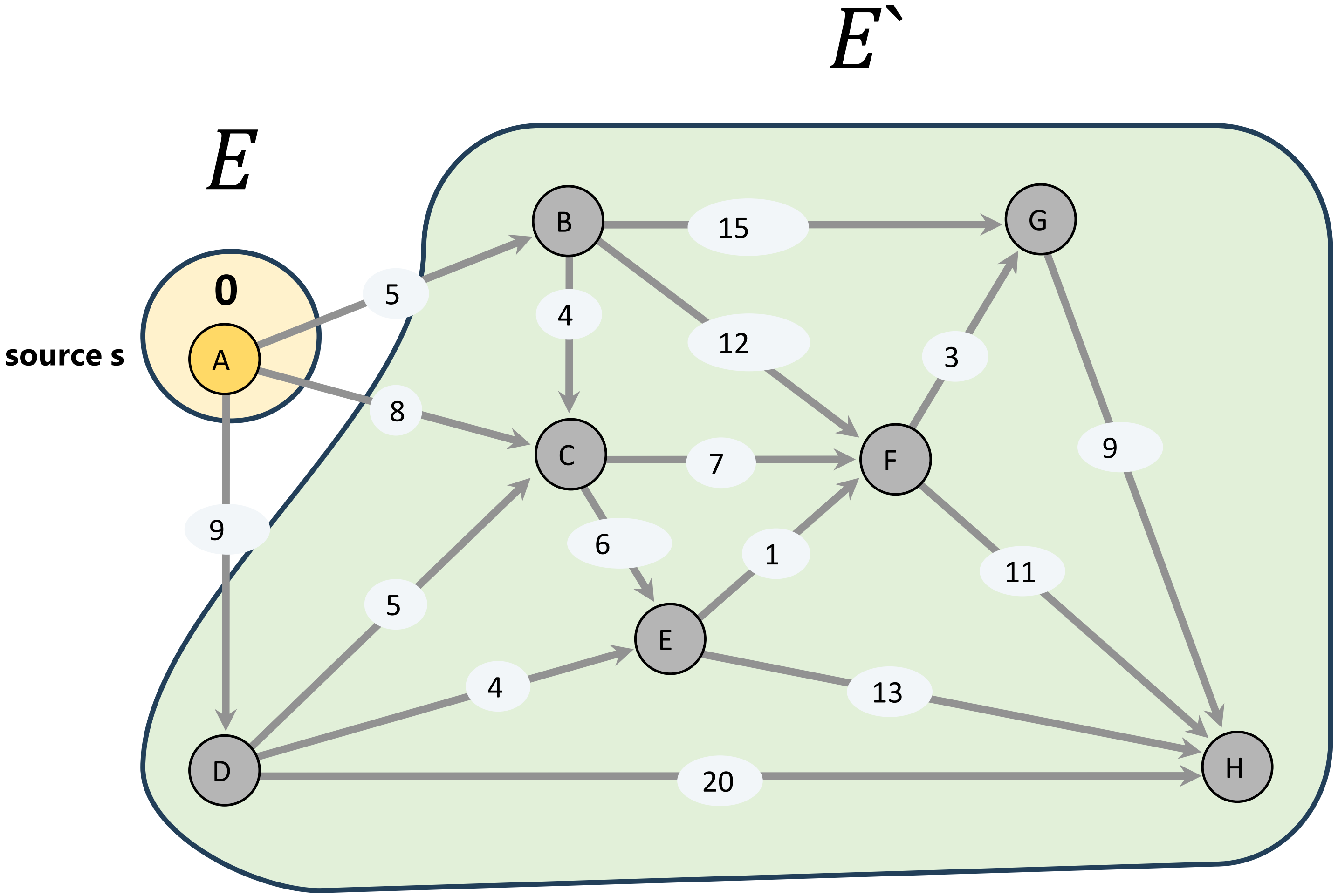
Single-source shortest paths problem



	distance[ ]	prev[ ]
A	**0**	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	
G	∞	
H	∞	

Which vertex from **E`** to add to **E**?

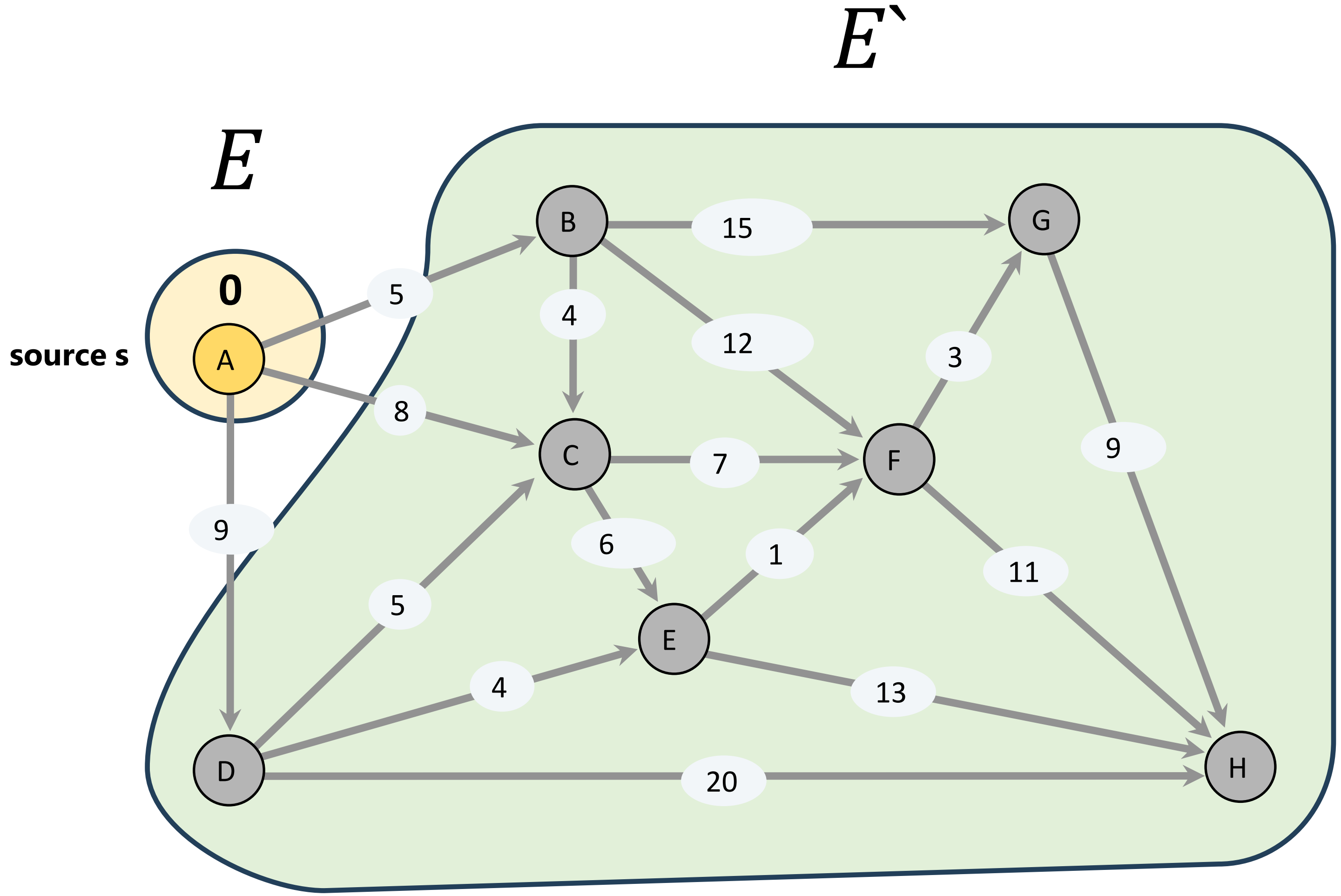
Single-source shortest paths problem



	distance[ ]	prev[ ]
A	**0**	
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	
G	∞	
H	∞	

The vertex $\boldsymbol{v \in E`}$ that is closest to **A**

Single-source shortest paths problem

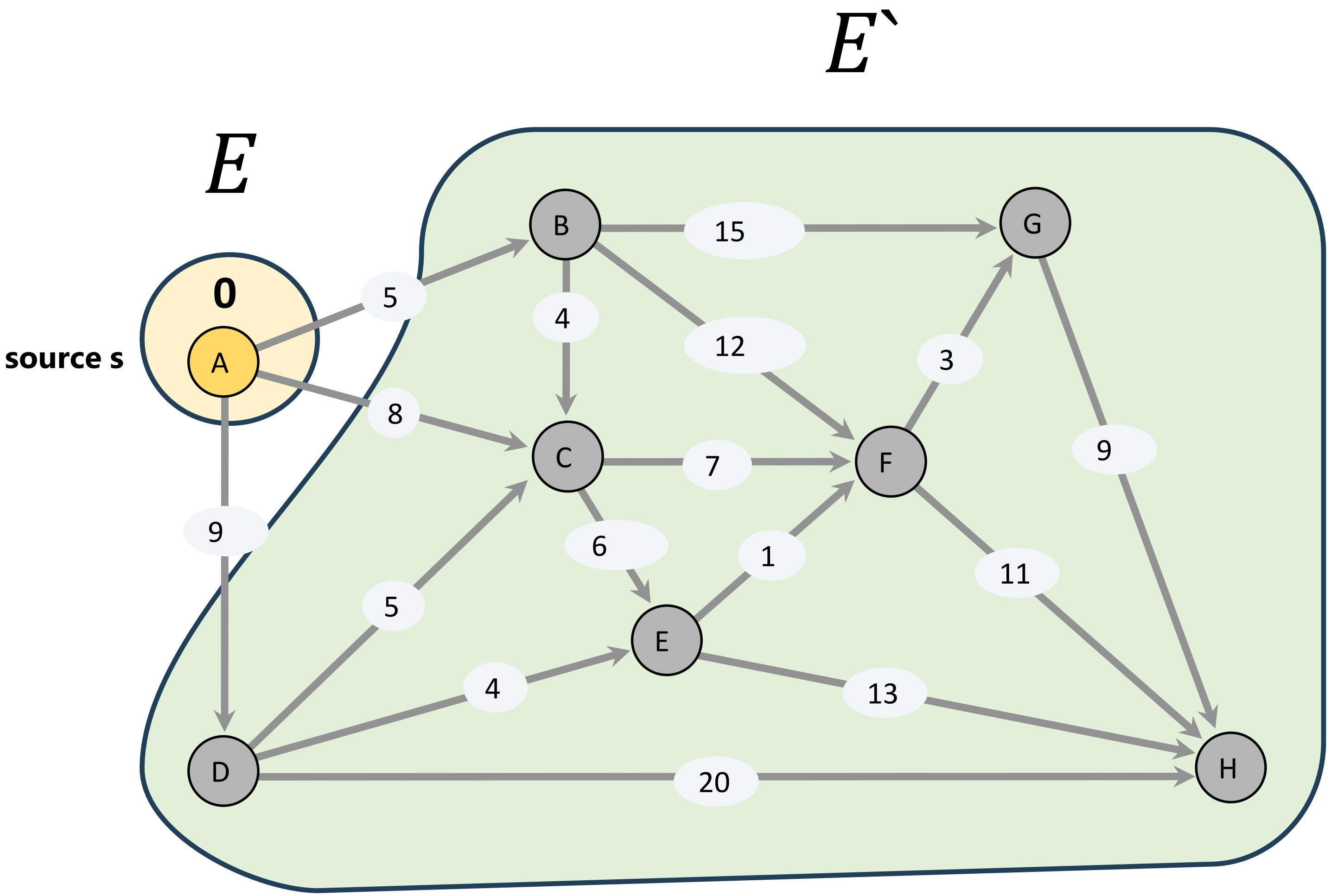


	distance[ ]		prev[ ]
A	**0**	**A**	
B	∞	B	
C	∞	C	
D	∞	D	
E	∞	E	
F	∞	F	
G	∞	G	
H	∞	H	

$$\pi(v) = \min_{e\,=\,(u,v)\,:\,u\in S} \boxed{d[u] + \ell_e}$$

the length of a shortest path from s to some node u in explored part E, followed by a single edge $e = (u, v)$

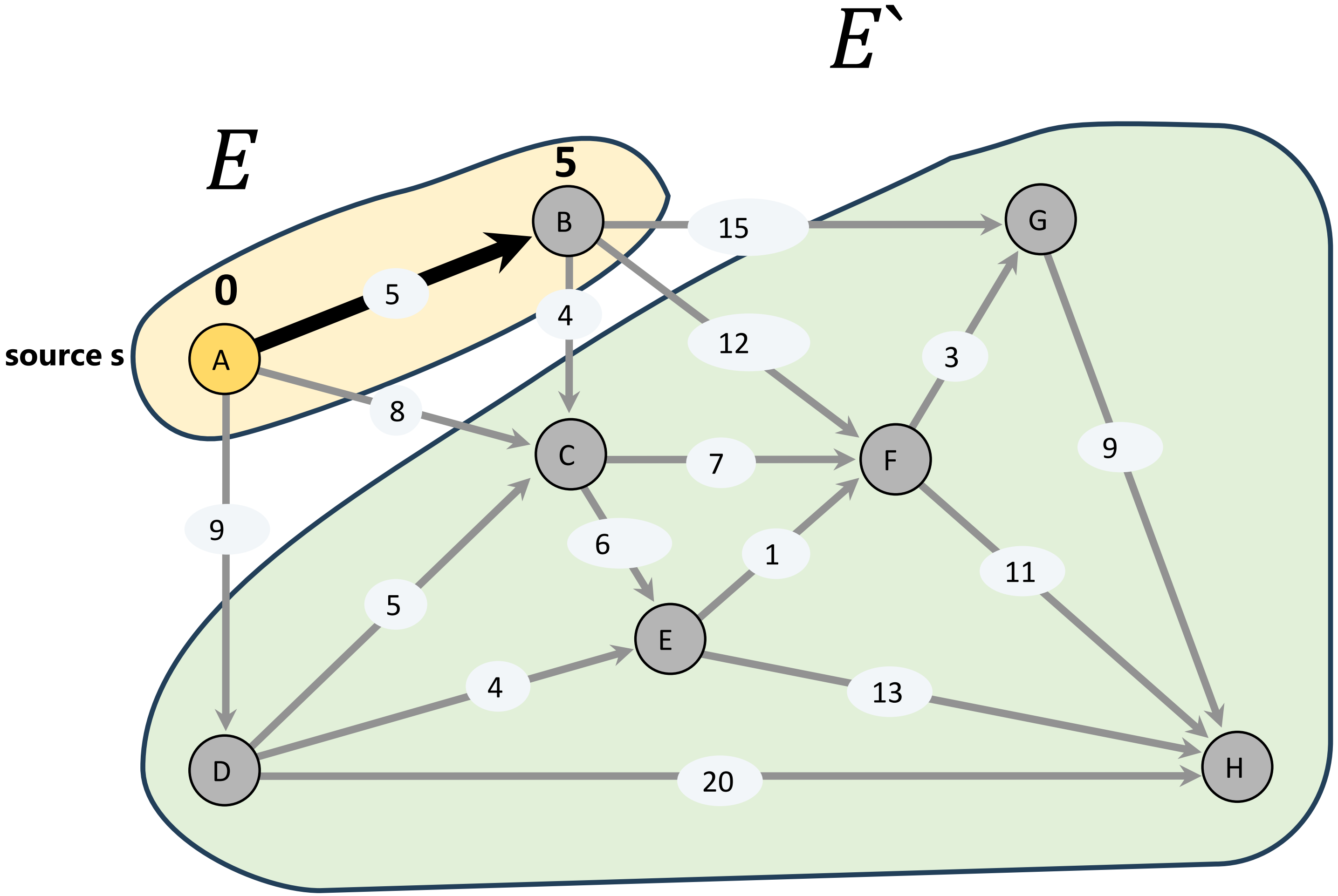
Single-source shortest paths problem



	distance[ ]		prev[ ]
A	**0**	**A**	
B	∞	B	
C	∞	C	
D	∞	D	
E	∞	E	
F	∞	F	
G	∞	G	
H	∞	H	

add $\boldsymbol{v}$ to $\boldsymbol{E}$, and set $d[v] \leftarrow \pi(v)$.

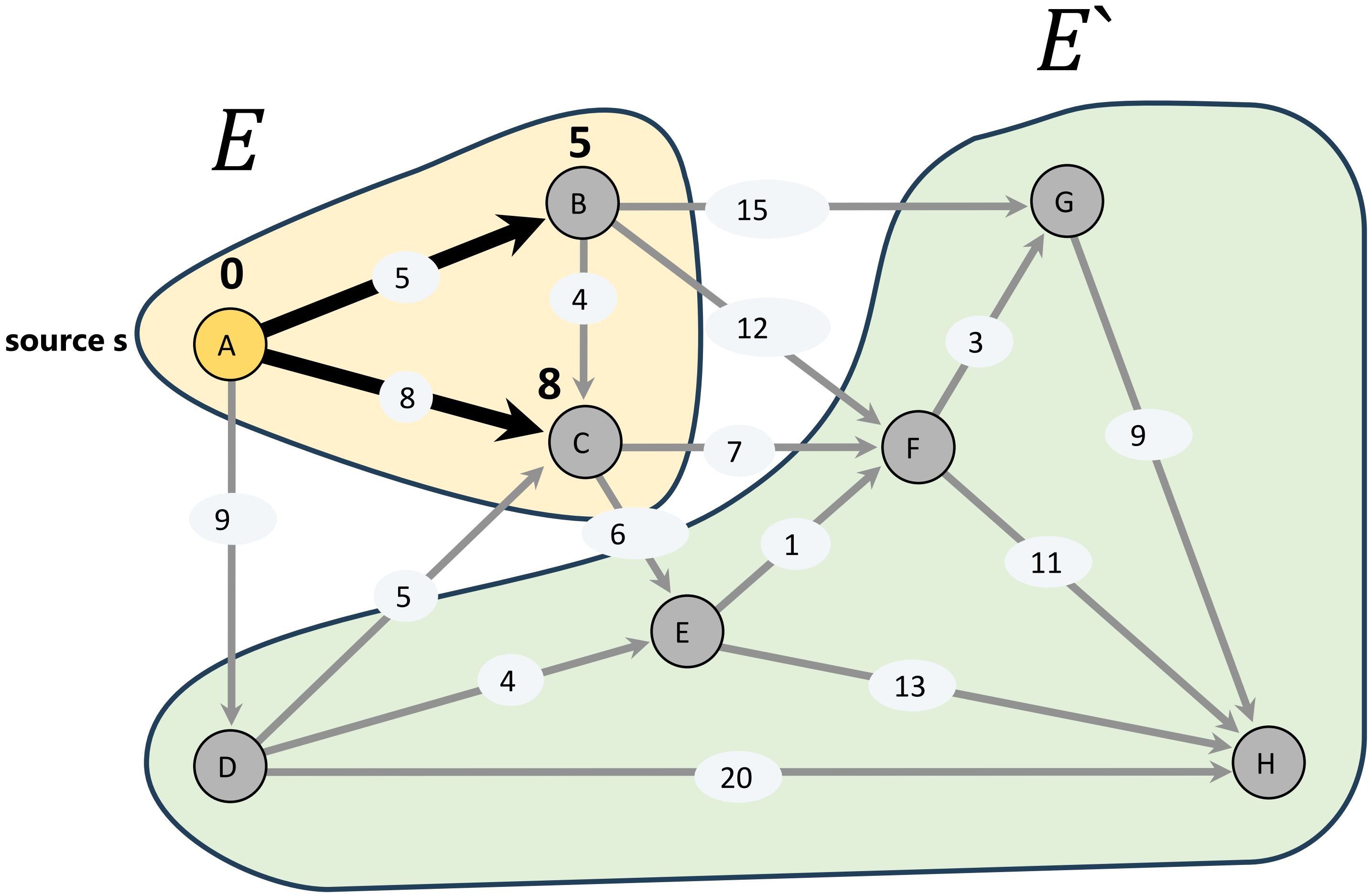
Single-source shortest paths problem



	distance[ ]	prev[ ]
A	**0**	
B	5	A
C	∞	
D	∞	
E	∞	
F	∞	
G	∞	
H	∞	

$$\pi(v) \ = \min_{e=(u,v)\,:\,u\in S} d[u] + \ell_e$$

Single-source shortest paths problem

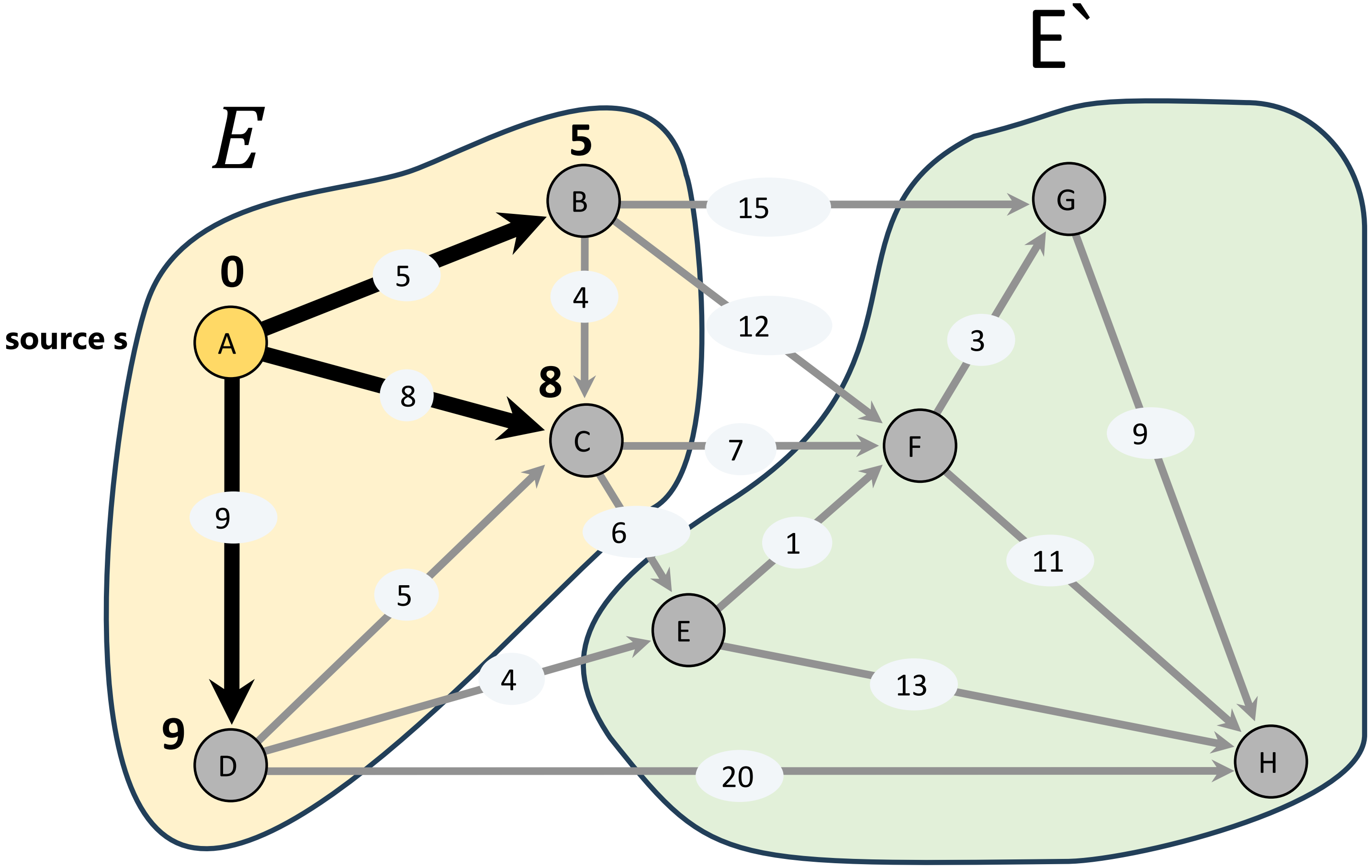


distance[ ]	
A	**0**
B	5
C	8
D	∞
E	∞
F	∞
G	∞
H	∞

prev[ ]	
A	
B	A
C	A
D	
E	
F	
G	
H	

$$\pi(v) \;=\; \min_{e\,=\,(u,v)\;:\;u\in S} \; d[u] + \ell_e$$

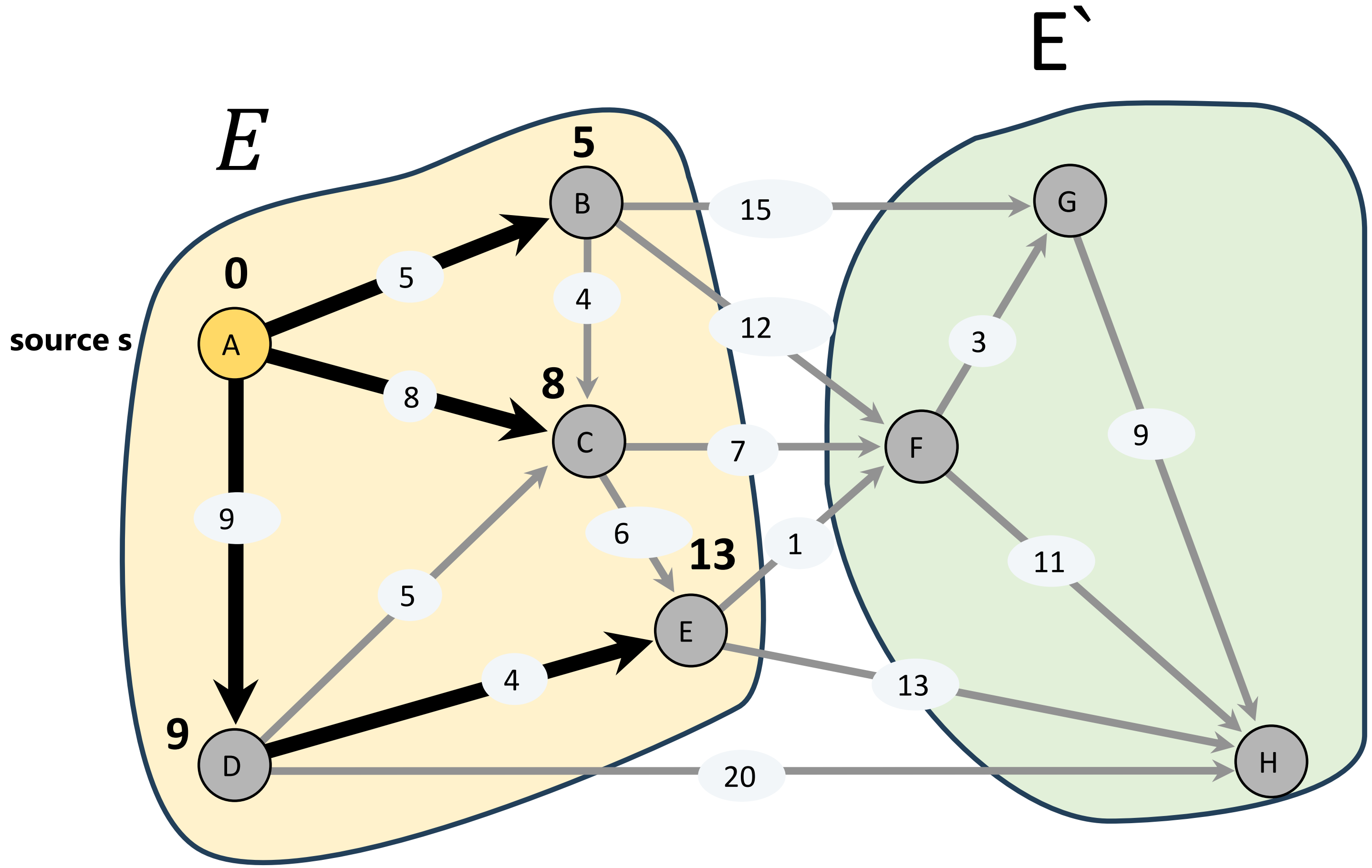
Single-source shortest paths problem



	distance[ ]		prev[ ]
A	**0**	**A**	
B	5	B	A
C	8	C	A
D	9	D	A
E	∞	E	
F	∞	F	
G	∞	G	
H	∞	H	

$$\pi(v) \;=\; \min_{e\,=\,(u,v)\;:\;u\in S} \; d[u] + \ell_e$$

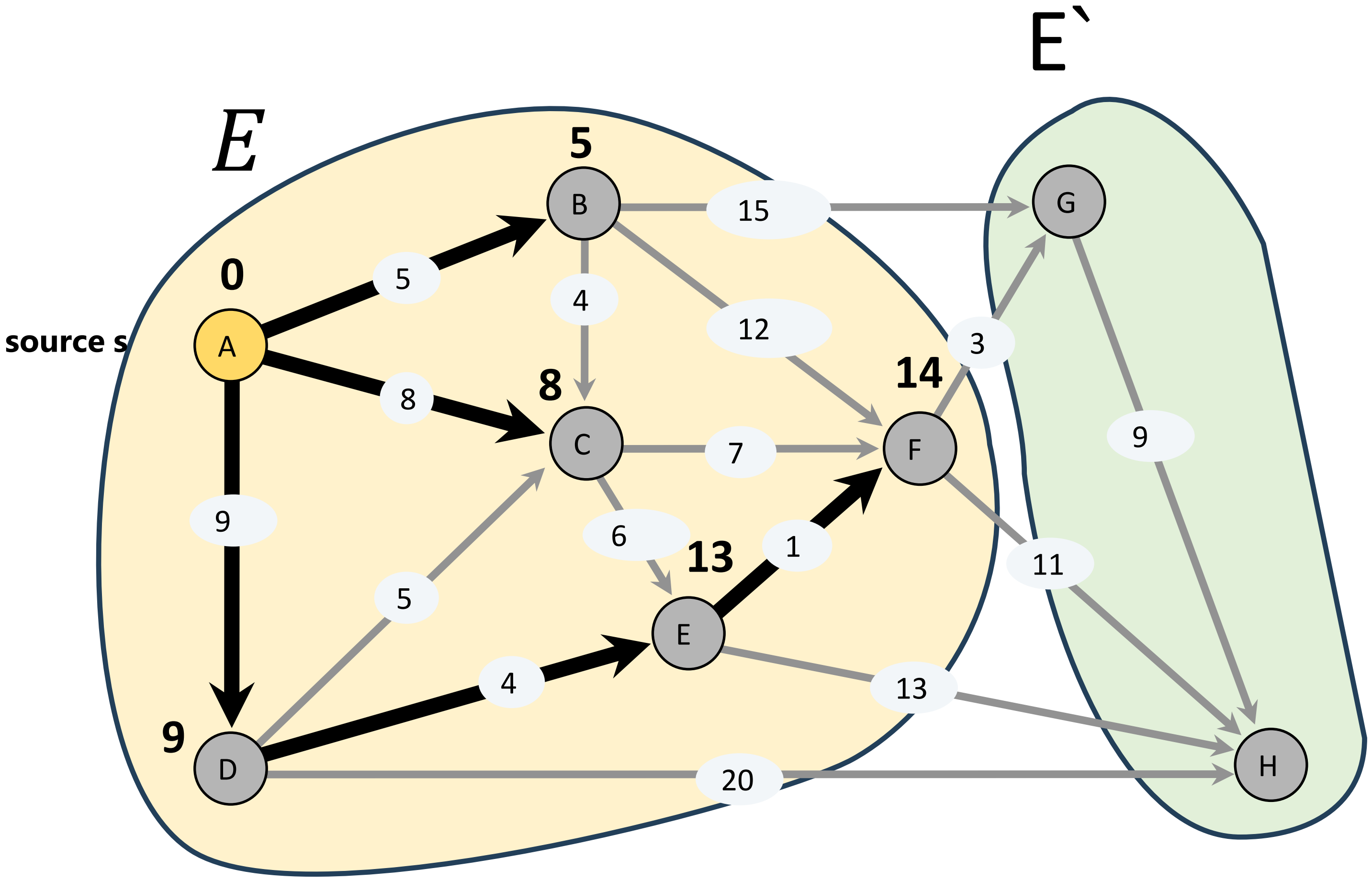
Single-source shortest paths problem



	distance[ ]	prev[ ]
A	**0**	
B	5	A
C	8	A
D	9	A
E	13	D
F	∞	
G	∞	
H	∞	

$$\pi(v) \;=\; \min_{e\,=\,(u,v)\,:\,u\in S} \; d[u] + \ell_e$$

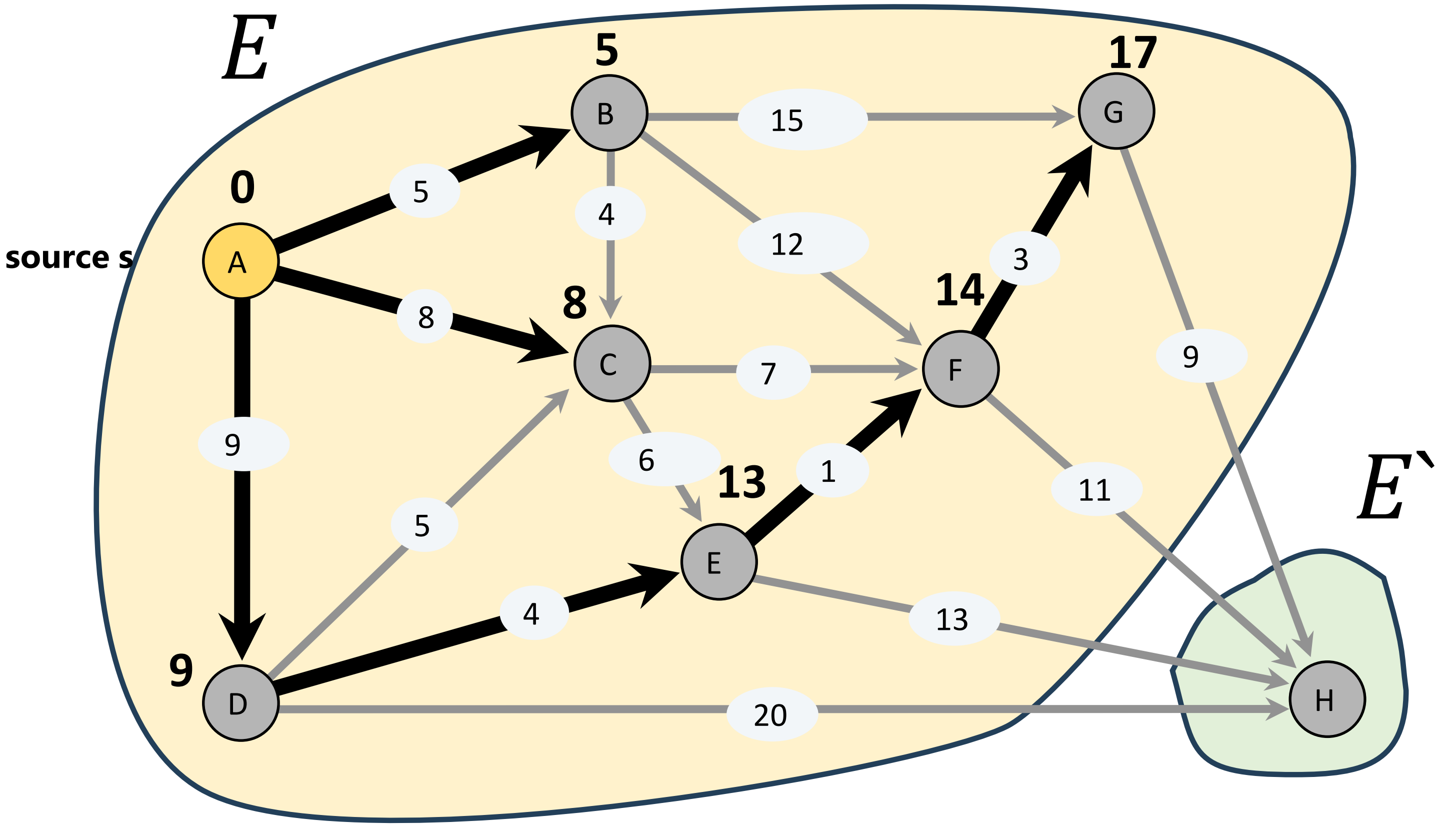
Single-source shortest paths problem



	distance[ ]		prev[ ]
A	**0**	**A**	
B	5	B	A
C	8	C	A
D	9	D	A
E	13	E	D
F	14	F	E
G	∞	G	
H	∞	H	

$$\pi(v) \ = \min_{e=(u,v)\,:\,u\in S} d[u] + \ell_e$$

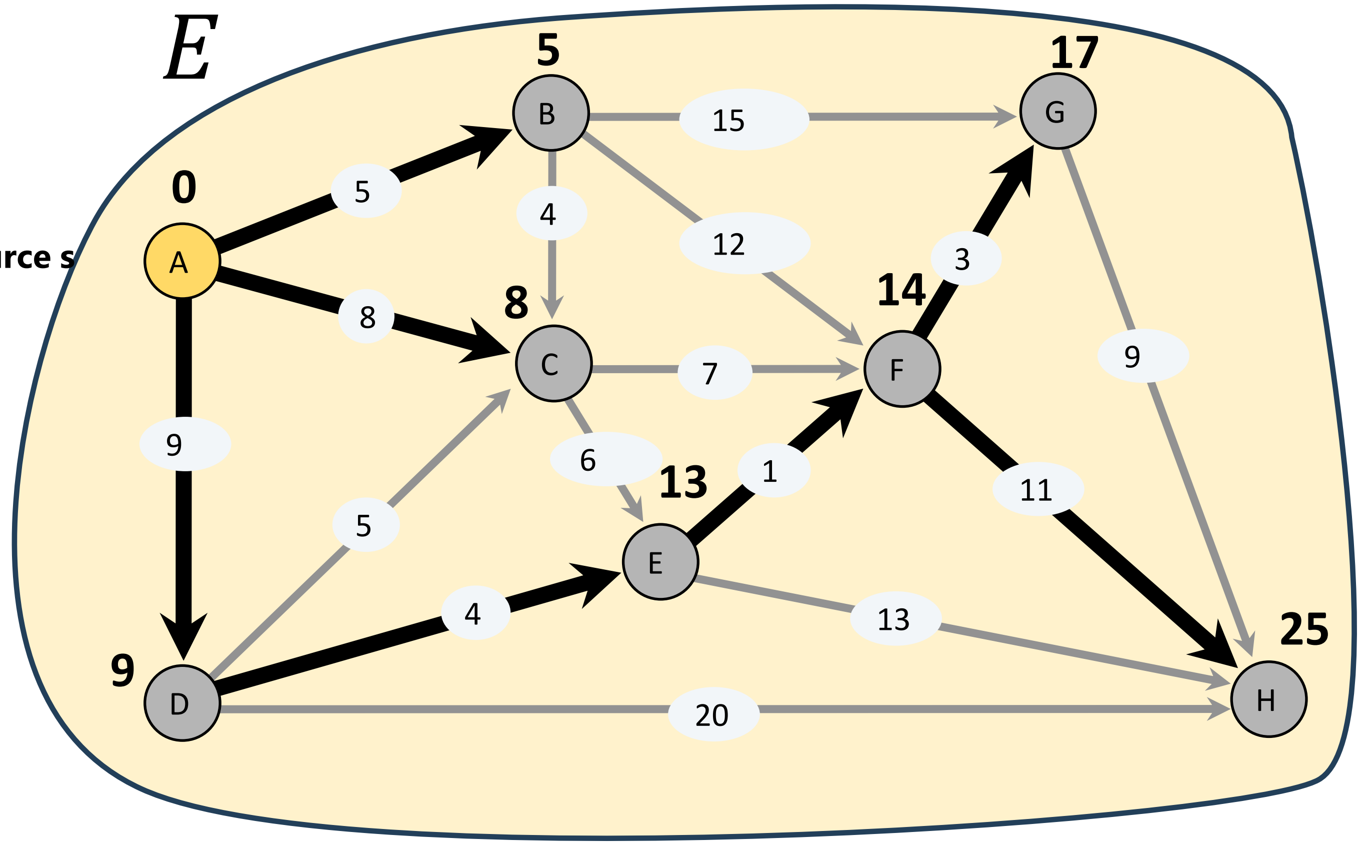
Single-source shortest paths problem



	distance[ ]	prev[ ]
A	**0**	
B	5	A
C	8	A
D	9	A
E	13	D
F	14	E
G	17	F
H	∞	

$$\pi(v) \;=\; \min_{e\,=\,(u,v)\,:\,u\in S} d[u] + \ell_e$$

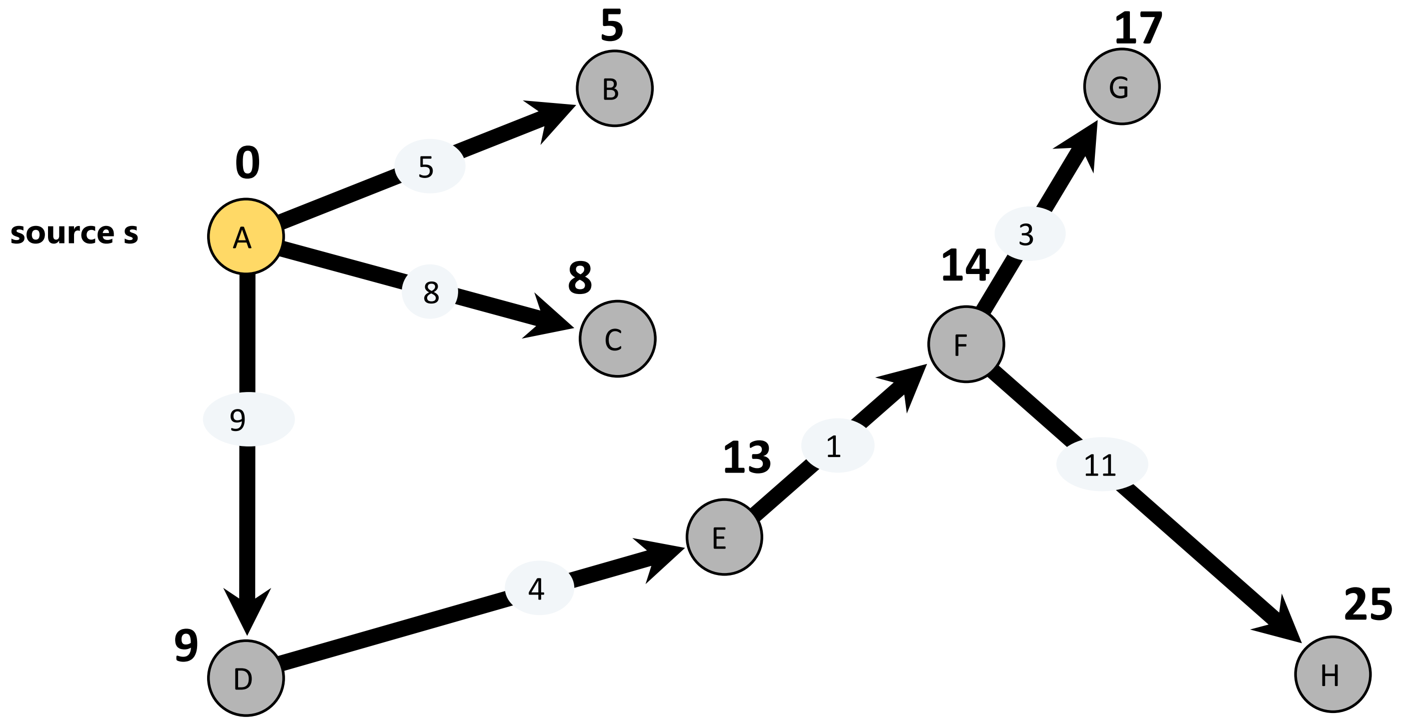
Single-source shortest paths problem



	distance[ ]	prev[ ]
A	**0**	
B	5	A
C	8	A
D	9	A
E	13	D
F	14	E
G	17	F
H	25	F

$$\pi(v) \; = \min_{e\,=\,(u,v)\;:\;u \in S} d[u] + \ell_e$$

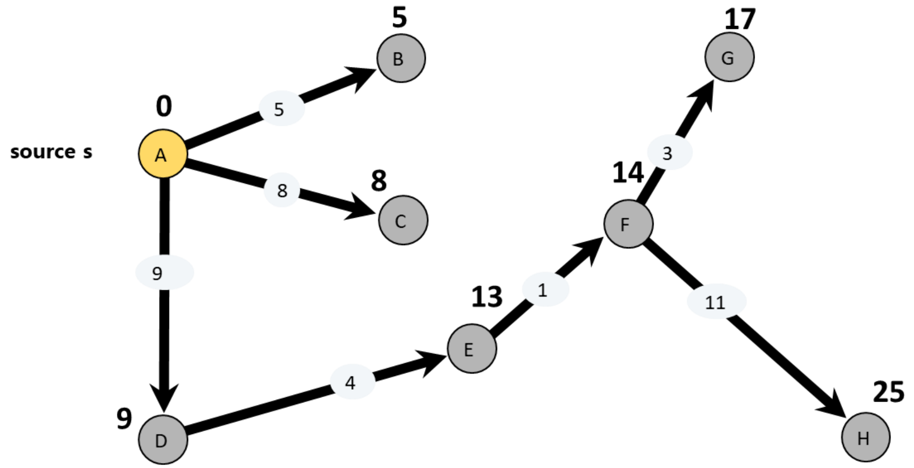
Single-source shortest paths problem



	distance[ ]
A	**0**
B	5
C	8
D	9
E	13
F	14
G	17
H	25

	prev[ ]
A	
B	A
C	A
D	A
E	D
F	E
G	F
H	F

Single-source shortest paths problem

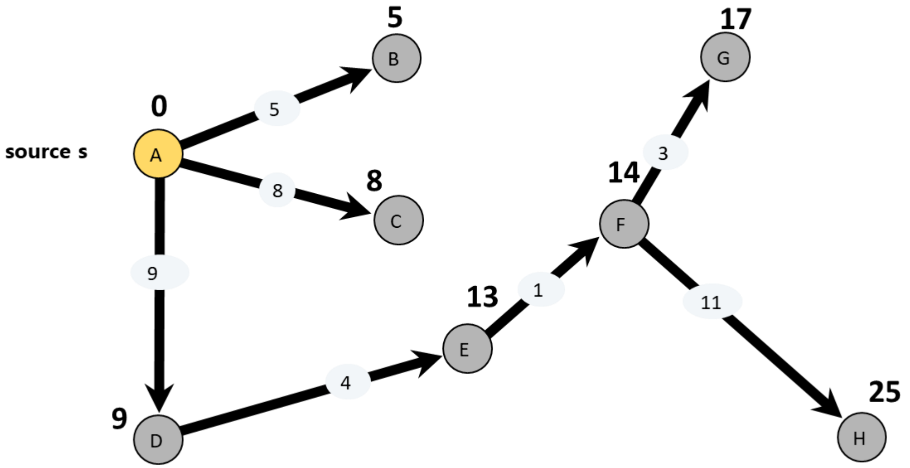


Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$P[1 \ldots n] \leftarrow [null \ \ldots \ null]$
$d[s] \leftarrow 0 \qquad E \leftarrow \{s\}$
while $E \neq V$ **do**
  Select $e = (u, v)$, $u \in E$, $v \notin E$ with minimum $d[u] + w(uv)$ O(E)
  $E \leftarrow E \cup \{v\}$
  $d[v] \leftarrow d[u] + w(uv)$
  $P[v] \leftarrow u$

O(V)

Single-source shortest paths problem



Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

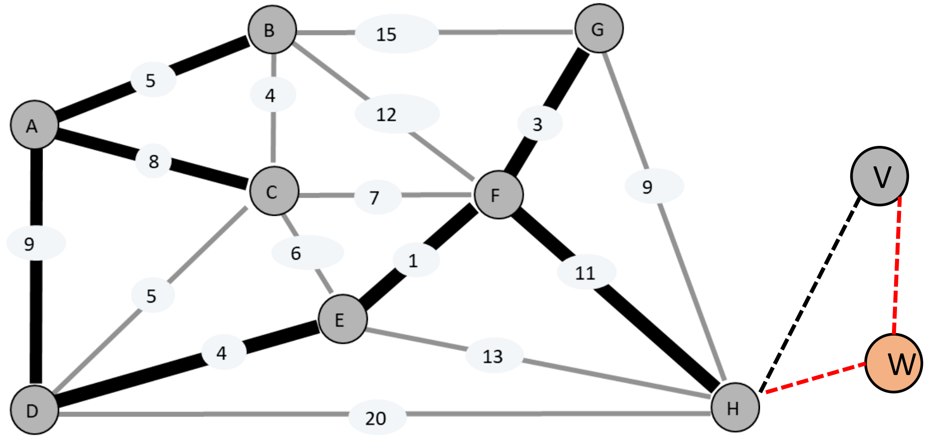
$$\left.\begin{array}{l}
d[1\ldots n] \leftarrow [\infty \ldots \infty] \\
P[1\ldots n] \leftarrow [null \ \ldots \ null] \\
d[s] \leftarrow 0 \qquad E \leftarrow \{s\} \\
\textbf{while } E \neq V \textbf{ do} \\
\quad \text{Select } e = (u, v),\ u \in E,\ v \notin E \text{ with minimum } d[u] + w(uv) \\
\quad E \leftarrow E \cup \{v\} \\
\quad d[v] \leftarrow d[u] + w(uv) \\
\quad P[v] \leftarrow u
\end{array}\right\} \mathbf{O}(\boldsymbol{EV})$$

Dijkstra's algorithm: Proof of correctness

- **Proof:** [by induction on $|S|$]
- **Base case:** $|S| = 1$ is easy since $S = \{ s \}$ and $d[s] = 0$.
- **Inductive hypothesis:** Assume that for some $\boldsymbol{k}$, the shortest path distance to the first $\boldsymbol{k}$ vertices added to $\boldsymbol{E}$ (let's call this subset $\boldsymbol{E_k}$) is correct. We will prove that the $\boldsymbol{d}[\boldsymbol{v}]$ for the **$\boldsymbol{k}$+1th** vertex $\boldsymbol{v}$ (next closest vertex) added to $\boldsymbol{E}$ is also correct.

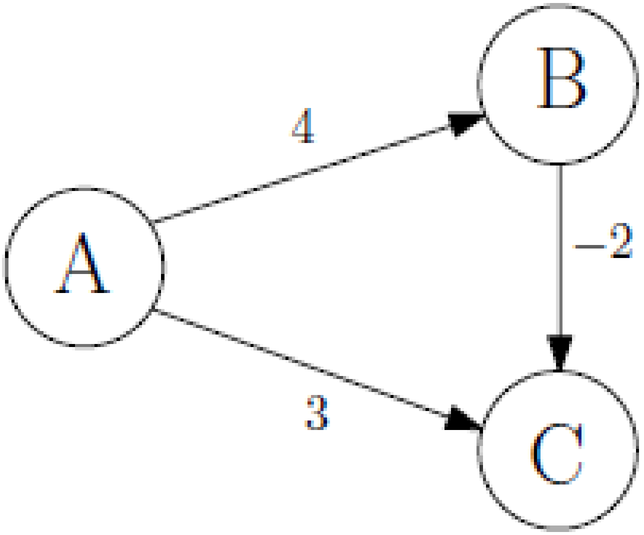
Steps:

- When vertex v added to S_k, it's chosen because it has the smallest $d[v]$ value among all vertices not in E_k
- Now, either the shortest path from s to v only goes through E_k, or it goes through at least one other node w (not in E_k)
- If the shortest path passes through w then $d[w] < d[v]$, it contradicts our inductive hypothesis
- Therefore, when the algorithm adds v to E_k, $d[v]$ correctly represents the shortest path from s to v



Live Poll 1: Dijkstra Algorithm

What paths do we get if we run Dijkstra on vertex A?



https://forms.office.com/r/bdpy1hjsK7

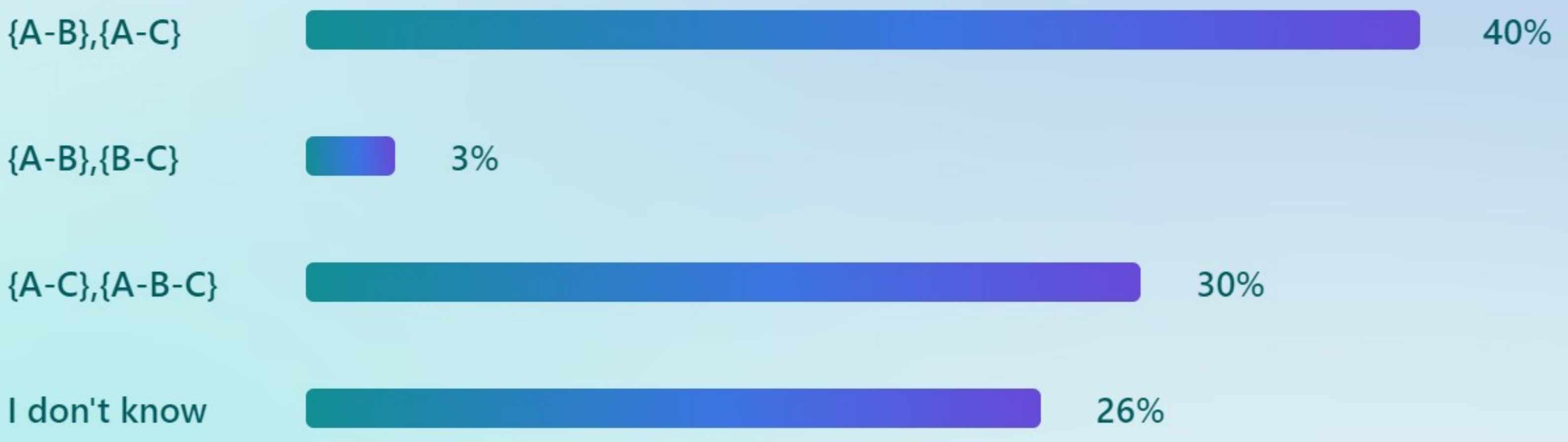
Scan the QR or use link to join

https://forms.office.com/r/bdpy1hjsK7

Copy link

31 responses submitted

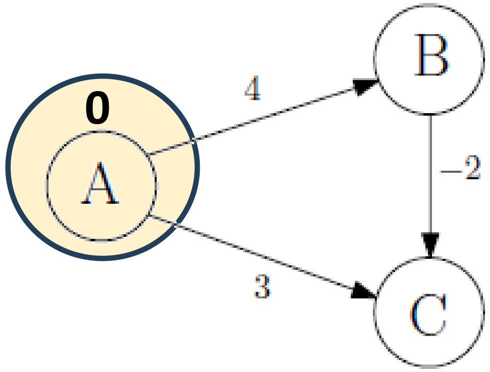
What paths do we get if we run Dijkstra on vertex A?



Treemap Bar

1 of 1

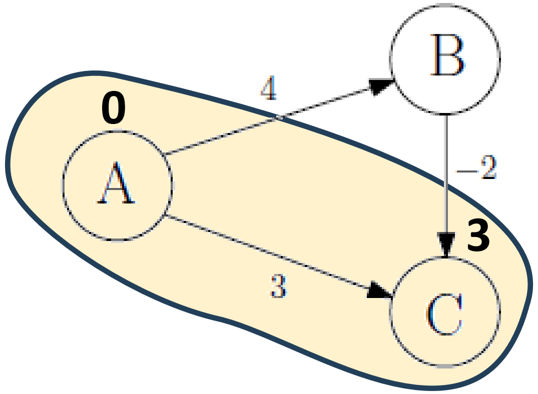
Dijkstra cannot handle *negative edges*



Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

$d[1\ldots n] \leftarrow [\infty \ldots \infty]$
$P[1\ldots n] \leftarrow [null \; \ldots \; null]$
$d[s] \leftarrow 0$ $\quad E \leftarrow \{s\}$
while $E \neq V$ **do**
  Select $e = (u, v)$, $u \in E$, $v \notin E$ with minimum $d[u] + w(uv)$
  $E \leftarrow E \cup \{v\}$
  $d[v] \leftarrow d[u] + w(uv)$
  $P[v] \leftarrow u$

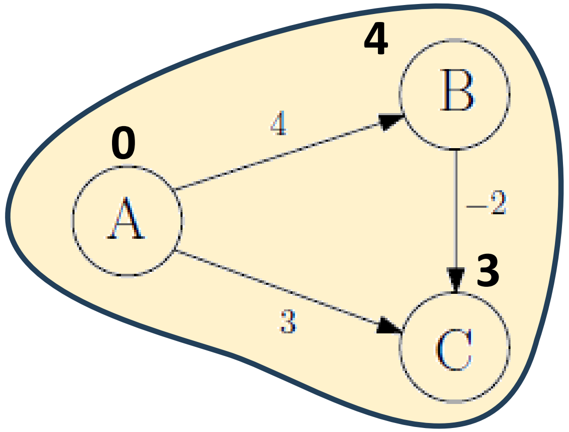
Dijkstra cannot handle *negative edges*



Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$P[1 \ldots n] \leftarrow [null \ \ldots \ null]$
$d[s] \leftarrow 0 \qquad E \leftarrow \{s\}$
while $E \neq V$ **do**
    Select $e = (u, v)$, $u \in E$, $v \notin E$ with minimum $d[u] + w(uv)$
    $E \leftarrow E \cup \{v\}$
    $d[v] \leftarrow d[u] + w(uv)$
    $P[v] \leftarrow u$

Dijkstra cannot handle *negative edges*



Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

$d[1\ldots n] \leftarrow [\infty \ldots \infty]$
$P[1\ldots n] \leftarrow [null \;\ldots\; null]$
$d[s] \leftarrow 0 \qquad E \leftarrow \{s\}$
while $E \neq V$ **do**
  Select $e = (u, v)$, $u \in E$, $v \notin E$ with minimum $d[u] + w(uv)$
  $E \leftarrow E \cup \{v\}$
  $d[v] \leftarrow d[u] + w(uv)$
  $P[v] \leftarrow u$

Can we improve the time complexity of Dijkstra?

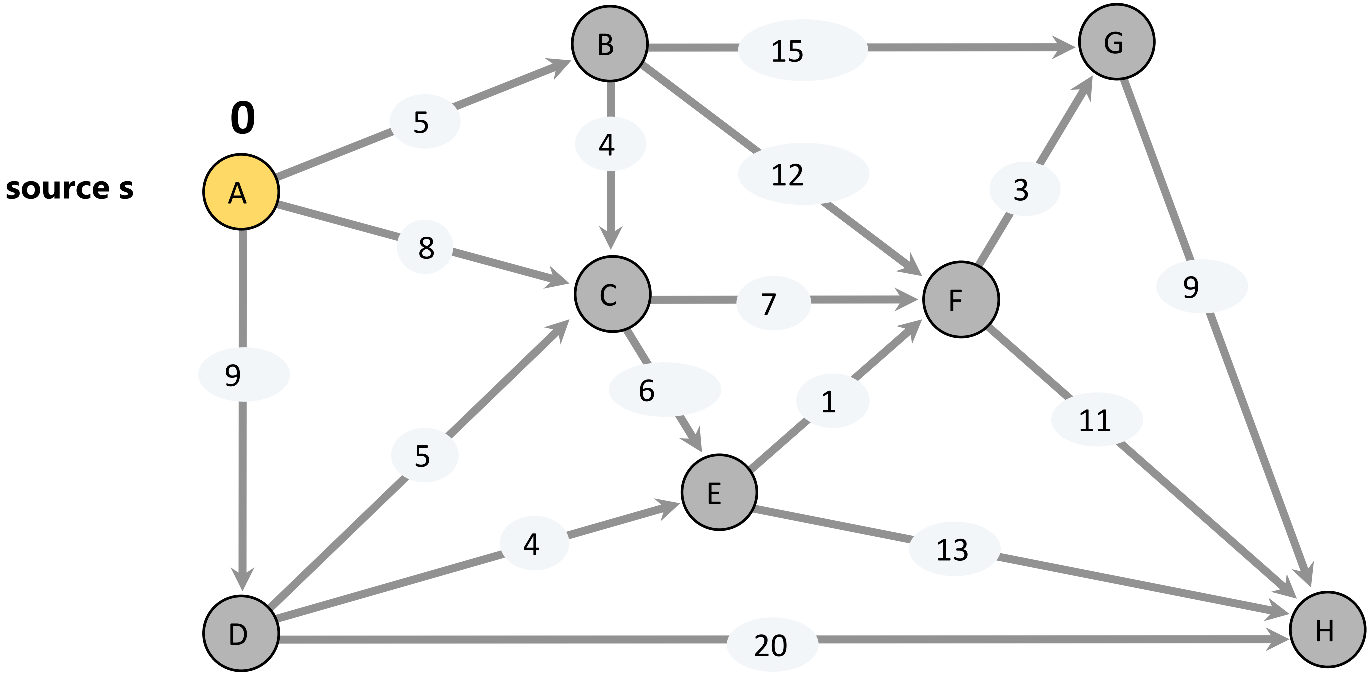
Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$P[1 \ldots n] \leftarrow [null \; \ldots \; null]$
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d[u] + w(uv)$
  $P[v] \leftarrow u$

This operation is very expensive

Using Priority Queue

Single-source shortest paths problem



distance[ ]

A	**0**
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞
H	∞

Priority Queue

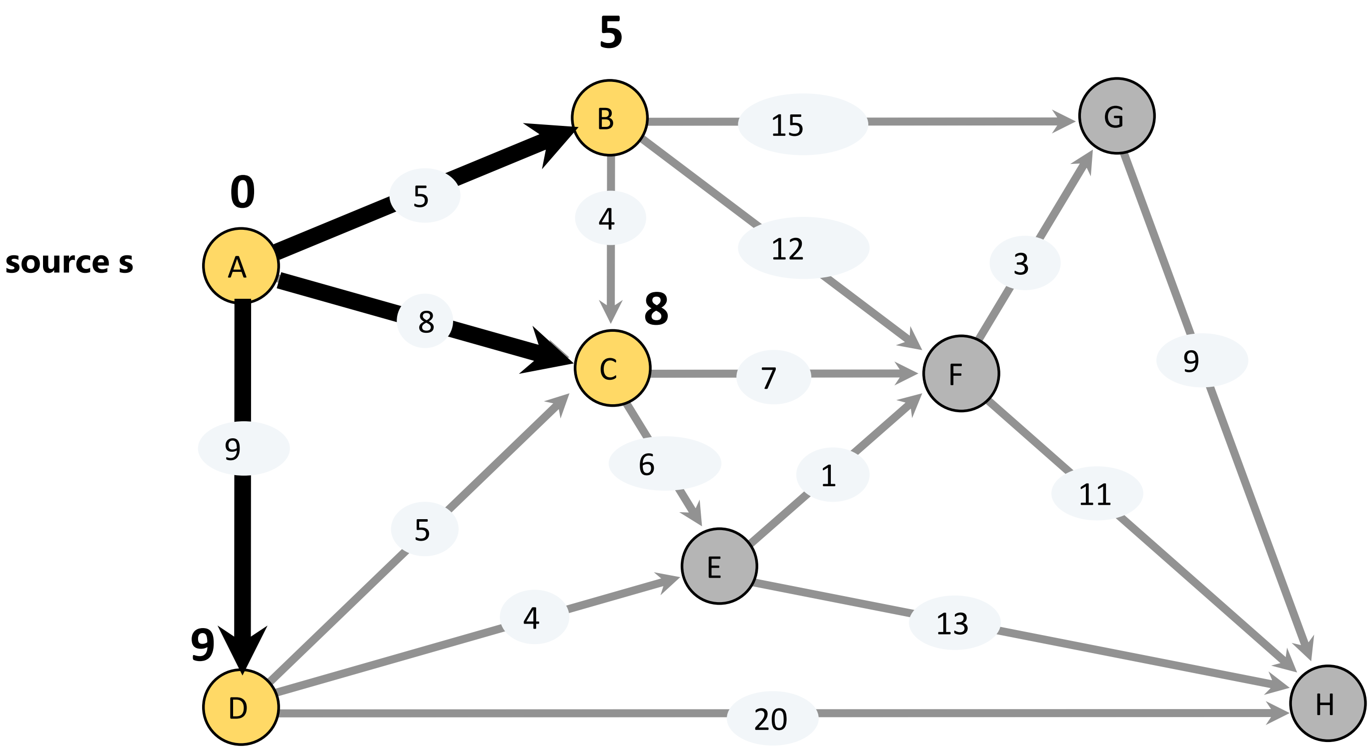
A	**0**
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞
H	∞

predecessor[ ]

A	
B	
C	
D	
E	
F	
G	
H	

```
if (distance[u] + weight(u, v) < distance[V])
       distance[V]= distance[u] + weight(u, v)
```

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	∞
F	∞
G	∞
H	∞

Priority Queue

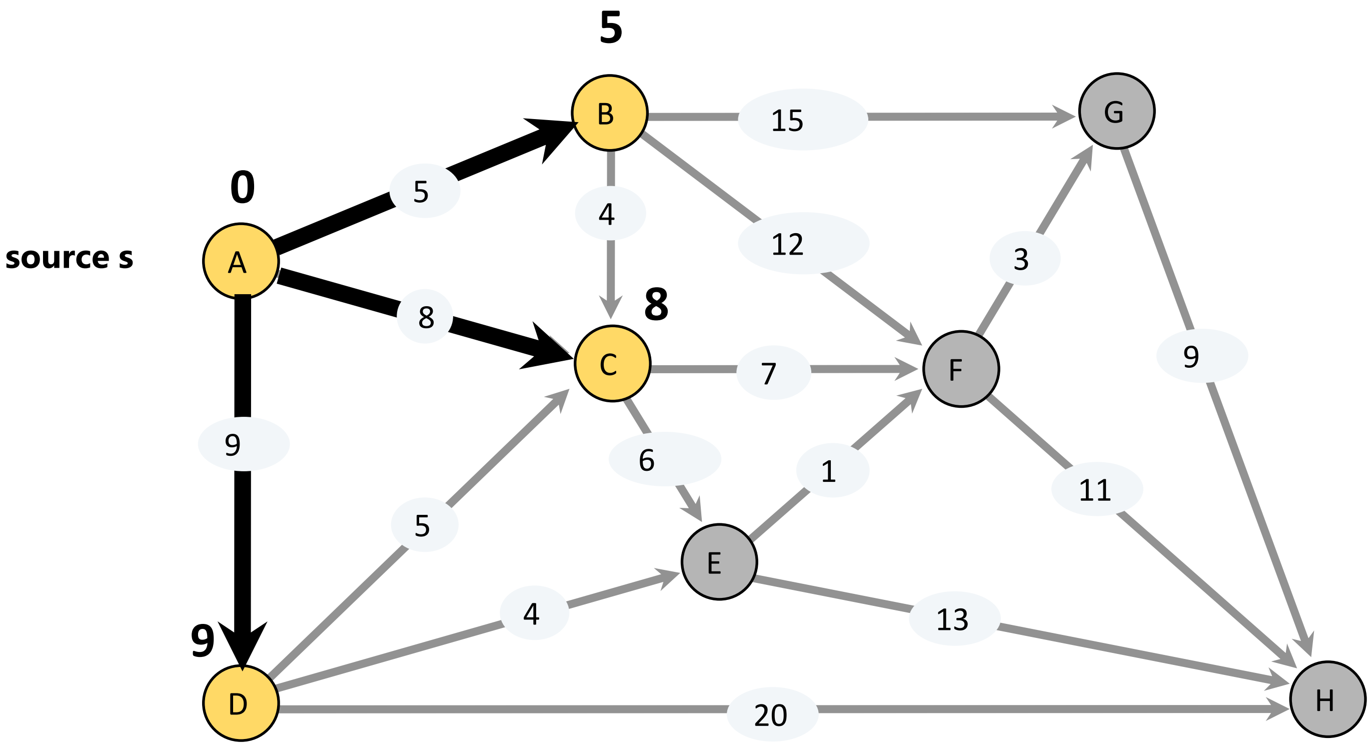
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞
H	∞

predecessor[ ]

A	
B	A
C	A
D	A
E	
F	
G	
H	

```
if (distance[u] + weight(u, v) < distance[V])
        distance[V]= distance[u] + weight(u, v)
```

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	∞
F	∞
G	∞
H	∞

Priority Queue

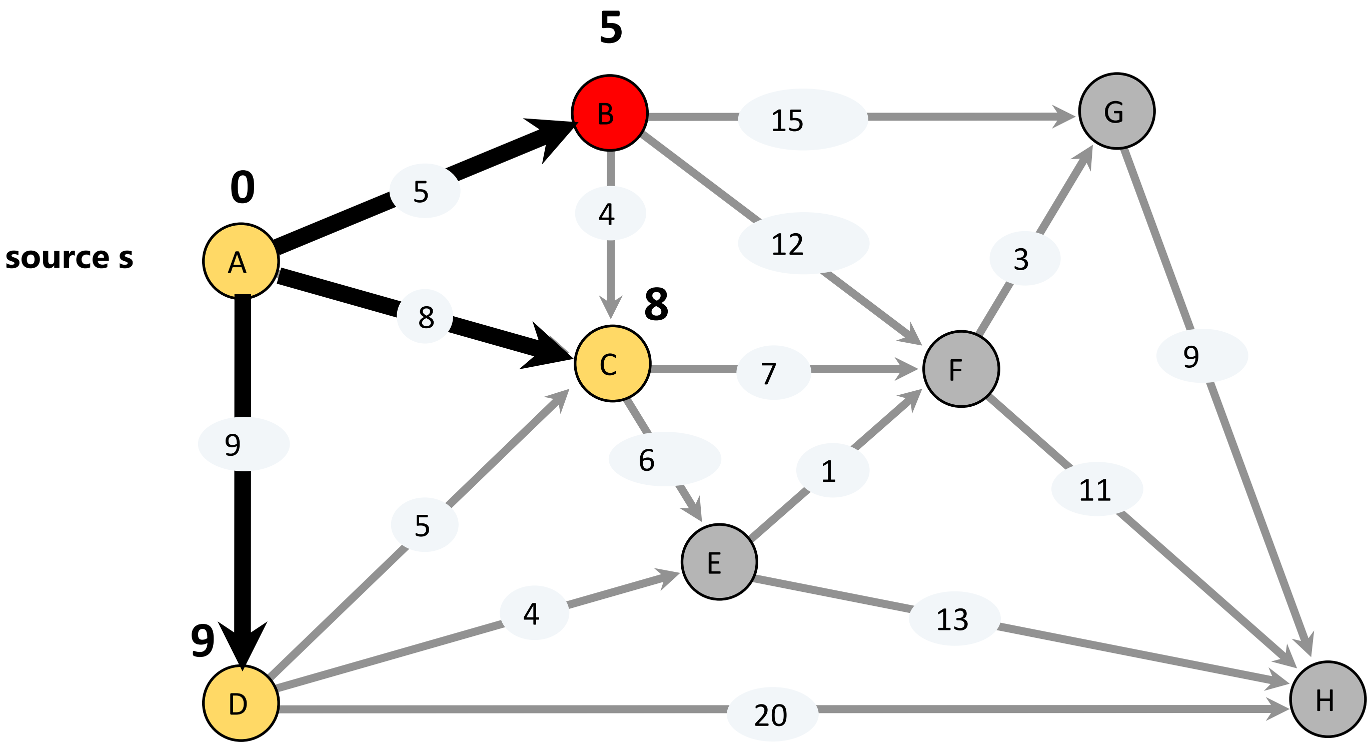
B	5
C	8
D	9
E	∞
F	∞
G	∞
H	∞

predecessor[ ]

A	
B	A
C	A
D	A
E	
F	
G	
H	

```
if (distance[u] + weight(u, v) < distance[V])
        distance[V]= distance[u] + weight(u, v)
```

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	∞
F	∞
G	∞
H	∞

Priority Queue

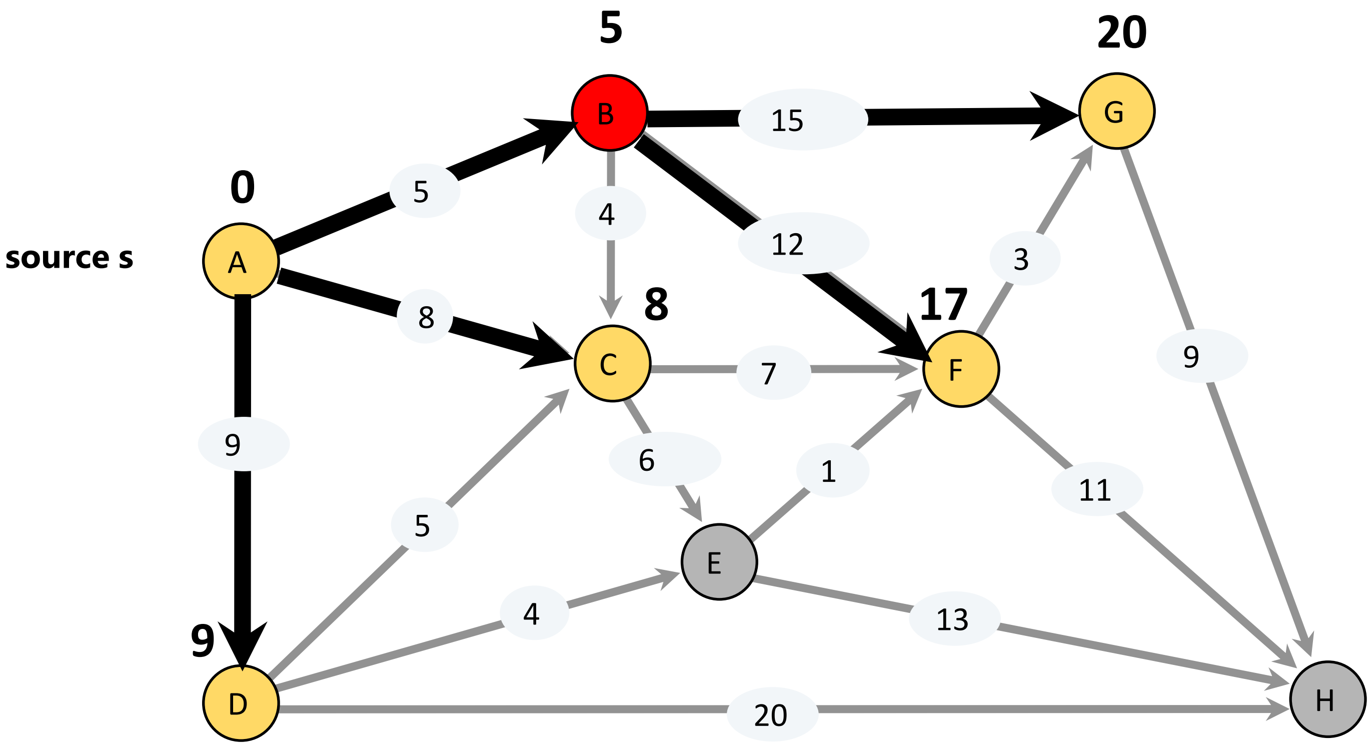
C	8
D	9
E	∞
F	∞
G	∞
H	∞

predecessor[ ]

A	
B	A
C	A
D	A
E	
F	
G	
H	

```
if (distance[u] + weight(u, v) < distance[V])
        distance[V]= distance[u] + weight(u, v)
```

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	∞
F	17
G	20
H	∞

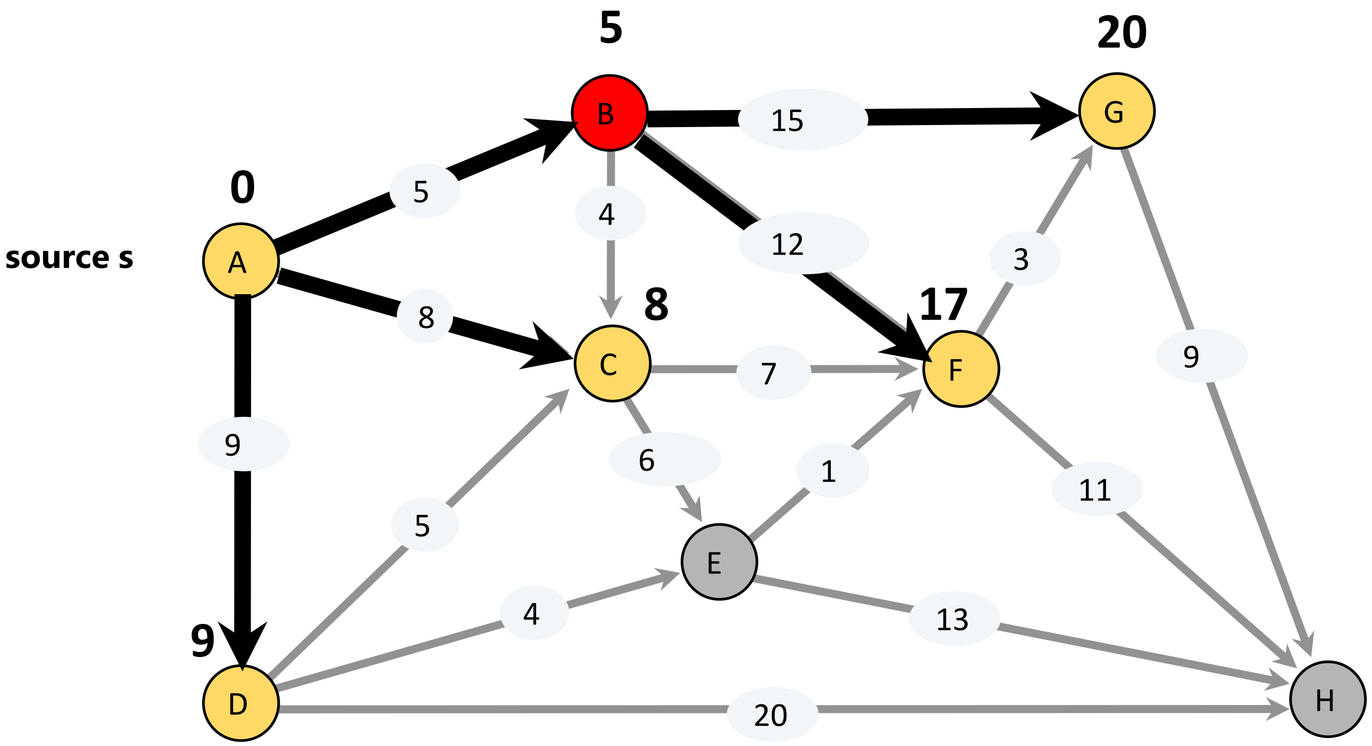
Priority Queue

C	8
D	9
E	∞
F	∞
G	∞
H	∞

predecessor[ ]

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

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	∞
F	17
G	20
H	∞

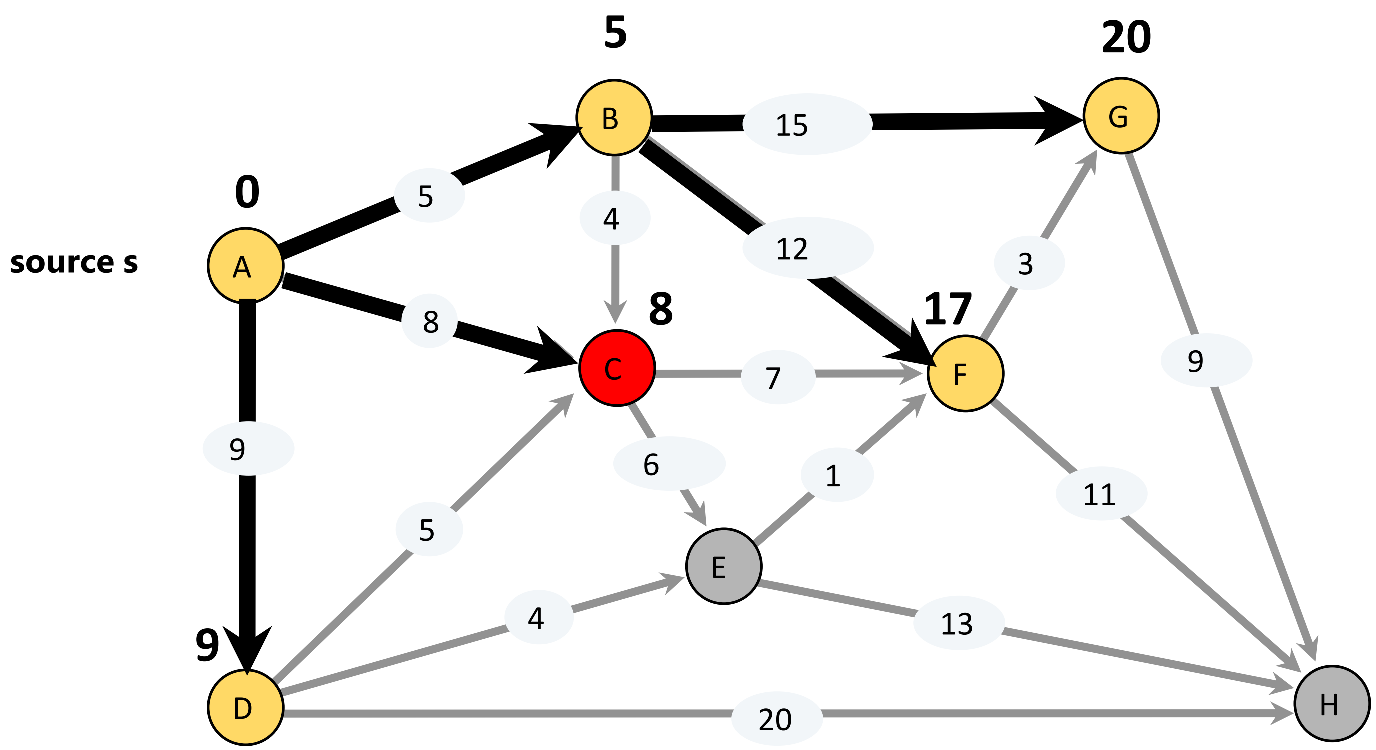
Priority Queue

C	8
D	9
F	17
G	20
E	∞
H	∞

predecessor[ ]

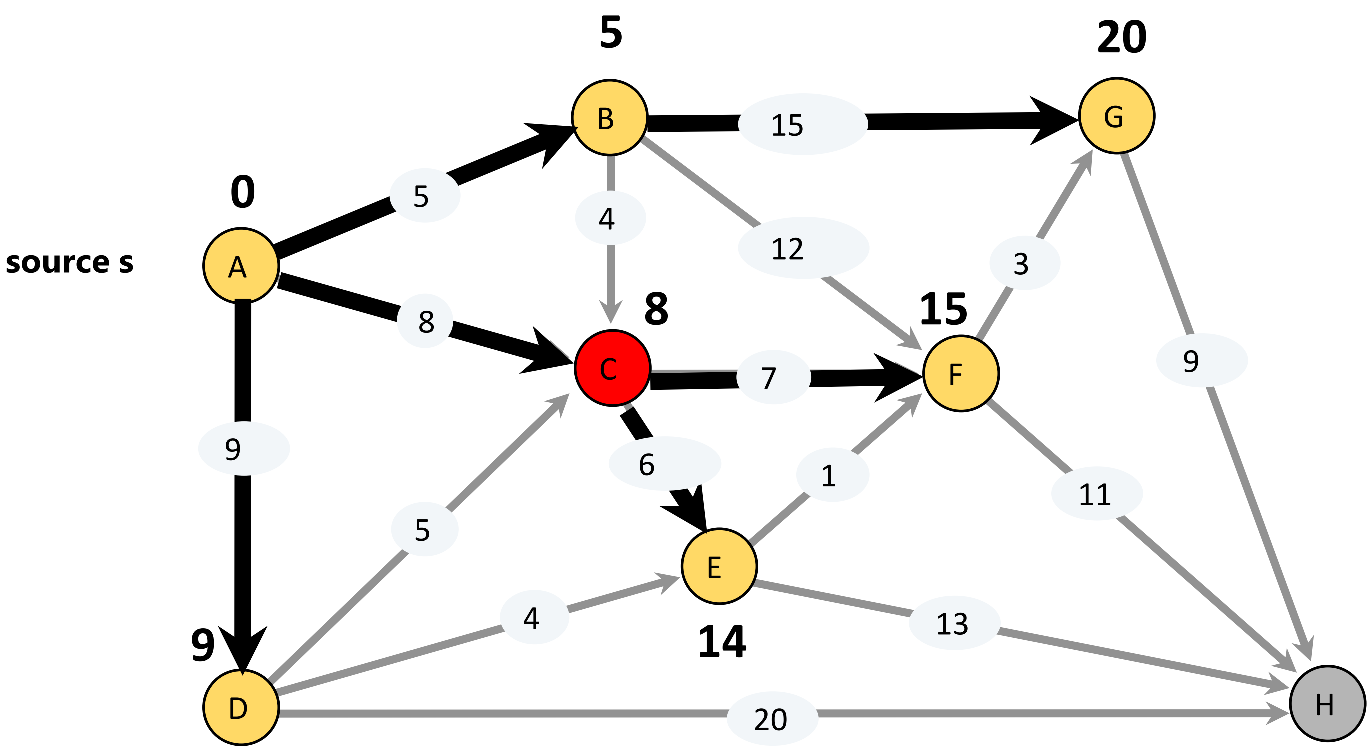
A	
B	A
C	A
D	A
E	
F	B
G	B
H	

Single-source shortest paths problem



	distance[ ]	Priority Queue			predecessor[ ]
A	**0**	**D**	9	**A**	
B	5	F	17	B	A
C	8	G	20	C	A
D	9	E	∞	D	A
E	∞	H	∞	E	
F	17			F	B
G	20			G	B
H	∞			H	

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	14
F	15
G	20
H	∞

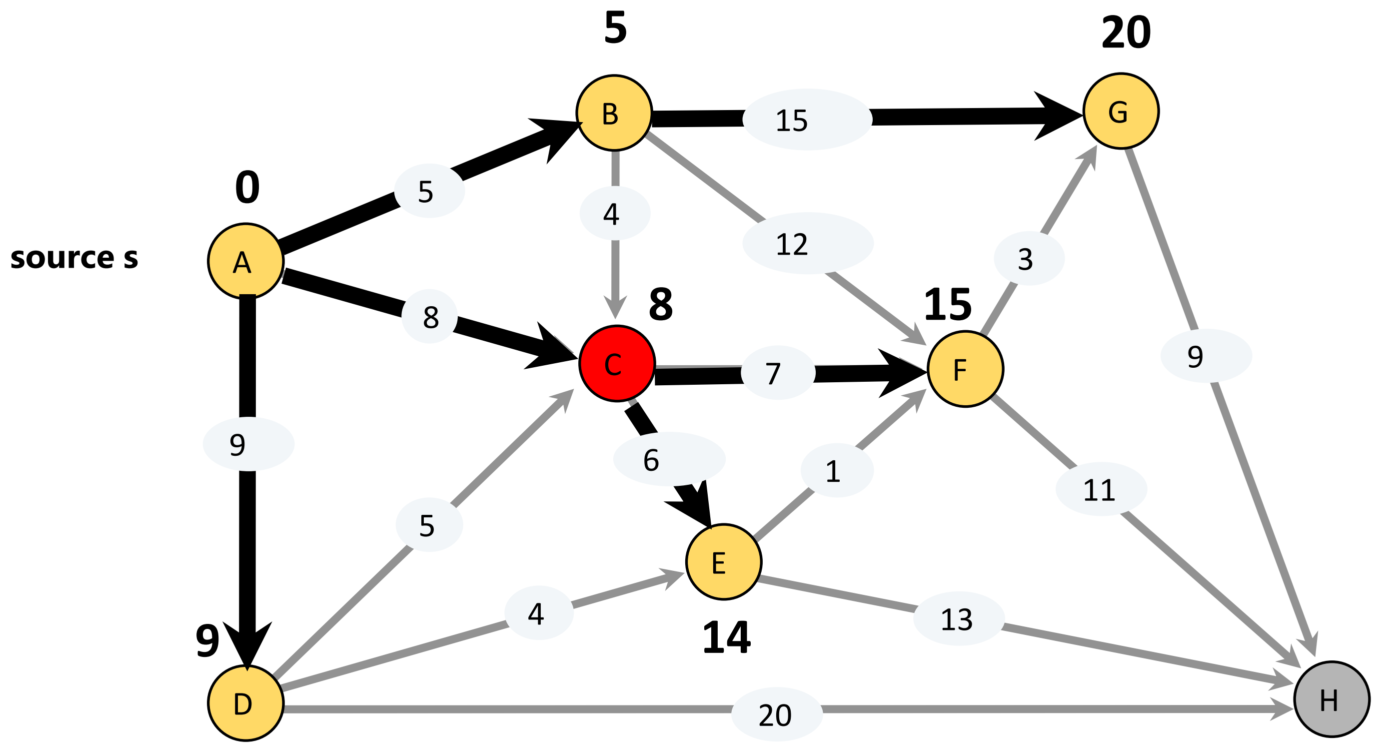
Priority Queue

D	9
F	17
G	20
E	∞
H	∞

predecessor[ ]

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

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	14
F	15
G	20
H	∞

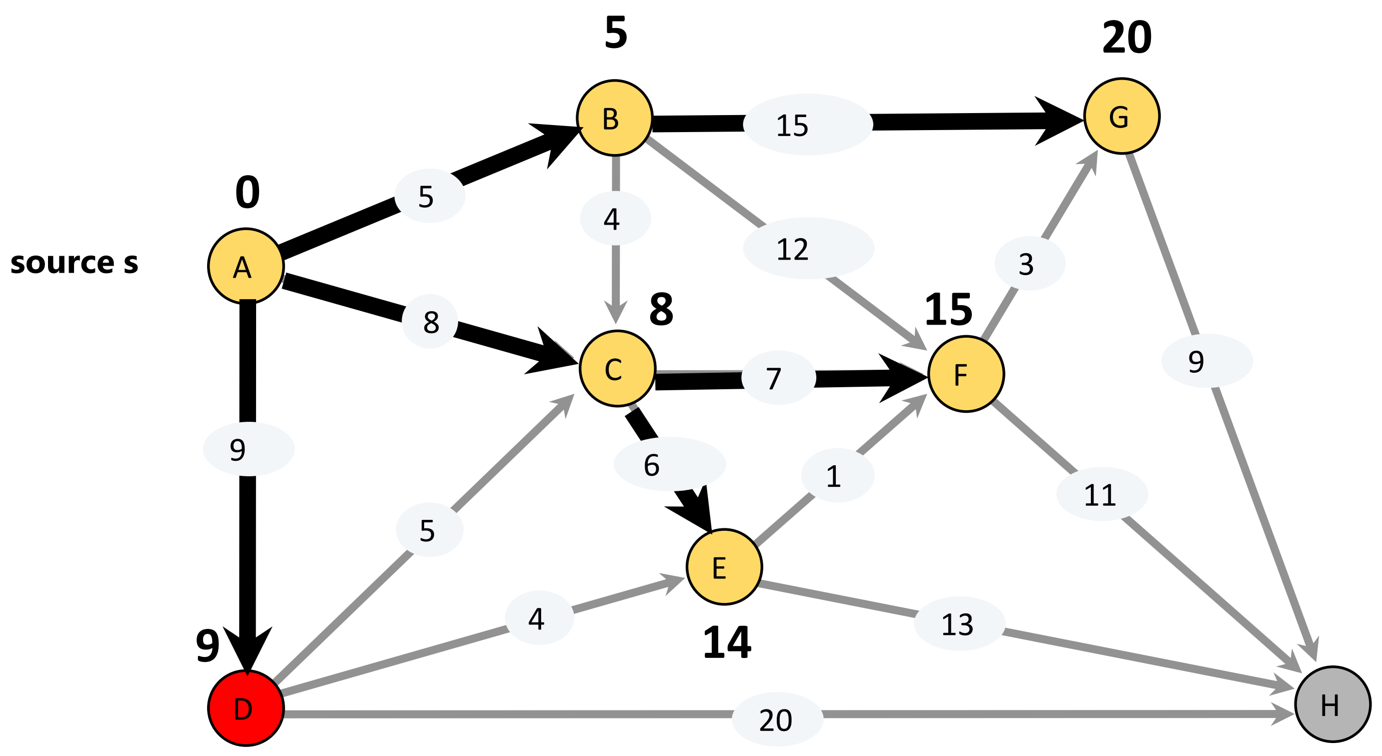
Priority Queue

D	9
E	14
F	15
G	20
H	∞

predecessor[ ]

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

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	14
F	15
G	20
H	∞

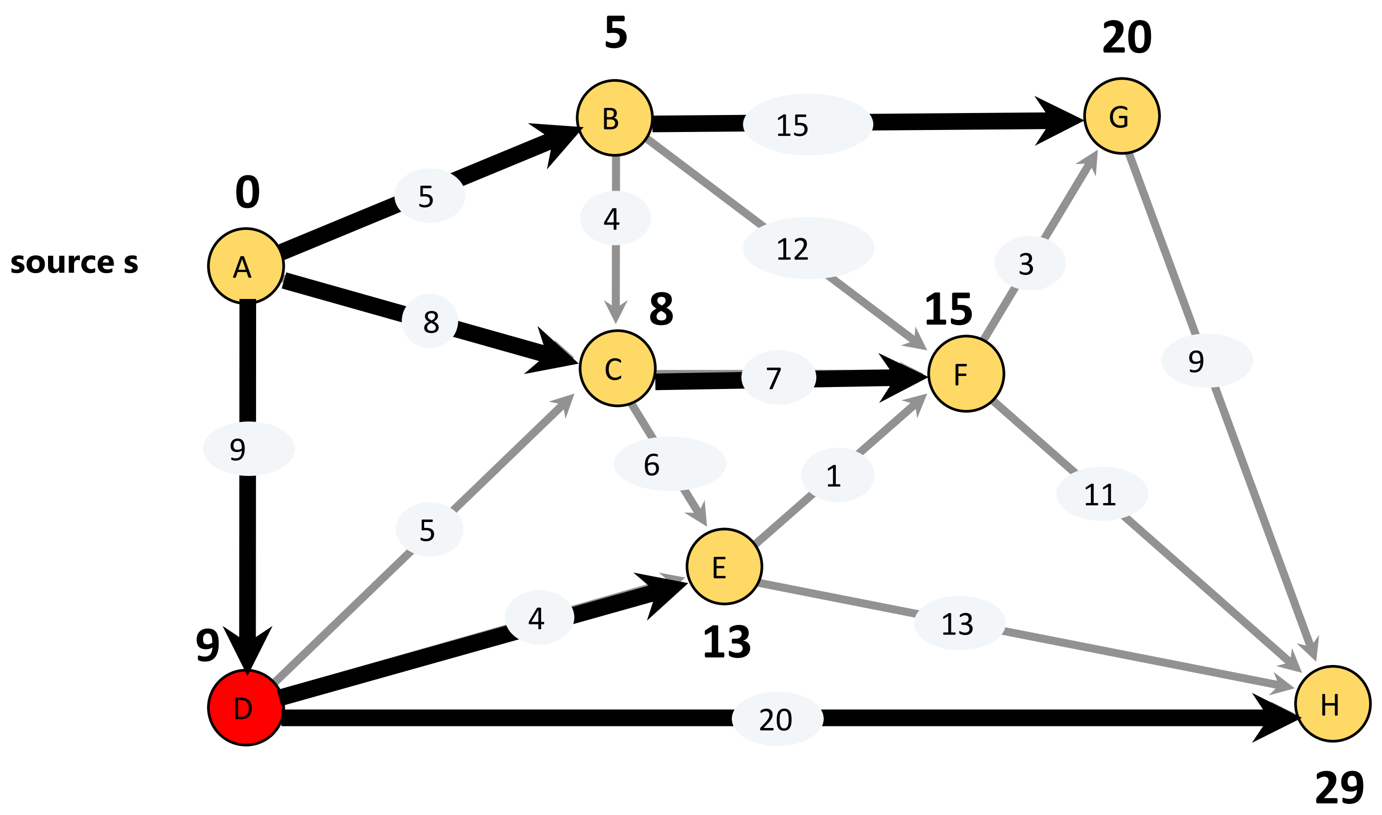
Priority Queue

E	14
F	15
G	20
H	∞

predecessor[ ]

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

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	13
F	15
G	20
H	29

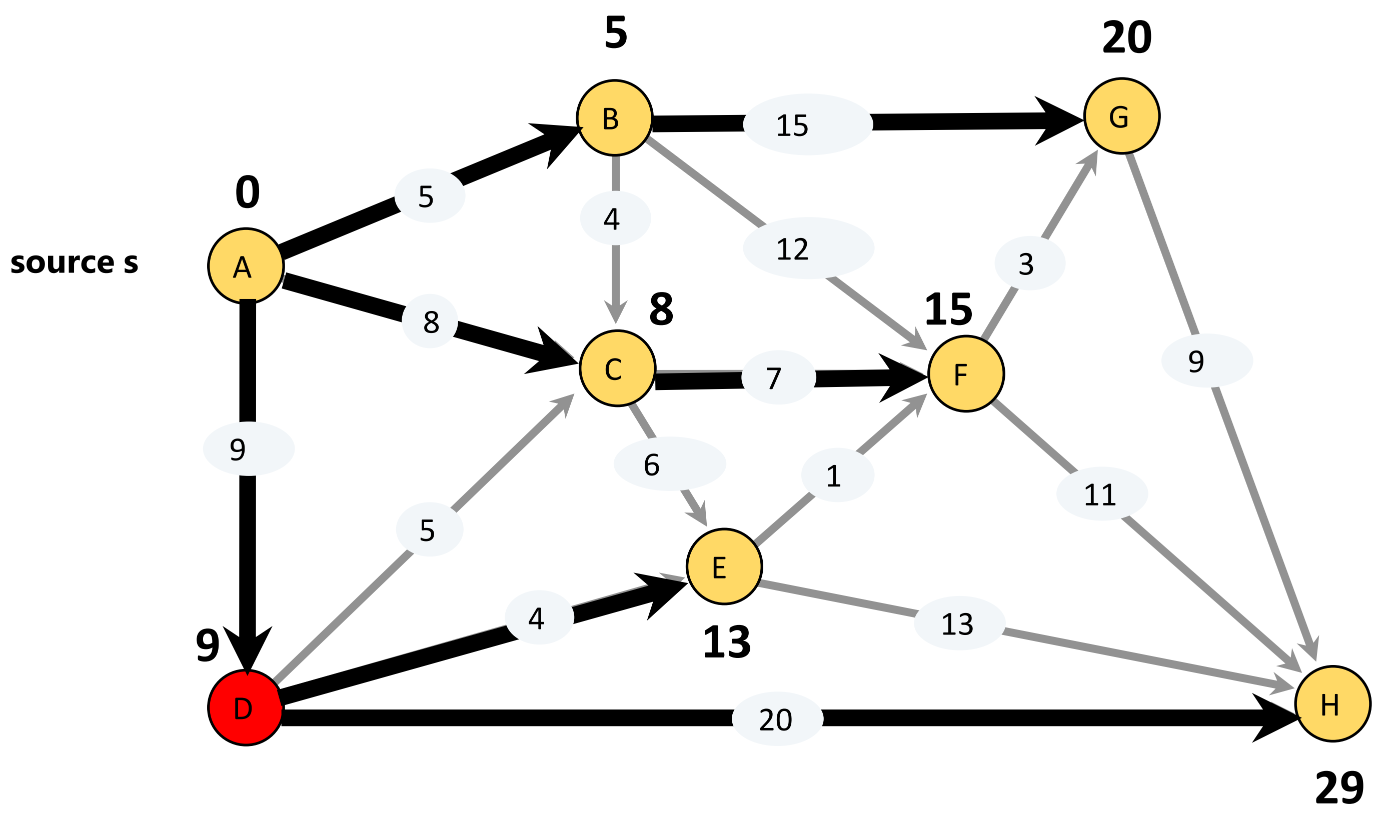
Priority Queue

E	14
F	15
G	20
H	∞

predecessor[ ]

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

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	13
F	15
G	20
H	29

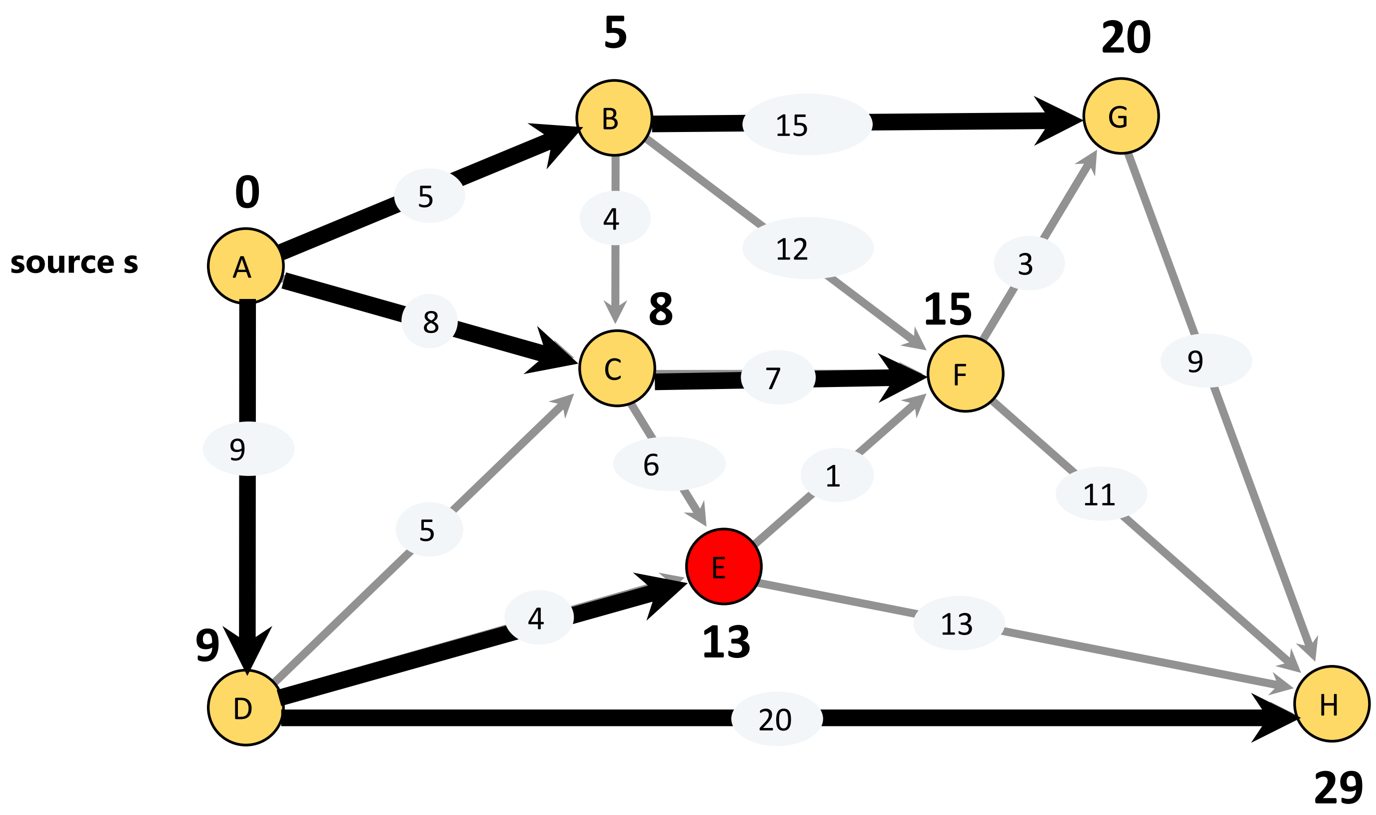
Priority Queue

E	13
F	15
G	20
H	29

predecessor[ ]

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

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	13
F	15
G	20
H	29

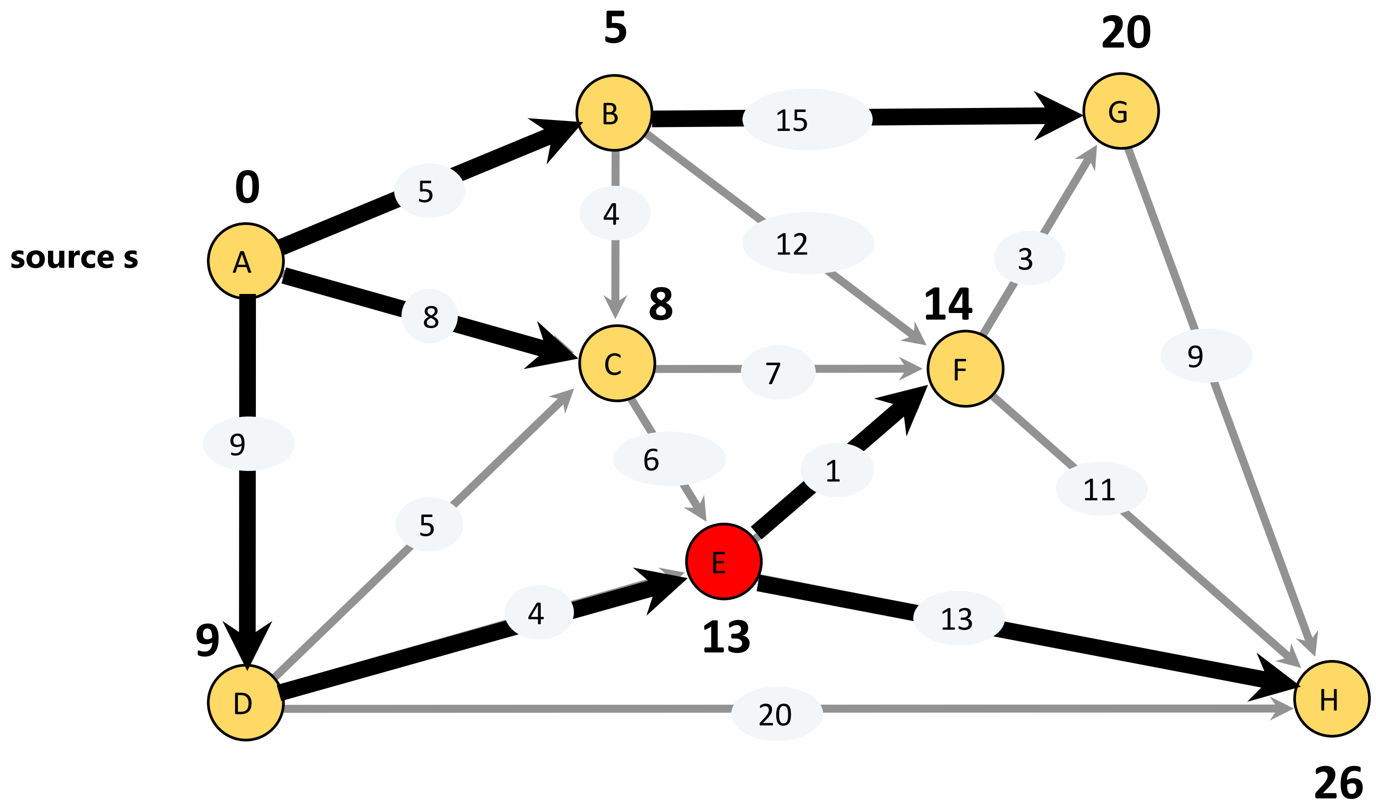
Priority Queue

F	15
G	20
H	29

predecessor[ ]

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

Single-source shortest paths problem



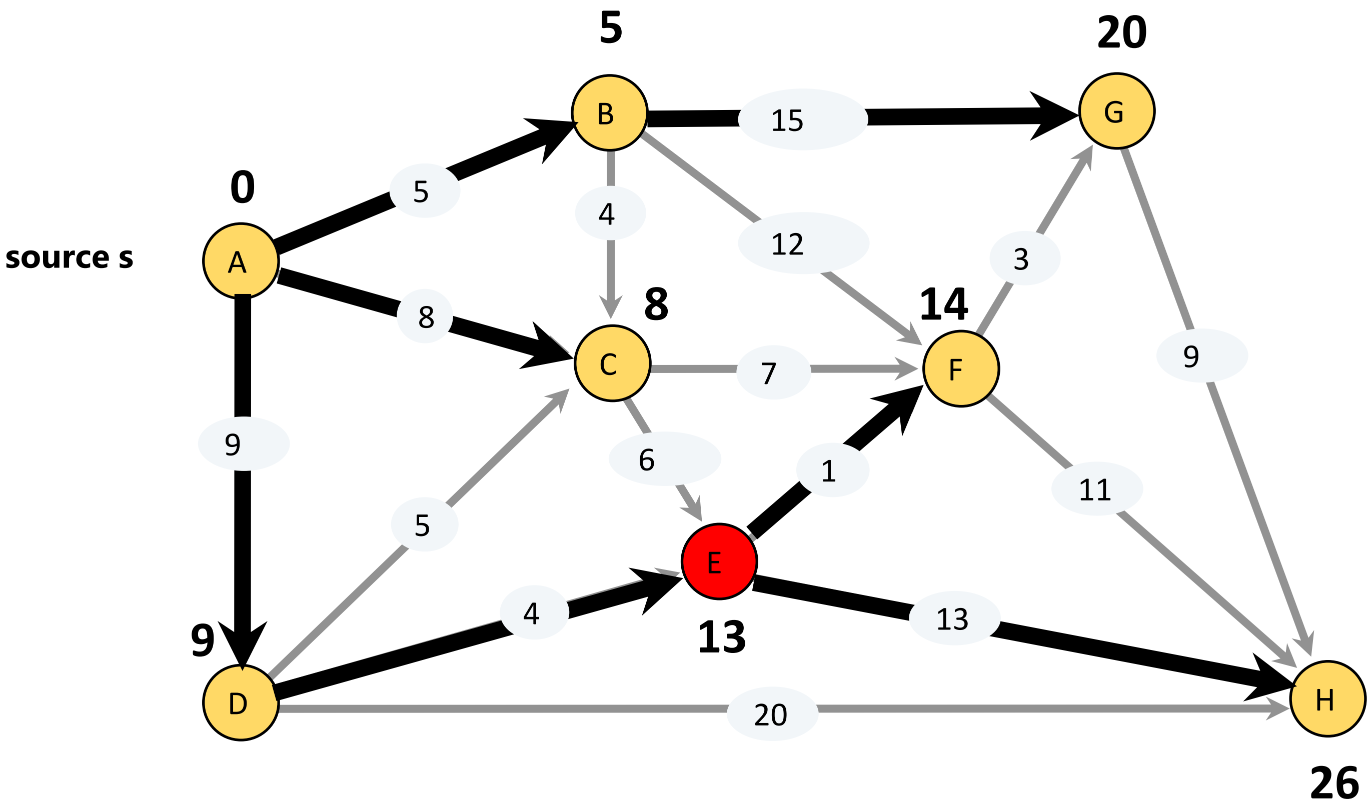
distance[ ]	
A	**0**
B	5
C	8
D	9
E	13
F	14
G	20
H	26

Priority Queue

F	15
G	20
H	29

predecessor[ ]	
A	
B	A
C	A
D	A
E	D
F	E
G	B
H	E

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	13
F	14
G	20
H	26

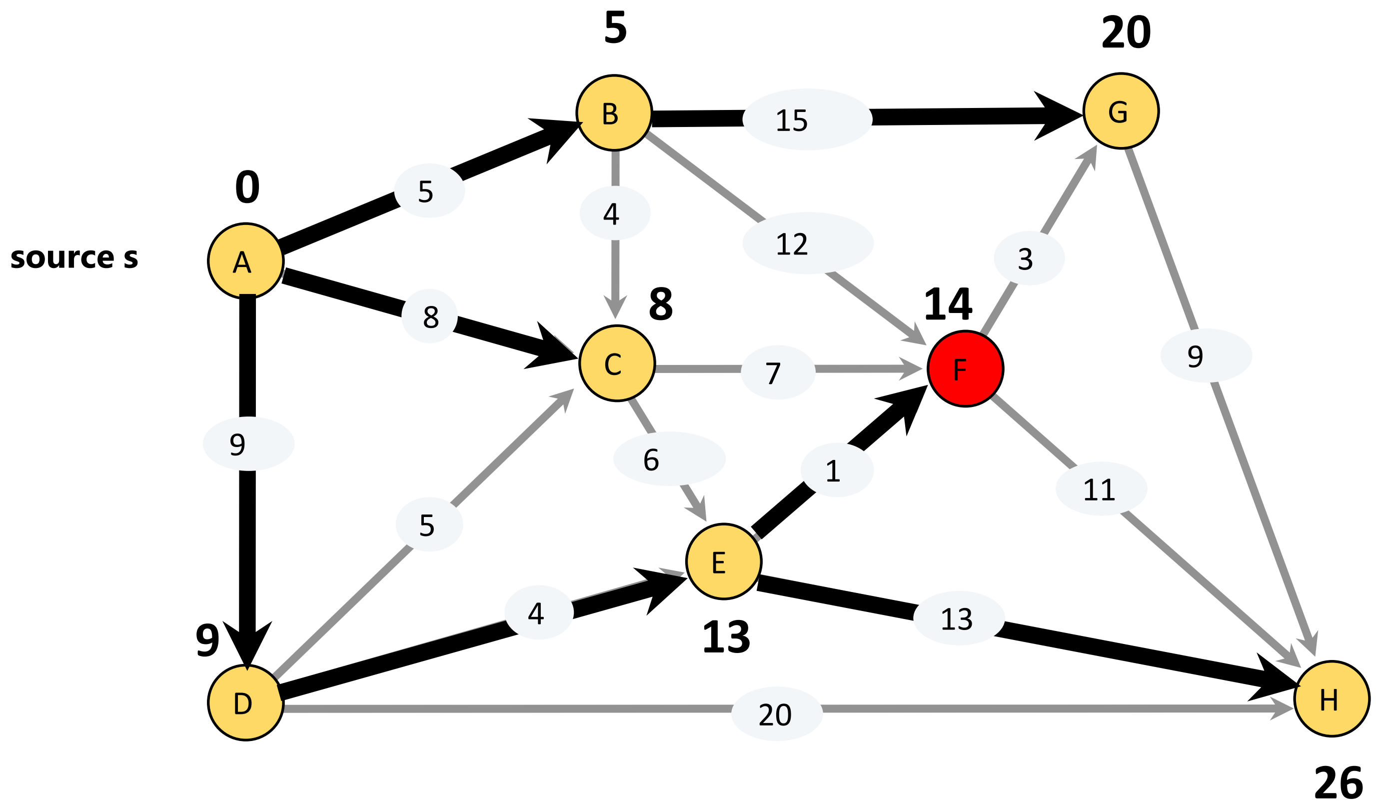
Priority Queue

F	14
G	20
H	26

predecessor[ ]

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

Single-source shortest paths problem



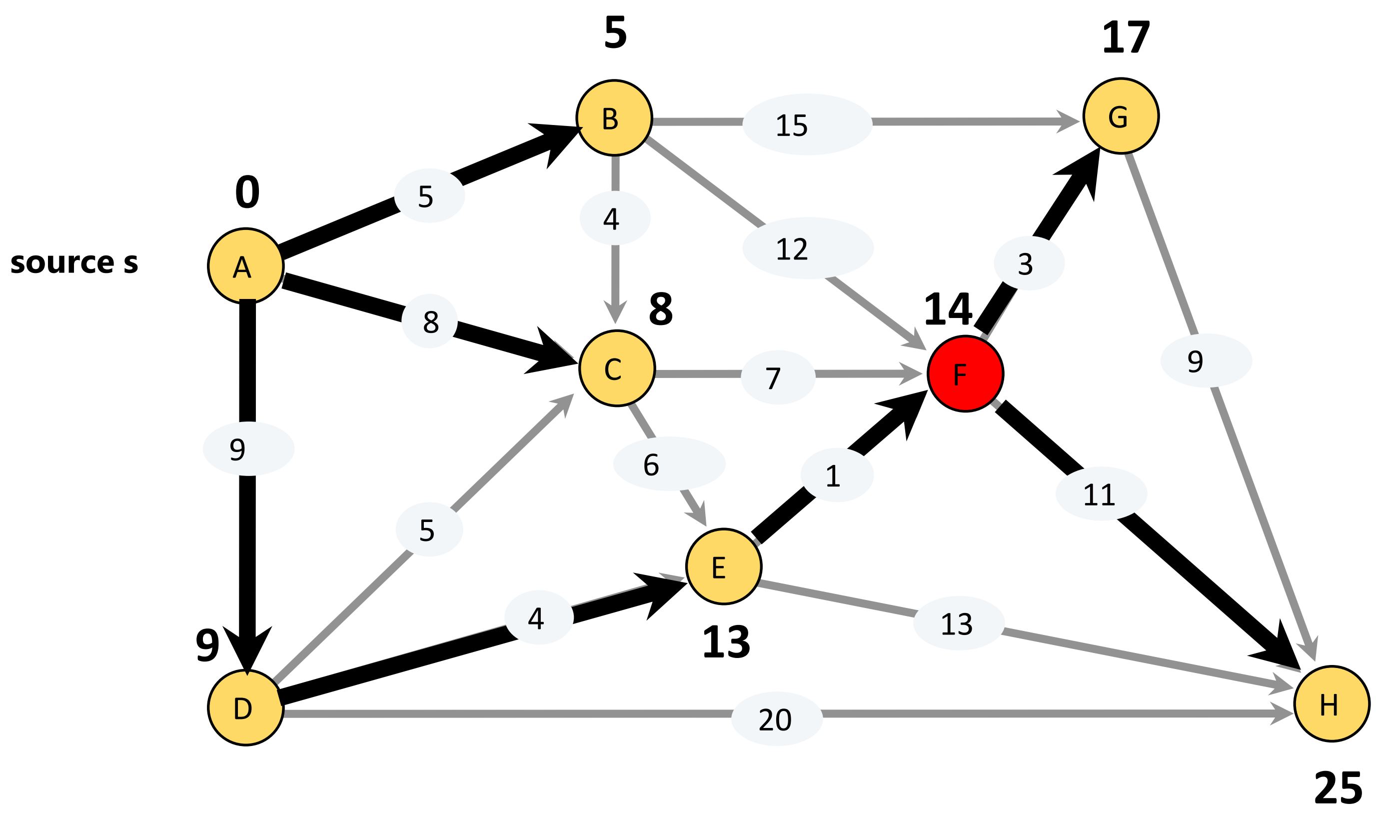
	distance[ ]
A	**0**
B	5
C	8
D	9
E	13
F	14
G	20
H	26

Priority Queue

G	20
H	26

	predecessor[ ]
A	
B	A
C	A
D	A
E	D
F	E
G	B
H	E

Single-source shortest paths problem



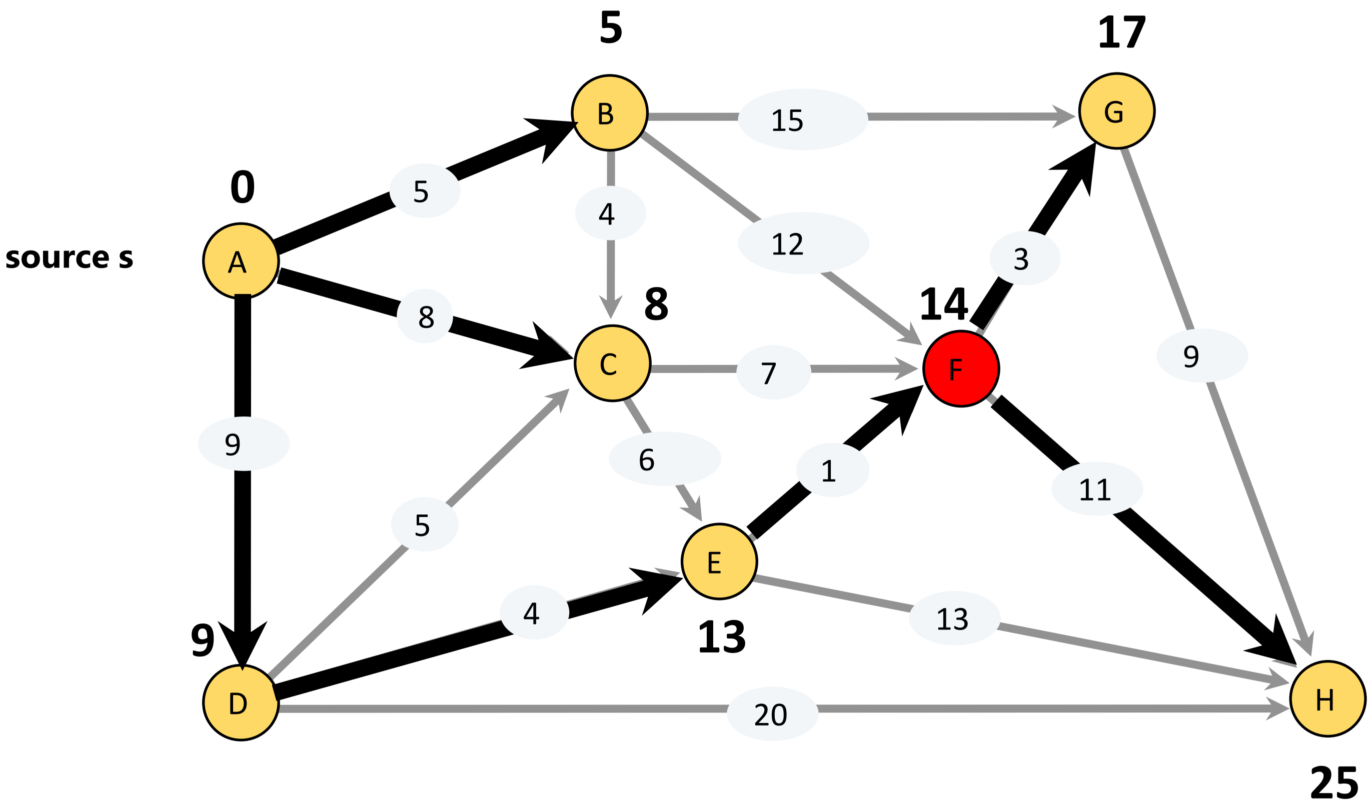
	distance[ ]
A	**0**
B	5
C	8
D	9
E	13
F	14
G	17
H	25

Priority Queue

G	20
H	26

	predecessor[ ]
A	
B	A
C	A
D	A
E	D
F	E
G	F
H	F

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	13
F	14
G	17
H	25

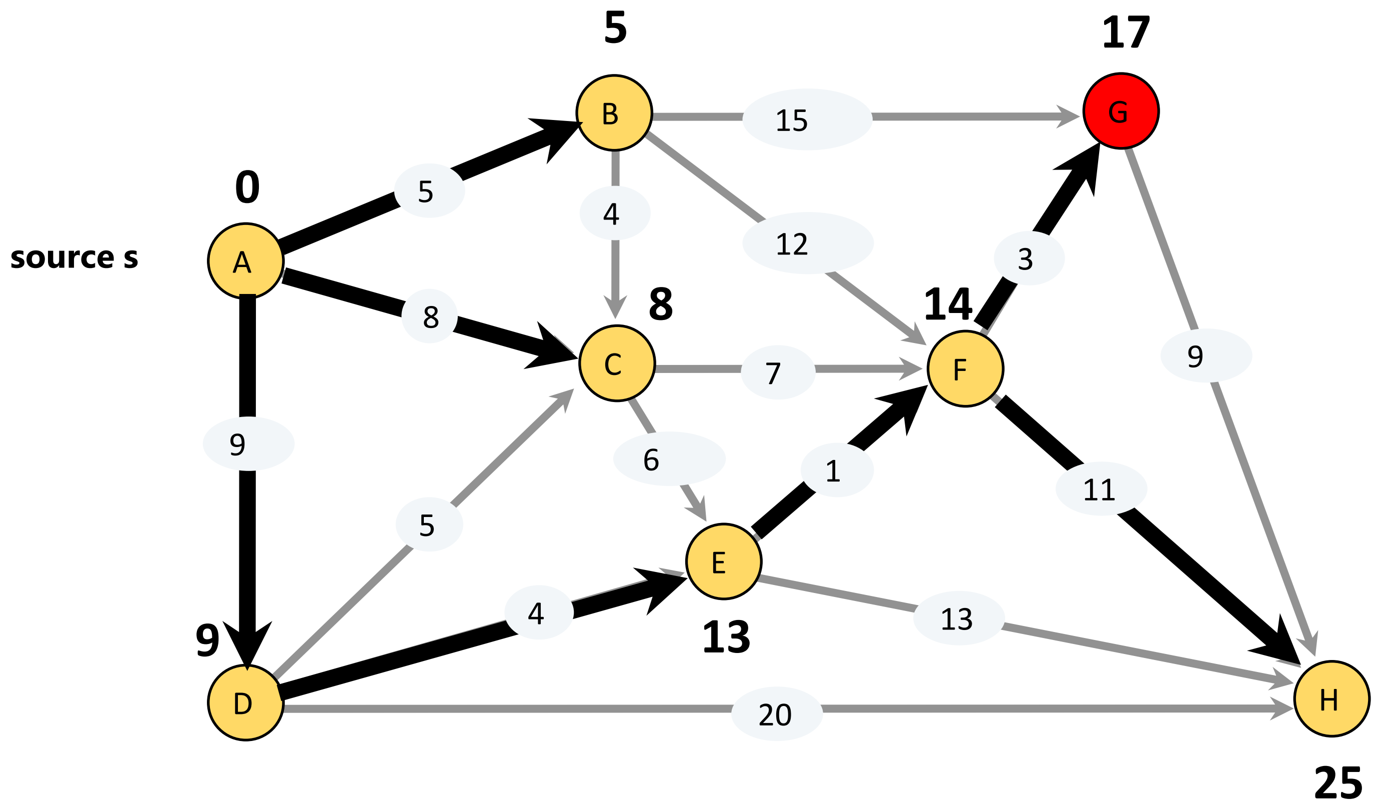
Priority Queue

G	17
H	25

predecessor[ ]

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

Single-source shortest paths problem



distance[ ]

A	**0**
B	5
C	8
D	9
E	13
F	14
G	17
H	25

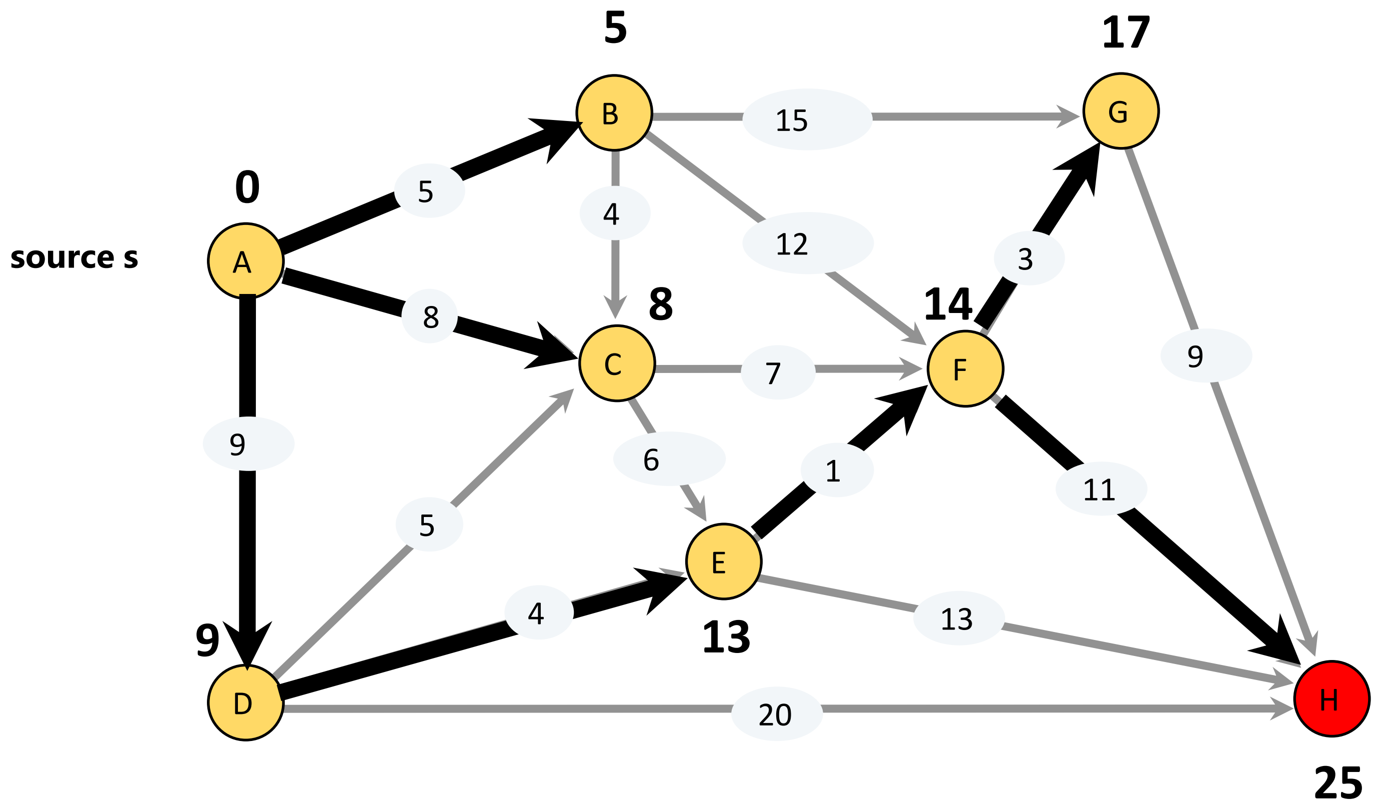
Priority Queue

H	**25**

predecessor[ ]

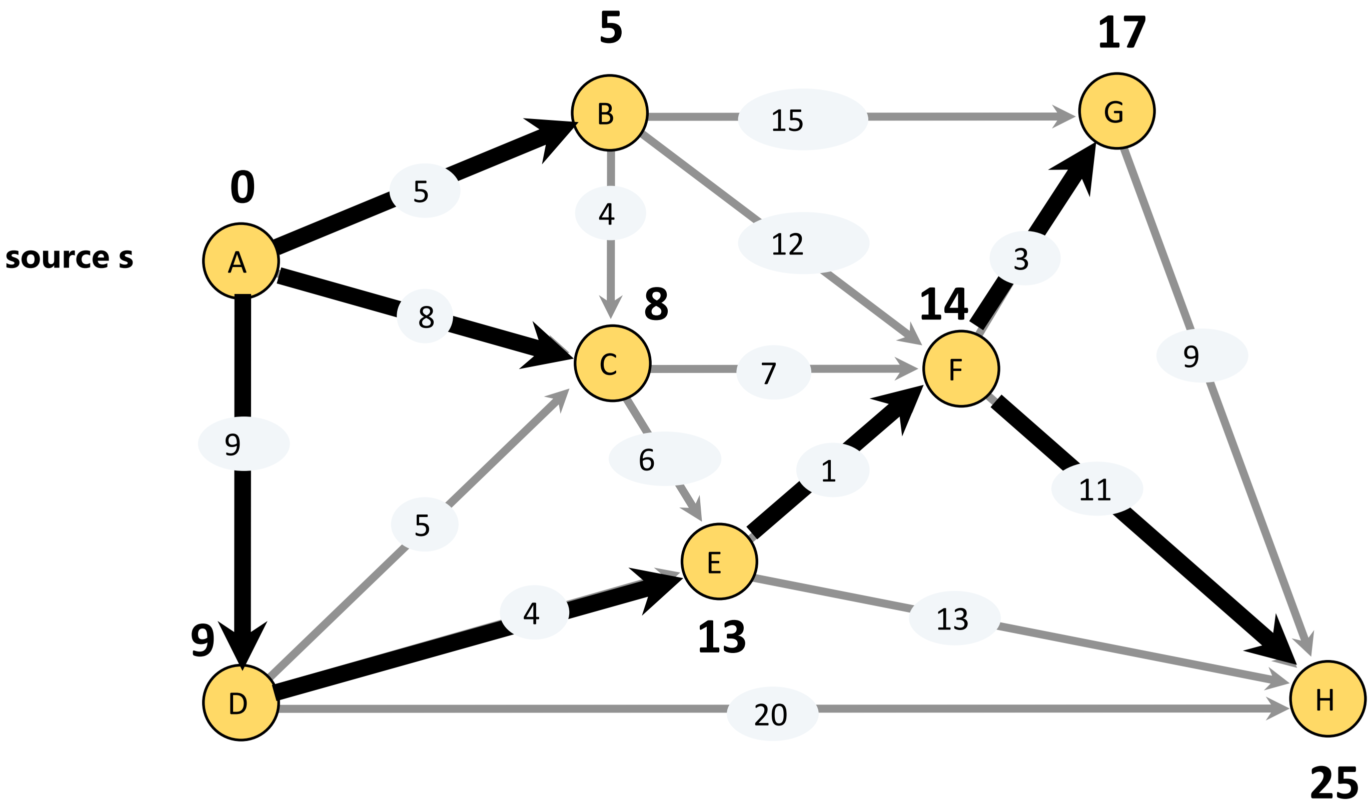
A	
B	A
C	A
D	A
E	D
F	E
G	F
H	F

Single-source shortest paths problem



	distance[ ]	Priority Queue		predecessor[ ]
A	**0**			
B	5			A
C	8			A
D	9			A
E	13			D
F	14			E
G	17			F
H	25			F

Single-source shortest paths problem



	distance[ ]	Priority Queue	predecessor[ ]
A	**0**		
B	5		A
C	8		A
D	9		A
E	13		D
F	14		E
G	17		F
H	25		F

Dijkstra's algorithm (for single-source shortest paths problem)

```
for each vertex v in graph:
        distance[v] = ∞
        predecessor[v] = undefined
```

$O(V)$

```
distance[s] = 0

PQ = empty priority queue
for each vertex v in graph:
        PQ.insert(v, distance[v])
```

$O(V)$

$2O(V) + O(V)\,(\mathrm{O}(\log V)$ +out-degree(u) $\mathrm{O}(\log V))$

$2O(V) + O\,(\mathrm{V}\log V) + O(V)$(out-degree(u)) $\mathrm{O}(\log V))$

$2O(V) + O\,(\mathrm{V}\log V) + O(E)\,\mathrm{O}(\log V))$

$\boldsymbol{2O(V) + O(V+E)\,(\log V))}$

```
while PQ is not empty:
        u = PQ.extract-min()
        for each out-degree(u,v) of u:
            alt = distance[u] + weight(u, v)
            if alt < distance[v]:
                distance[v] = alt
                predecessor[v] = u
                PQ.decrease-key(v, alt)
```

$\mathbf{O}(\boldsymbol{\log V})$

$\mathbf{O}(\mathbf{log}\,\boldsymbol{V})$

out-degree(u)

$\boldsymbol{O(V)}$

This lecture Dedicated to…

“What is the shortest way to travel from Rotterdam to Groningen? It is the algorithm for the shortest path which **I designed in about 20 minutes**. One morning I was shopping with my young fiancée, and tired, we sat down on the cafe terrace to drink a cup of coffee and I was just thinking about whether I could do this, and then designed the algorithm for the shortest path.”

Edsger W. Dijkstra
Turing award 1972

This lecture Dedicated to…

" Do only what only you can do. "

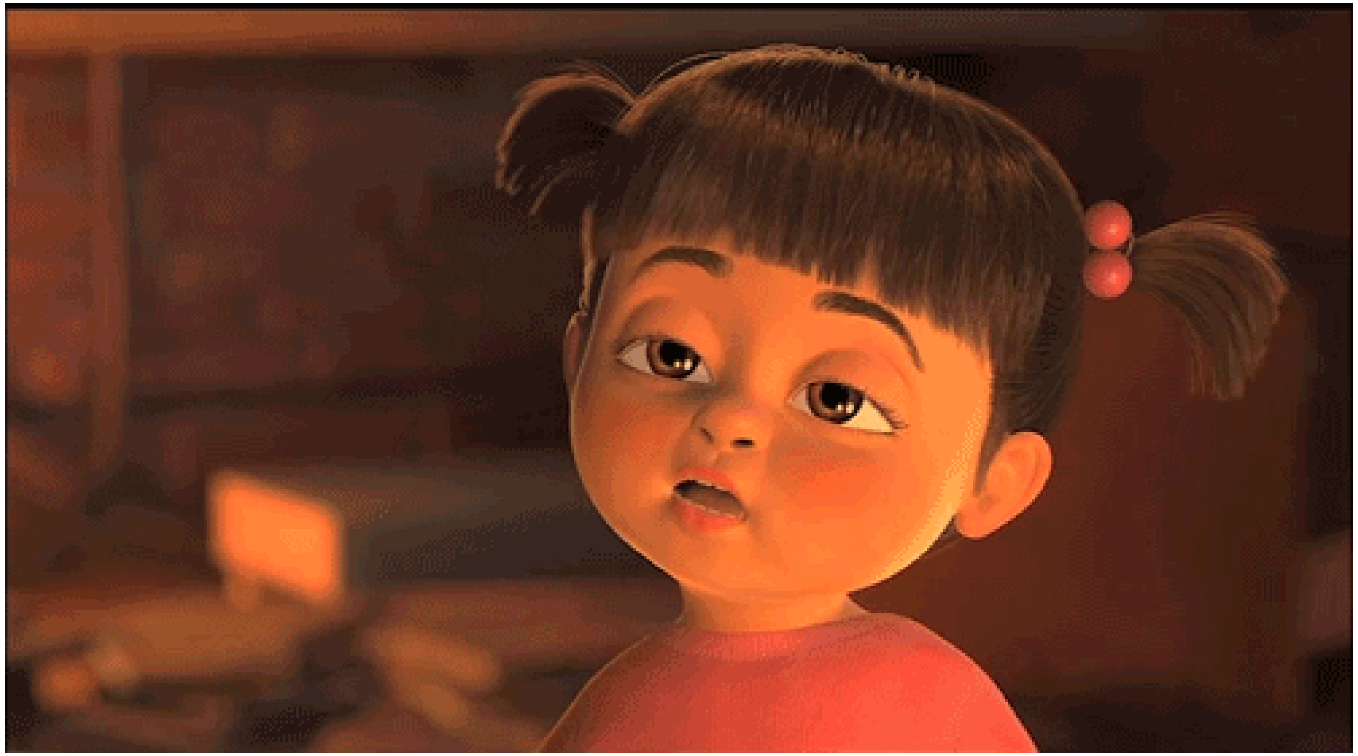
" The use of COBOL cripples the mind; its teaching should, therefore, be regarded as a criminal offence. "

" It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration. "

" APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums. "

Edsger W. Dijkstra
Turing award 1972

Thanks a lot



If you are taking a Nap, **wake up**........Lecture Over