Graph Theory

Shortest Path in Weighted Graphs

Sergio Peignier sergio.peignier@insa-lyon.fr

Associate Professor
INSA Lyon
Biosciences department

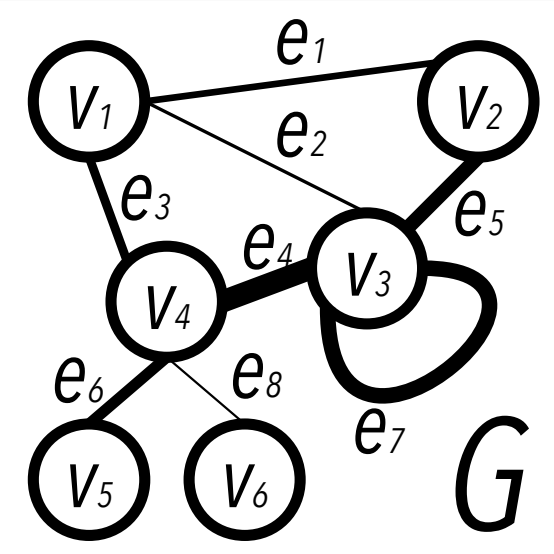
Table of contents

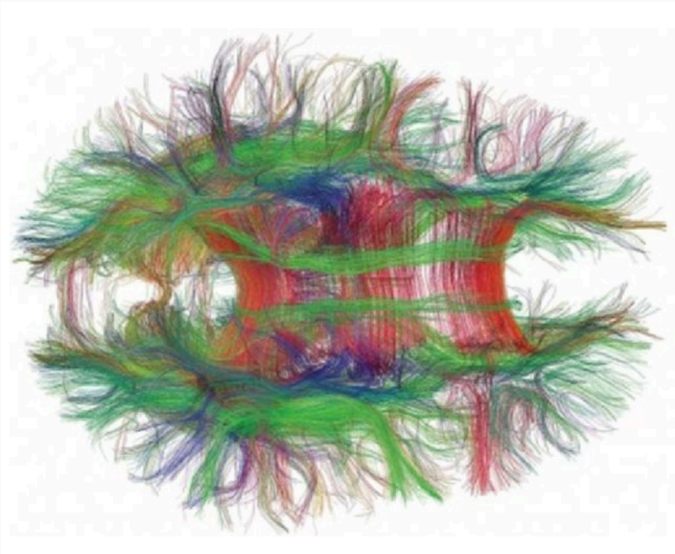
Shortest Weighted Path

Weighted Graphs

$$G = \langle V, E, w \rangle, \quad w : \langle v_i, v_j \rangle \to \mathbb{R}$$

- **Interaction strength** assigned by the **edge weight**.
- e.g., Connectome [McCoss et al.].
  nodes: neurons, edges: synapses

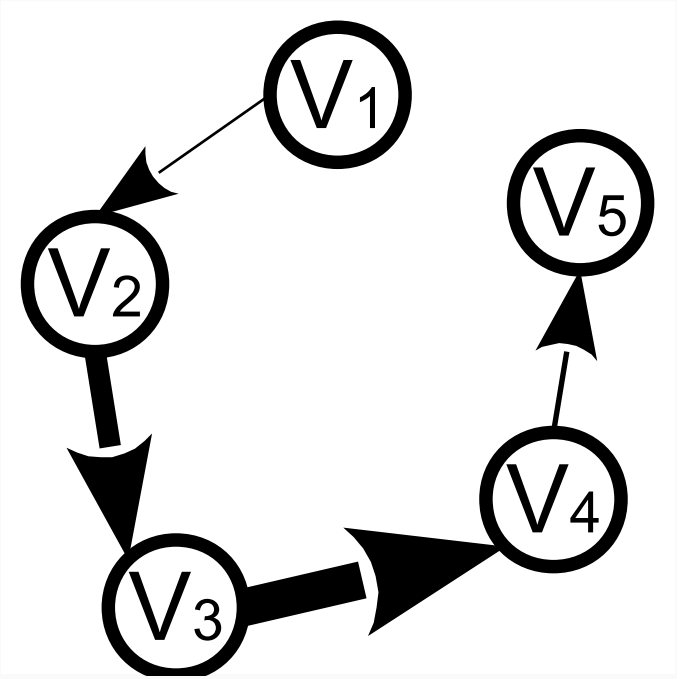




Weight of a Path

Path: $p = (v_1, e_1, v_2, e_2, v_3, \ldots, v_{k-1}, e_{k-1}, v_k)$

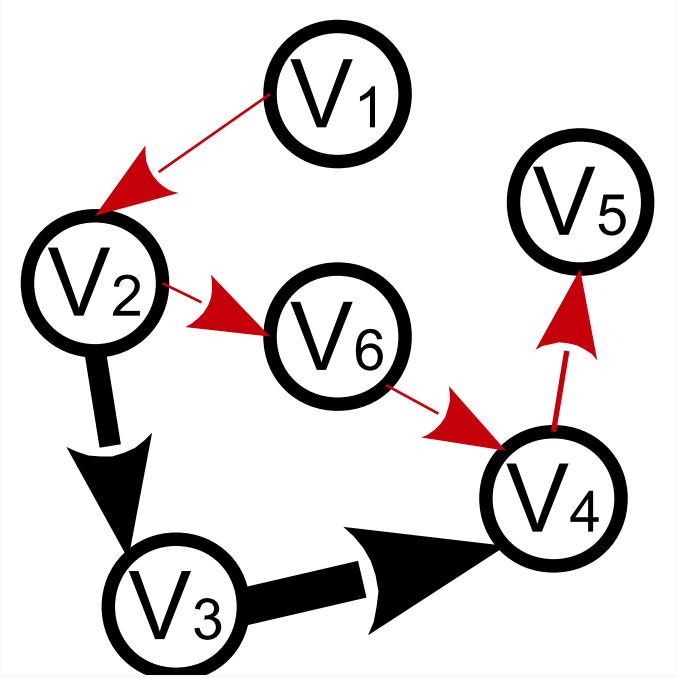
$$w(p) = \sum_{i=1}^{k-1} w(e_i)$$



Shortest Path

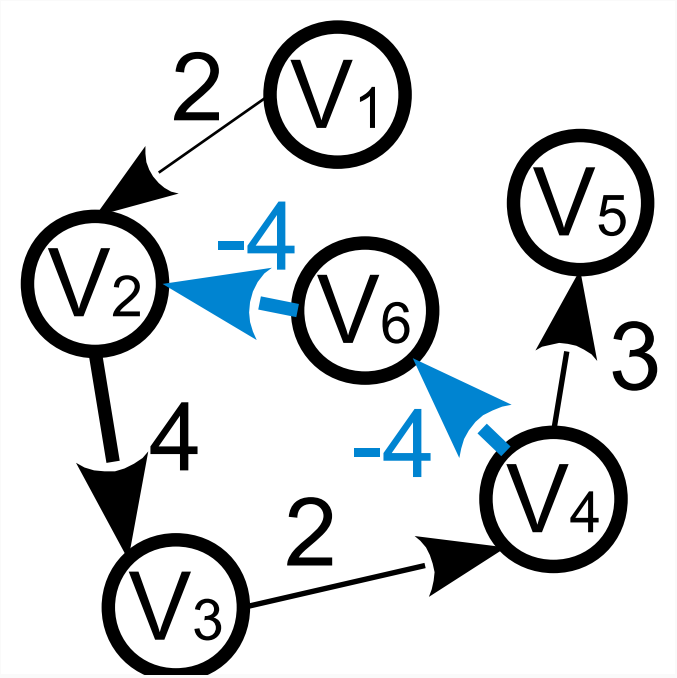
Definition: Path of minimum weight from u to v

$$\boxed{dist(u, v) = min\{w(p) \quad \text{s.t.} \quad p \text{ is a path from } u \text{ to } v\}}$$



Negative Edges

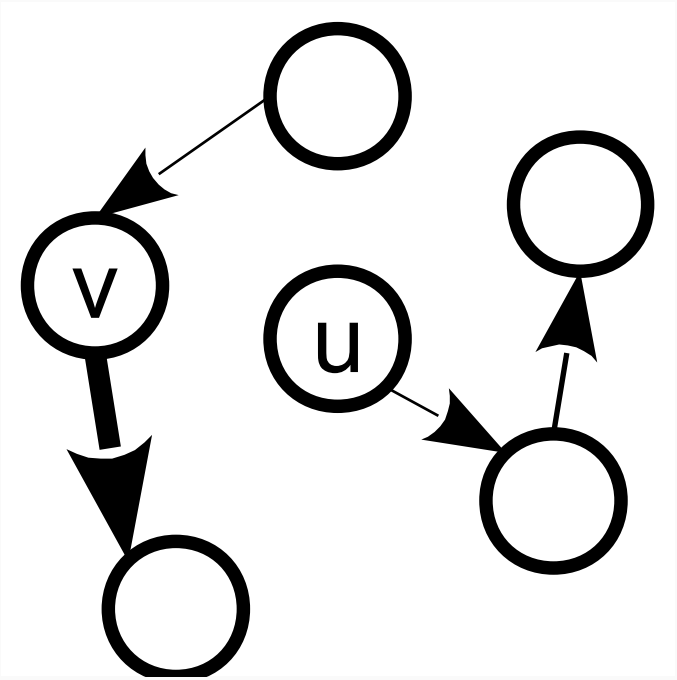
Shortest path does **not** exist when *G* has **negative edges**.



Not connected graph

Shortest path between nodes u and v from two **distinct connected components**.

$$\boxed{dist(u, v)} = \infty$$



Structural Property

Any **subpath** of a **shortest path** is a **shortest path**

Proof?

Structural Property | Proof

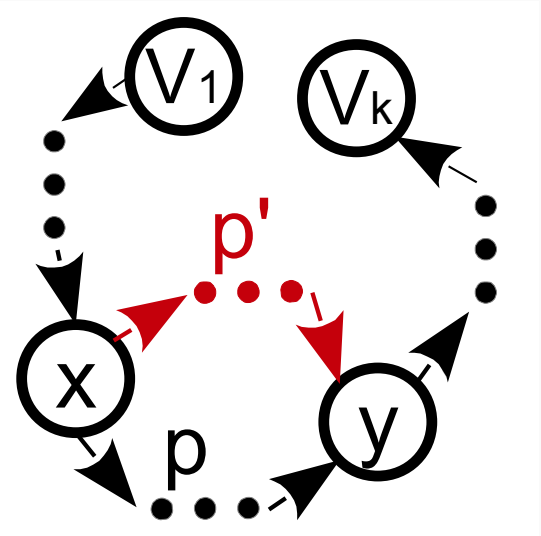
$q = (v_1, \ldots, v_k)$: shortest path between v_1 and v_k.

x and y: 2 **nodes visited** in q.

p: **subpath** of q between x and y.

Suppose $\exists$ path p' s.t. $w(p') < w(p)$.

Then q' s.t. p replaced by p' in q, would be the **shortest path** between v_1 and v_k $(w(q') < w(q)) \rightarrow$ **absurd**.



Shortest Path | Algorithm

Let $w : \langle v_i, v_j \rangle \rightarrow \mathbb{N}$

Questions: How could we use one of the algorithms studied during the previous lecture to find the shortest path (without modifying the algorithm)?

Why could this solution be impractical?

Dijkstra Algorithm

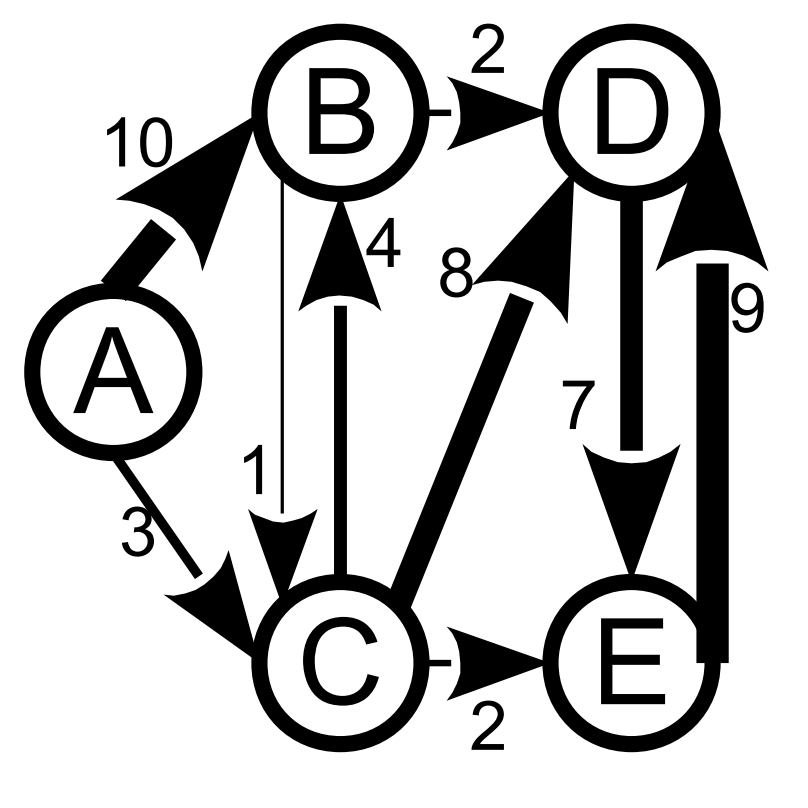
Intuitions

- Keep track of **set of visited nodes** S
- Start at an **arbitrary node** $s \in V$
- Extend S **incrementally**
  - **Current node:** v
  - For each **unvisited neighbors** of v:
    compute the **tentative dist. through** v.
  - If **dist. through** $v <$ **previous dist.**: **update dist.**
  - When **every neighbor** has been **visited**, **add** v to S.
  - $v \leftarrow$ **unvisited node** with **smallest tentative dist.**

Pseudocode

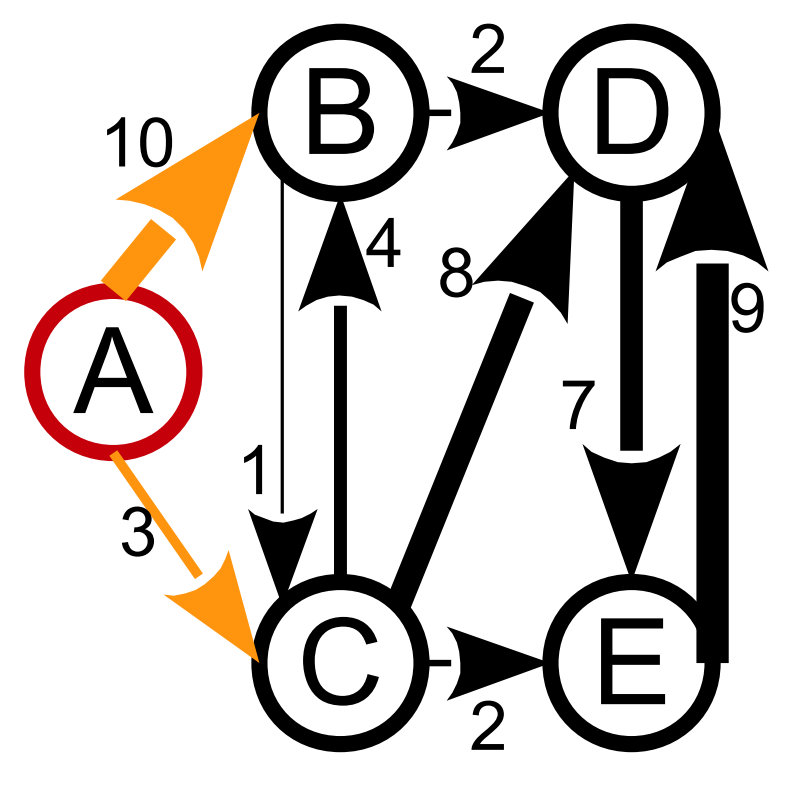
function DIJKSTRA($G = \langle V, E, w \rangle$, $s \in V$)
  for all $v \in V$ **do**
    $dist[v] \leftarrow \infty \quad parent[v] \leftarrow \emptyset$
  end for
  $dist[s] \leftarrow 0, \quad parent[s] \leftarrow root$
  $Q \leftarrow V, \quad S \leftarrow \emptyset$
  while $Size(Q) > 0$ **do**
    $u \leftarrow argmin_u(dist[u])$ // Node with least distance
    for all $v \in Neighbors(u)$ **do**
      $d \leftarrow dist[u] + w(u, v)$
      if $d < dist[v]$ **then**
        $dist[v] \leftarrow d, \quad parent[v] \leftarrow u$
      end if
    end for
    $Q \leftarrow Q \setminus \{u\}, \quad S \leftarrow S \cup \{u\}$
  end while
  Return $parent, dist$
end function

Example



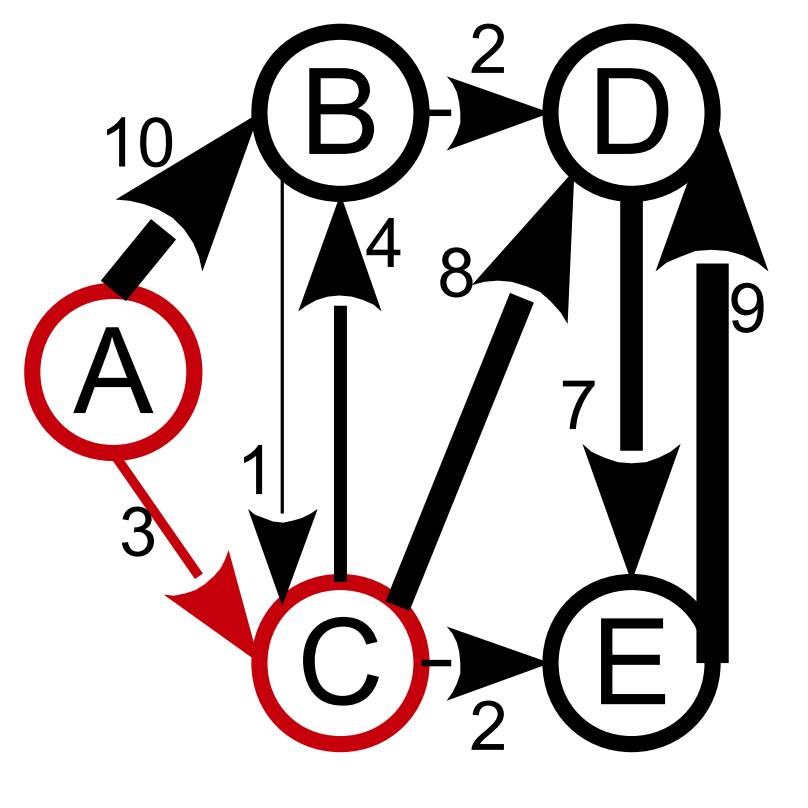
Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞

Example



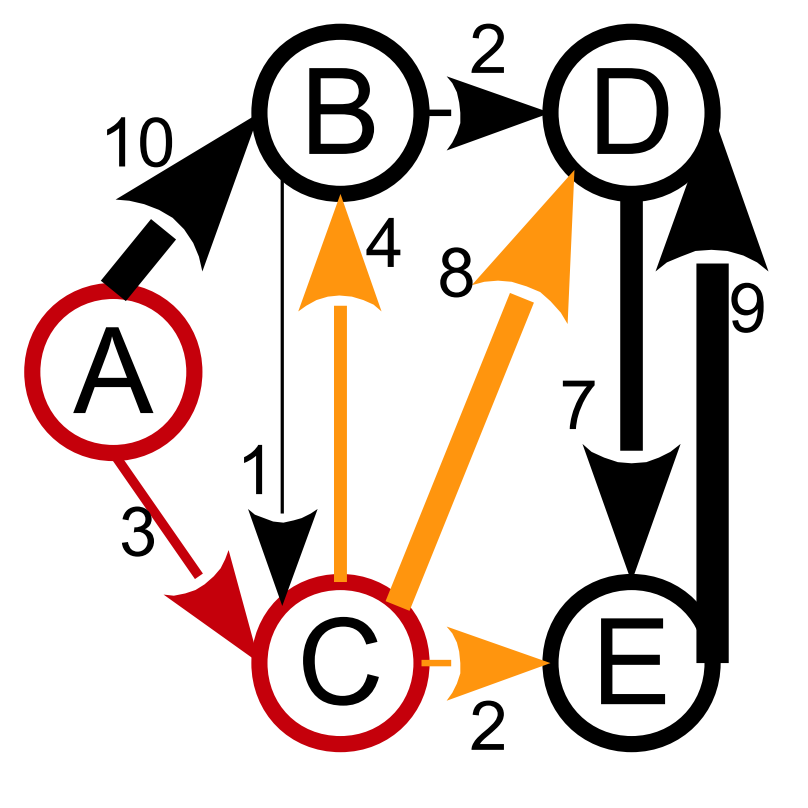
Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞
1	0	10	3	∞	∞

Example



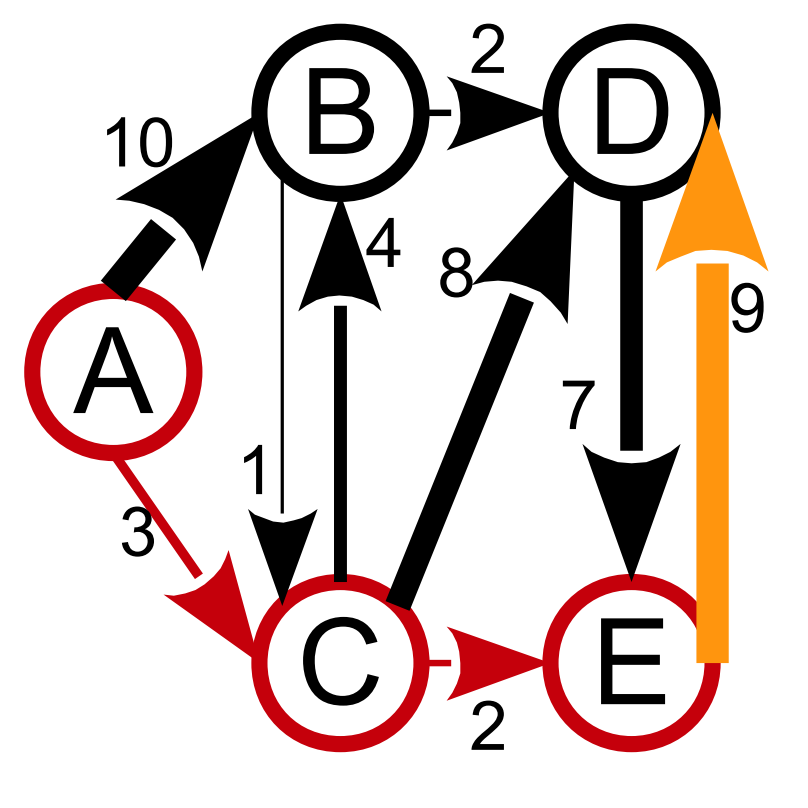
Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞
1	0	10	3	∞	∞

Example



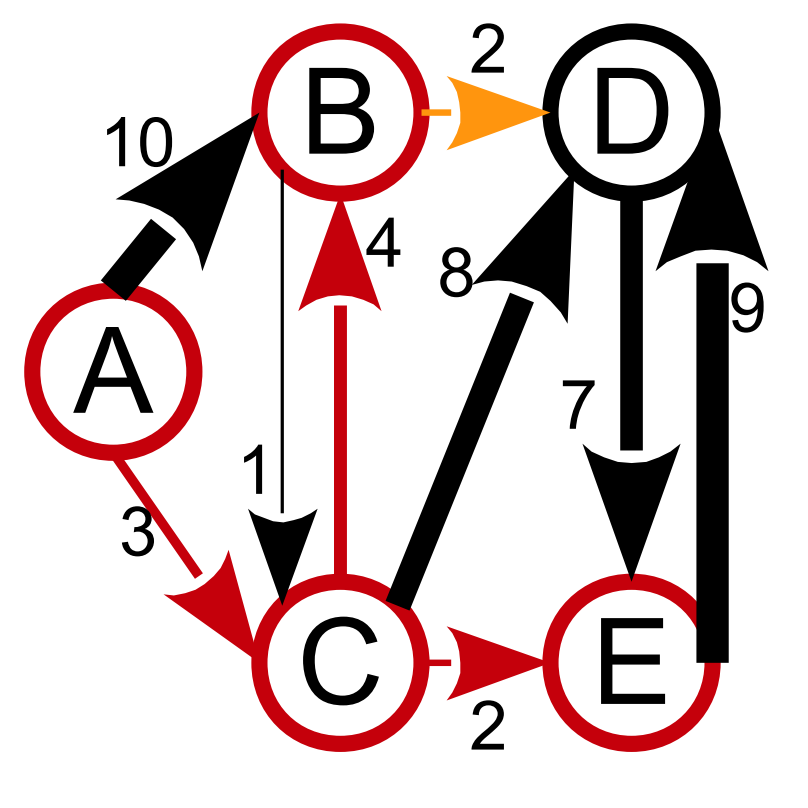
Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞
1	0	10	3	∞	∞
2	0	7	3	11	5

Example



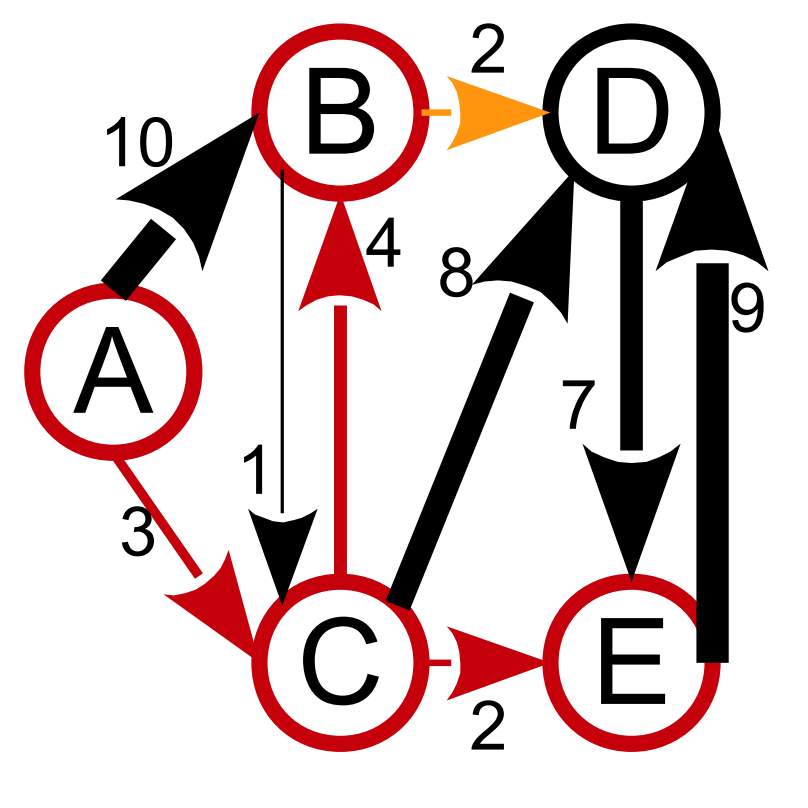
Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞
1	-	10	3	∞	∞
2	-	7	-	11	5

Example



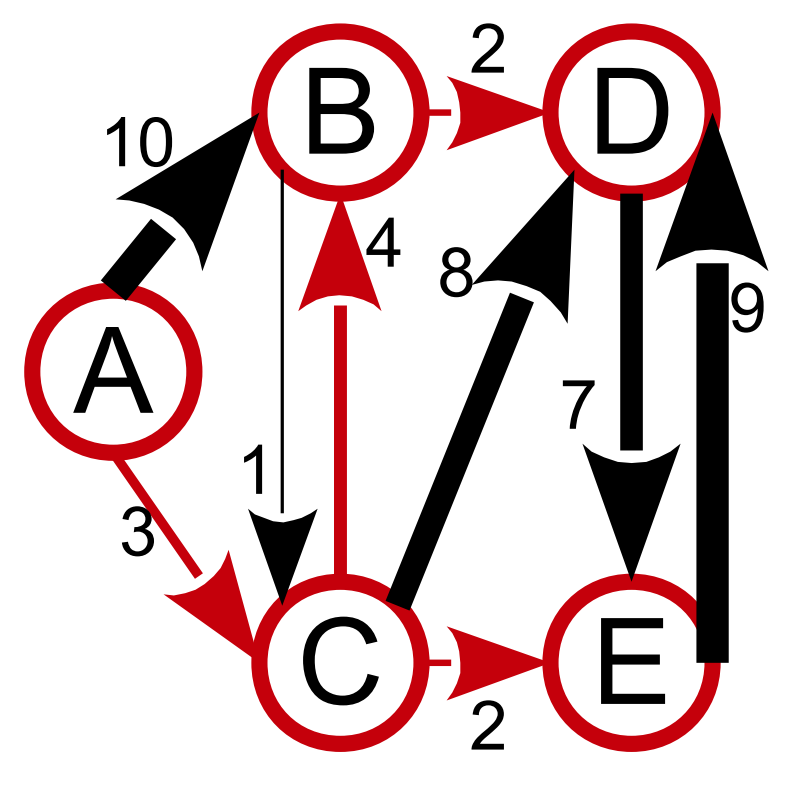
Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞
1	0	10	3	∞	∞
2	-	7	-	11	5
3	-	7	-	11	-

Example



Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞
1	0	10	3	∞	∞
2	-	7	-	11	5
3	-	7	-	9	-

Example



Step\dist	A	B	C	D	E
0	0	∞	∞	∞	∞
1	0	10	3	∞	∞
2	-	7	-	11	5
3	-	7	-	9	-
4	-	7	-	9	-

Properties

Complexity: Depends on the data structure of Q.

Best complexity: $\mathcal{O}(|E| + |V|log(|V|))$