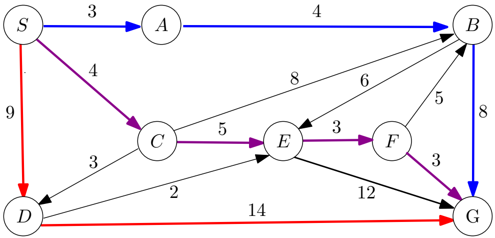
Single Source Shortest Path

- Weighted Graphs and Shortest Paths
- Dijkstra Algorithm
- Proof of Correctness
- Runtime
    - Basic Implementation
    - Vertex-Centric Implementation
    - Heap Based Implementation

Imdad ullah Khan

Shortest Paths

Weight of a path in weighted graphs is sum of weights of its edges



Three $S - G$ paths

$\mathbf{C(p_1) = 3 + 4 + 8}$

$\mathbf{C(p_2) = 4 + 5 + 3 + 3}$

$\mathbf{C(p_3) = 9 + 14}$

Shortest path from s to t is a path of smallest weight

Distance from s to t, $\mathbf{d(s,t)}$: weight of the shortest $s - t$ path

There can be multiple shortest paths

Shortest Path Problems

1. Shortest $s - t$ path:

   Given $G = (V, E, w)$ and $s, t \in V$, find a shortest path from s to t

   - For an undirected graph, it will be a path between s and t
   - Unweighted graphs are weighted graphs with all edge weights $= 1$
   - Shortest path is not unique, any path with minimum weight will work

2. Single source shortest paths (SSSP):

   Given $G = (V, E, w)$ and $s \in V$, find shortest paths from s to all $t \in V$

   - Problems of undirected and unweighted graphs are covered as above
   - It includes the first problem

We focus on SSSP

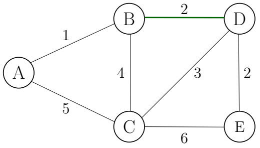
SSSP Problem

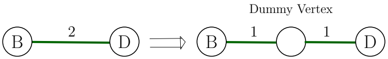
Input: A weighted graph G and a source vertex $s \in V$
Output: Shortest paths from s to all vertices $v \in V$

For unweighted graphs (unit weights) BFS from s will work

▷ BFS running time: $O(n + m)$

For weighted graph replace each edge e by a directed path of $w(e)$ unit weight edges





- What if weights are not integers or are negative
- Blows up size of the graph a lot

Dijkstra Algorithm

Input: A weighted graph G and a source vertex $s \in V$
Output: Shortest paths from s to all vertices $v \in V$

Dijkstra's algorithm solves SSSP for both directed and undirected graphs

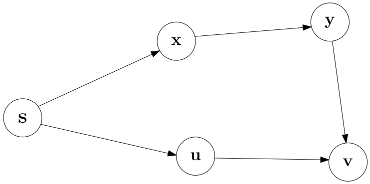
Assumptions:

1. All vertices are reachable from s
   - Otherwise there is no shortest path (distance $= \infty$)
   - Easy to get $R(s)$ in preprocessing (e.g., BFS or DFS)
2. All edge weights are non-negative
   - Bellman-Ford algorithm deals with negative weights

Dijkstra Algorithm

- First step: only find distances $d[1 \dots n]$ $\quad d[i] = d(s, v_i)$
- $d[s] = 0$
- Maintains a set $R \subset V$ (known region), $\quad d[x \in R]$ is finalized
- Initially $R = \{s\}$ and iteratively add one vertex to R

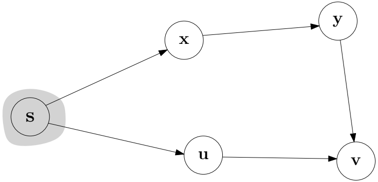
$d[1 \dots n] \leftarrow [\infty \dots \infty]$
$d[s] \leftarrow 0$
$R \leftarrow \{s\}$
while $R \neq V$ **do**
    Select $v \in \overline{R}$
    $R \leftarrow R \cup \{v\}$
    $d[v] \leftarrow d(s, v)$



Dijkstra Algorithm

- First step: only find distances $d[1 \dots n]$ $\quad d[i] = d(s, v_i)$
- $d[s] = 0$
- Maintains a set $R \subset V$ (known region), $\quad d[x \in R]$ is finalized
- Initially $R = \{s\}$ and iteratively add one vertex to R

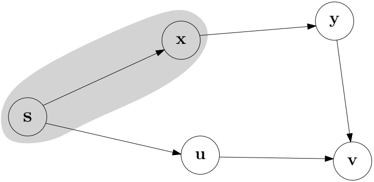
$d[1 \dots n] \leftarrow [\infty \dots \infty]$
$d[s] \leftarrow 0$
$R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$



Dijkstra Algorithm

- First step: only find distances $d[1 \ldots n]$ $\quad d[i] = d(s, v_i)$
- $d[s] = 0$
- Maintains a set $R \subset V$ (known region), $\quad d[x \in R]$ is finalized
- Initially $R = \{s\}$ and iteratively add one vertex to R

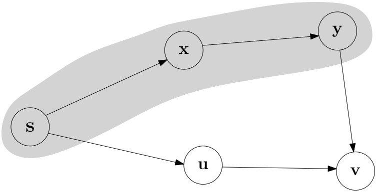
$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0$
$R \leftarrow \{s\}$
while $R \neq V$ **do**
    Select $v \in \overline{R}$
    $R \leftarrow R \cup \{v\}$
    $d[v] \leftarrow d(s, v)$



Dijkstra Algorithm

- First step: only find distances $d[1 \dots n]$ $\quad d[i] = d(s, v_i)$
- $d[s] = 0$
- Maintains a set $R \subset V$ (known region), $\quad d[x \in R]$ is finalized
- Initially $R = \{s\}$ and iteratively add one vertex to R

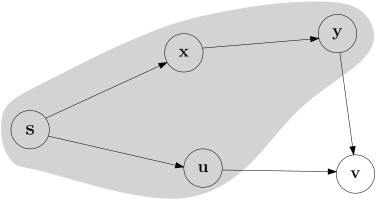
$d[1 \dots n] \leftarrow [\infty \dots \infty]$
$d[s] \leftarrow 0$
$R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$



Dijkstra Algorithm

- First step: only find distances $d[1 \dots n]$ $\quad d[i] = d(s, v_i)$
- $d[s] = 0$
- Maintains a set $R \subset V$ (known region), $\quad d[x \in R]$ is finalized
- Initially $R = \{s\}$ and iteratively add one vertex to R

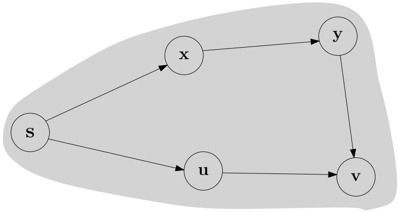
$d[1 \dots n] \leftarrow [\infty \dots \infty]$
$d[s] \leftarrow 0$
$R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$



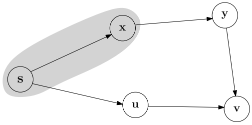
Dijkstra Algorithm

- First step: only find distances $d[1 \ldots n]$ $\quad d[i] = d(s, v_i)$
- $d[s] = 0$
- Maintains a set $R \subset V$ (known region), $\quad d[x \in R]$ is finalized
- Initially $R = \{s\}$ and iteratively add one vertex to R

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0$
$R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$

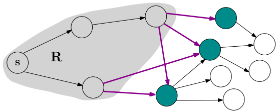


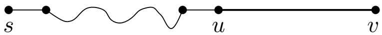
Dijkstra Algorithm: Greedy Criteria



$d[1\ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0 \quad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$

- Which vertex from $\overline{R}$ to add to R?
- The vertex $v \in \overline{R}$ that is closest to s
- Such a v must be at the **"frontier"** of $\overline{R}$

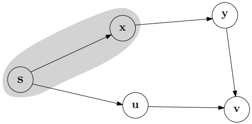




Shortest path to $v \in \overline{R}$, closest to s

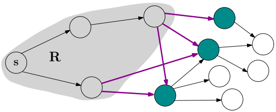
- Let $v \in \overline{R}$ be the closest to s and let a shortest $s - v$ path be $s, \ldots, u, v$
-
-
-

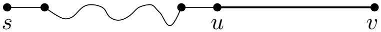
Dijkstra Algorithm: Greedy Criteria



$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0 \quad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s,v)$

- Which vertex from $\overline{R}$ to add to R?
- The vertex $v \in \overline{R}$ that is closest to s
- Such a v must be at the **"frontier"** of $\overline{R}$

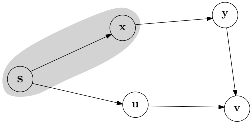




Shortest path to $v \in \overline{R}$, closest to s

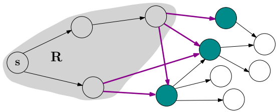
- Let $v \in \overline{R}$ be the closest to s and let a shortest $s - v$ path be $s, \ldots, u, v$
- $w(uv) \geq 0 \implies d(s,u) \leq d(s,v) \implies u$ is closer to s than $v \implies u \in R$
-
-

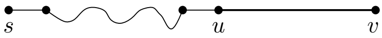
Dijkstra Algorithm: Greedy Criteria



$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0 \quad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$

- Which vertex from $\overline{R}$ to add to R?
- The vertex $v \in \overline{R}$ that is closest to s
- Such a v must be at the **"frontier"** of $\overline{R}$

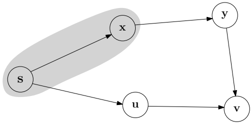




Shortest path to $v \in \overline{R}$, closest to s

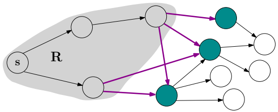
- Let $v \in \overline{R}$ be the closest to s and let a shortest $s - v$ path be $s, \ldots, u, v$
- $w(uv) \geq 0 \implies d(s,u) \leq d(s,v) \implies u$ is closer to s than $v \implies u \in R$
- Otherwise we get contradiction to v being closest to s in $\overline{R}$
-

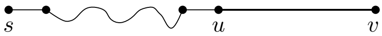
Dijkstra Algorithm: Greedy Criteria



$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0 \quad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$

- Which vertex from $\overline{R}$ to add to R?
- The vertex $v \in \overline{R}$ that is closest to s
- Such a v must be at the **"frontier"** of $\overline{R}$

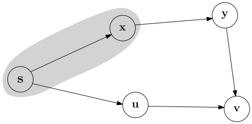




Shortest path to $v \in \overline{R}$, closest to s

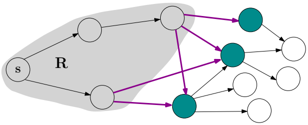
- Let $v \in \overline{R}$ be the closest to s and let a shortest $s - v$ path be $s, \ldots, u, v$
- $w(uv) \geq 0 \implies d(s,u) \leq d(s,v) \implies u$ is closer to s than $v \implies u \in R$
- Otherwise we get contradiction to v being closest to s in $\overline{R}$
- This implies that v is only one edge away from R, i.e. (u, v)

Dijkstra Algorithm: Greedy Criteria



$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$d[s] \leftarrow 0 \quad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $v \in \overline{R}$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d(s, v)$

- Which vertex from $\overline{R}$ to add to R?
- The vertex $v \in \overline{R}$ that is closest to s
- Such a v must be at the **"frontier"** of $\overline{R}$



Restrict search to **"single edge extensions"** of paths to $u \in R$

- Dijkstra assigns a score to each crossing edge

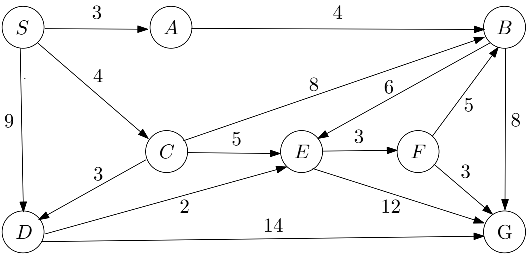
$$score(u, v) \;=\; d[u] + w(uv) \quad \text{for} \quad (u, v) \in E, u \in R, v \notin R$$

- Add a frontier vertex adjacent through minimum scoring edge

Dijkstra Algorithm

Algorithm Dijkstra's Algorithm for distances from s to all vertices

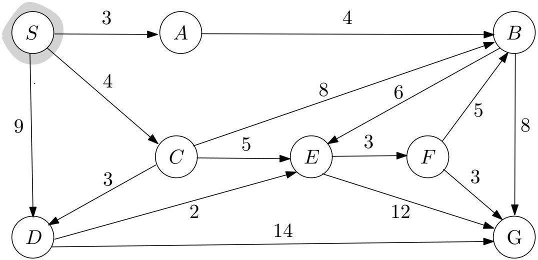
$d[1 \dots n] \leftarrow [\infty \dots \infty]$
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d[u] + w(uv)$



	d
S	∞
A	∞
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞

Algorithm Dijkstra's Algorithm for distances from s to all vertices

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



	d
S	0
A	∞
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞

Dijkstra Algorithm

Algorithm Dijkstra's Algorithm for distances from s to all vertices

$d[1 \dots n] \leftarrow [\infty \dots \infty]$

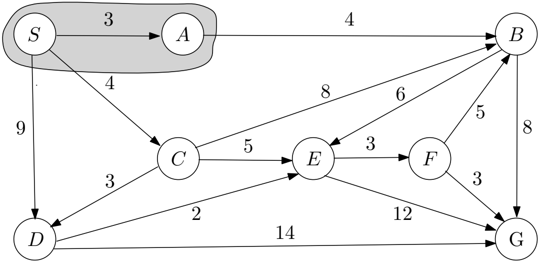
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$

while $R \neq V$ **do**

  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$

  $R \leftarrow R \cup \{v\}$

  $d[v] \leftarrow d[u] + w(uv)$



	d
S	0
A	3
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞

Dijkstra Algorithm

Algorithm Dijkstra's Algorithm for distances from s to all vertices

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$

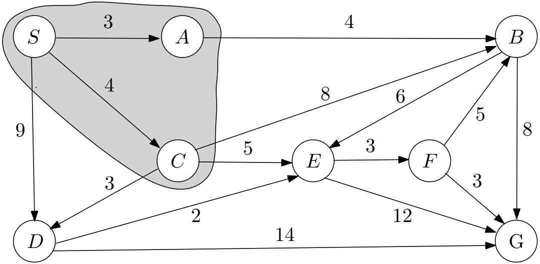
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$

while $R \neq V$ **do**

  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$

  $R \leftarrow R \cup \{v\}$

  $d[v] \leftarrow d[u] + w(uv)$



	d
S	0
A	3
B	∞
C	4
D	∞
E	∞
F	∞
G	∞

Algorithm Dijkstra's Algorithm for distances from s to all vertices

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$

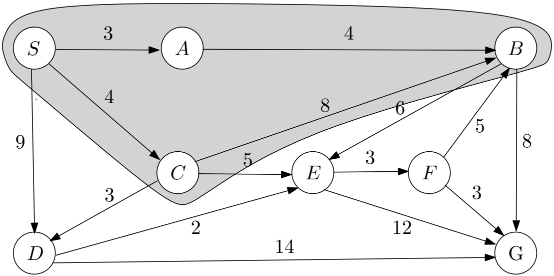
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$

while $R \neq V$ **do**

  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$

  $R \leftarrow R \cup \{v\}$

  $d[v] \leftarrow d[u] + w(uv)$



	d
S	0
A	3
B	7
C	4
D	∞
E	∞
F	∞
G	∞

Dijkstra Algorithm

Algorithm Dijkstra's Algorithm for distances from s to all vertices

$d[1 \dots n] \leftarrow [\infty \dots \infty]$

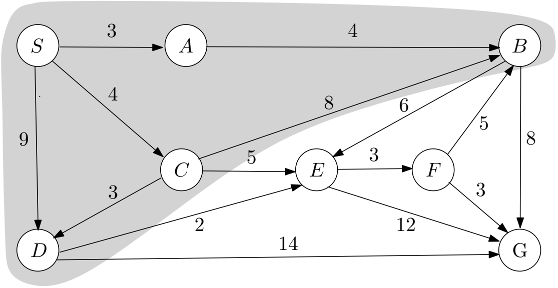
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$

while $R \neq V$ **do**

  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$

  $R \leftarrow R \cup \{v\}$

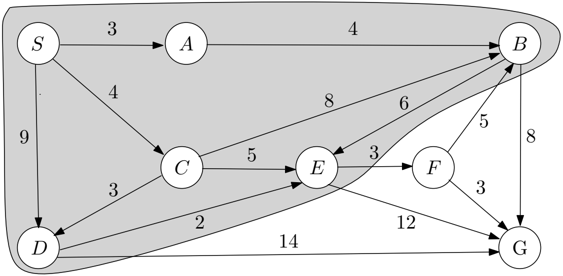
  $d[v] \leftarrow d[u] + w(uv)$



	d
S	0
A	3
B	7
C	4
D	7
E	∞
F	∞
G	∞

Algorithm Dijkstra's Algorithm for distances from s to all vertices

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

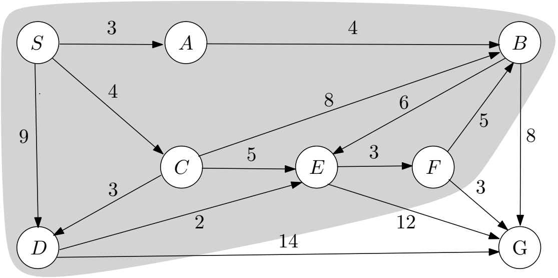


	d
S	0
A	3
B	7
C	4
D	9
E	9
F	∞
G	∞

Dijkstra Algorithm

Algorithm Dijkstra's Algorithm for distances from s to all vertices

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



	d
S	0
A	3
B	7
C	4
D	9
E	9
F	12
G	∞

Algorithm Dijkstra's Algorithm for distances from s to all vertices

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$

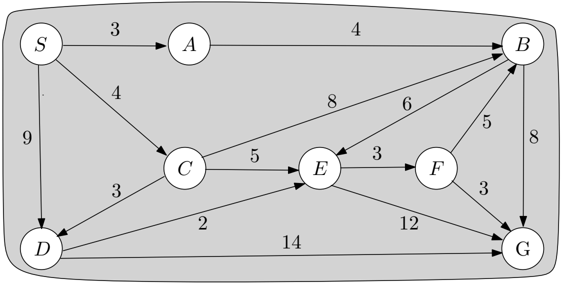
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$

while $R \neq V$ **do**

  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$

  $R \leftarrow R \cup \{v\}$

  $d[v] \leftarrow d[u] + w(uv)$



	d
S	0
A	3
B	7
C	4
D	9
E	9
F	12
G	15

Dijkstra Algorithm with paths

Record predecessor relationships (sources of used edges)

Implicitly builds a tree (shortest path tree)

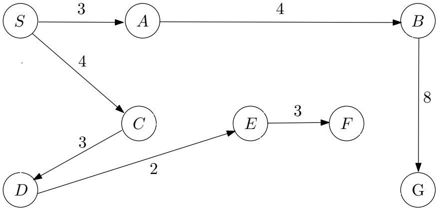
Algorithm Dijkstra's Algorithm for Shortest Paths from s to all vertices

$d[1 \ldots n] \leftarrow [\infty \ldots \infty]$
$prev[1 \ldots n] \leftarrow [null \ \ldots \ null]$
$d[s] \leftarrow 0$
$R \leftarrow \{s\}$
while $R \neq V$ **do**
 Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$
 $R \leftarrow R \cup \{v\}$
 $d[v] \leftarrow d[u] + w(uv)$
 $prev[v] \leftarrow u$ ▷ predecessor is the vertex whose path is single-edge extended

Dijkstra Algorithm with paths

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

$d[1\ldots n] \leftarrow [\infty \ldots \infty]$
$prev[1\ldots n] \leftarrow [null\ \ldots\ null]$
$d[s] \leftarrow 0 \qquad R \leftarrow \{s\}$
while $R \neq V$ **do**
  Select $e = (u, v)$, $u \in R, v \notin R$, with minimum $d[u] + w(uv)$
  $R \leftarrow R \cup \{v\}$
  $d[v] \leftarrow d[u] + w(uv)$
  $prev[v] \leftarrow u$ ▷ predecessor is the vertex whose path is single-edge extended



	d	$prev$
S	0	Nil
A	3	S
B	7	A
C	4	S
D	9	C
E	9	D
F	12	E
G	15	B