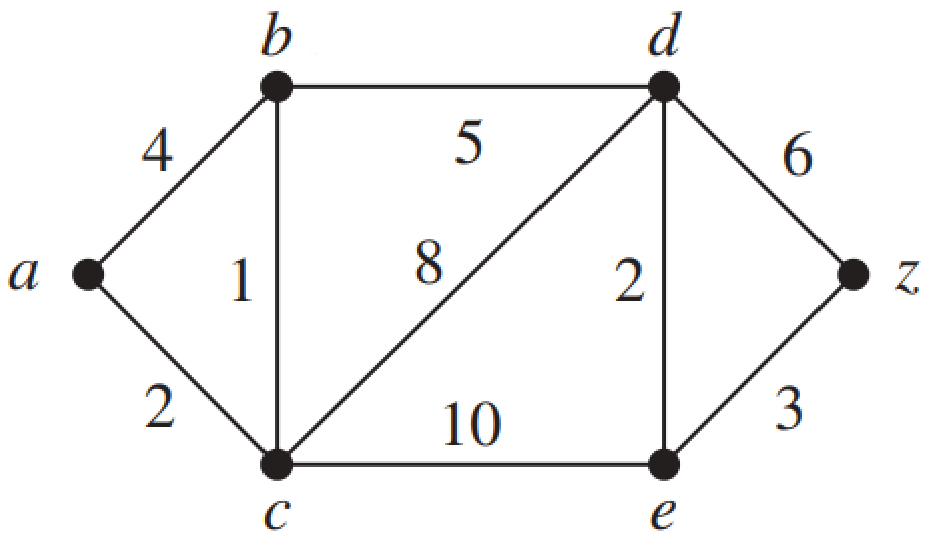
Dijkstra's Algorithm

An Bui

December 9, 2023

ALGORITHM 1 Dijkstra's Algorithm.

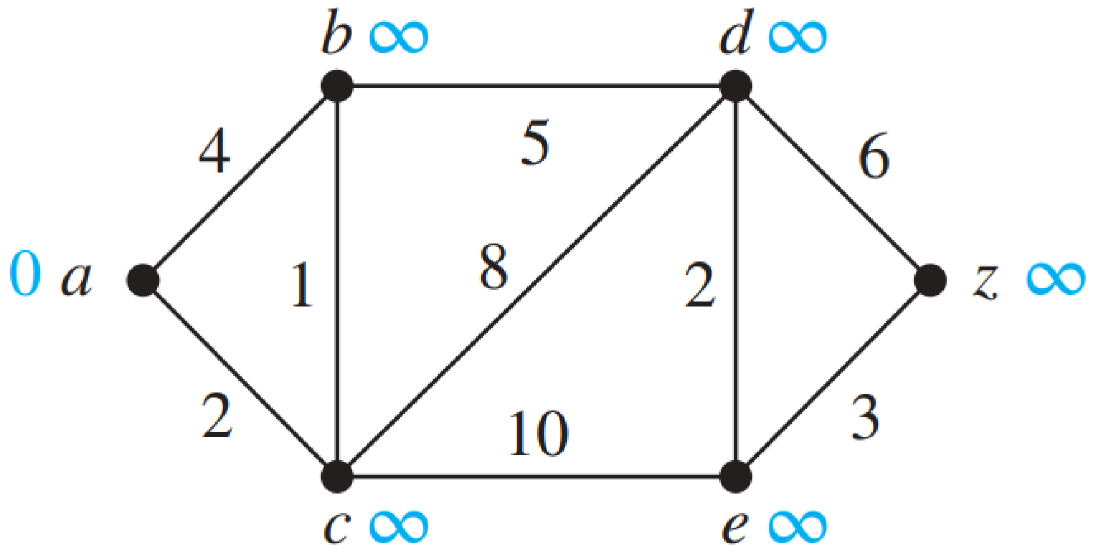
procedure *Dijkstra*(G: weighted connected simple graph, with all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$ where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
return $L(z)$ {$L(z)$ = length of a shortest path from a to z}



PURPOSE: find the shortest path in a weighted graph

ALGORITHM 1 Dijkstra's Algorithm.

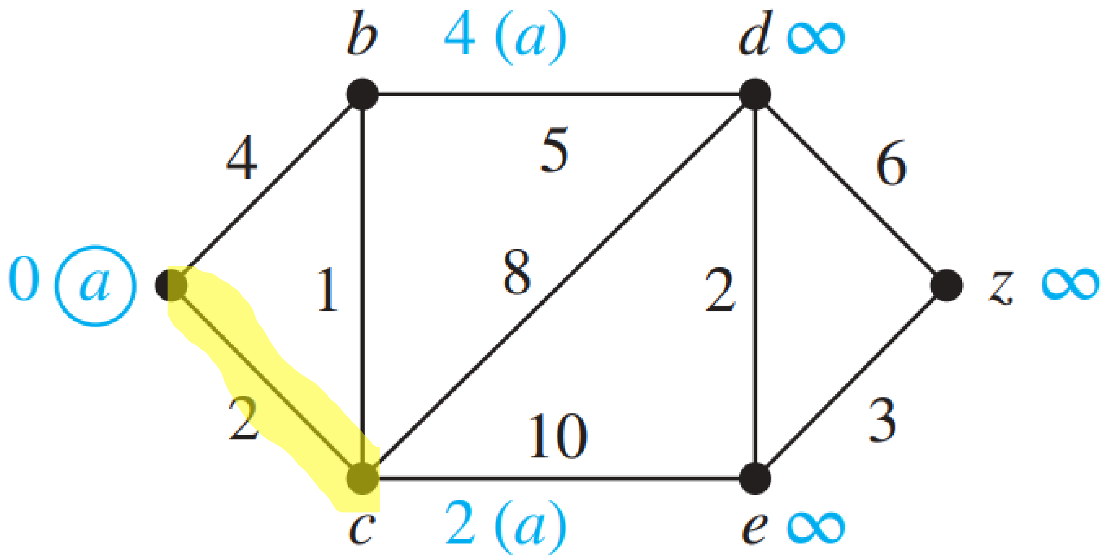
procedure *Dijkstra*(G: weighted connected simple graph, with
all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$
where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the
        labels of vertices not in S}
return $L(z)$ {$L(z)$ = length of a shortest path from a to z}



- Initialize the distance to source vertex to 0
- Initialize the distance from source to all other vertices to the max value
- Enter while loop, a is added to S

ALGORITHM 1 Dijkstra's Algorithm.

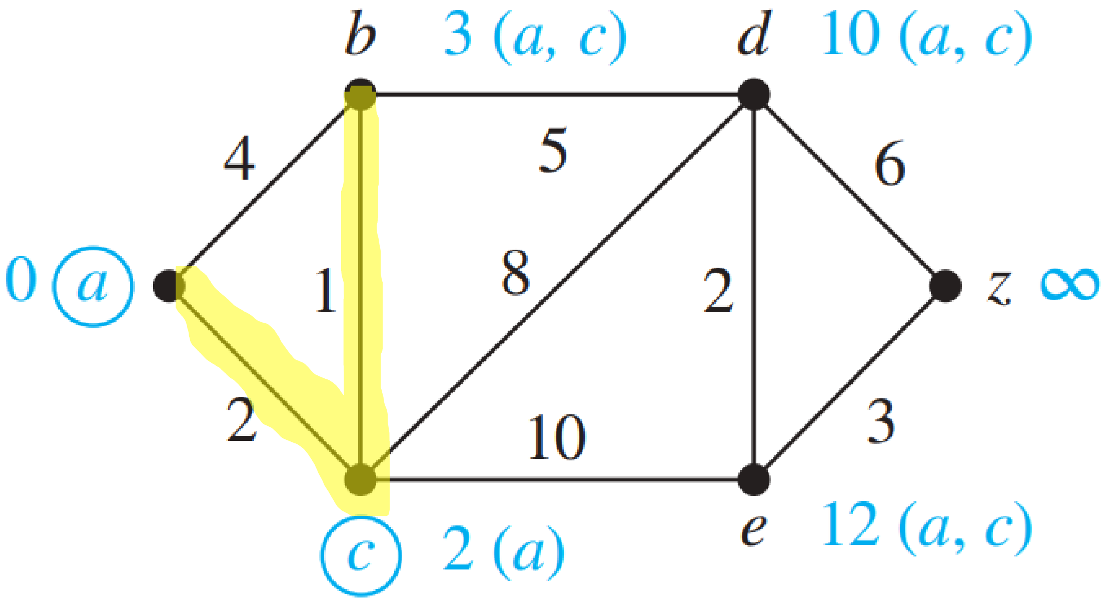
procedure *Dijkstra*(G: weighted connected simple graph, with
    all weights positive)
$\{G$ has vertices $a = v_0, v_1, \dots, v_n = z$ and lengths $w(v_i, v_j)$
    where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in $G\}$
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
    other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the
        labels of vertices not in S}
return $L(z)$ $\{L(z) =$ length of a shortest path from a to $z\}$



- **Get the path length from source vertex (a) to next adjacent vertices**
- **Adding the vertex that has smallest length from source vertex to S**
- **S now contains {a, c}**

ALGORITHM 1 Dijkstra's Algorithm.

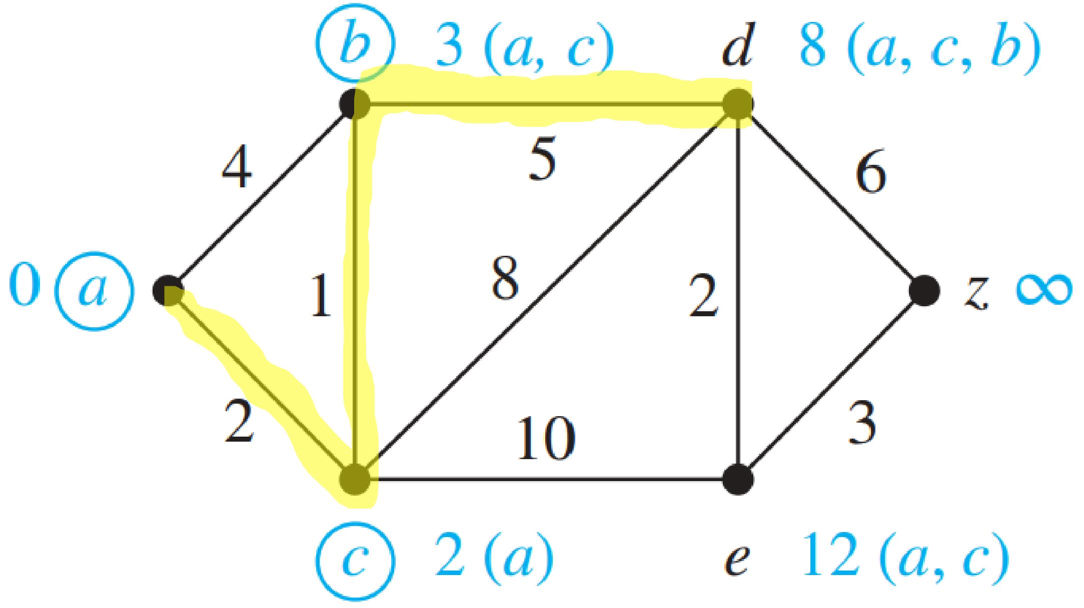
procedure *Dijkstra*(G: weighted connected simple graph, with all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$ where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
return $L(z)$ {$L(z)$ = length of a shortest path from a to z}



- **Get path length from vertices adjacent to latest vertex added to S**
- **Add the vertex that creates the smallest path length**
- **S now contains {a, c, b}**

ALGORITHM 1 Dijkstra's Algorithm.

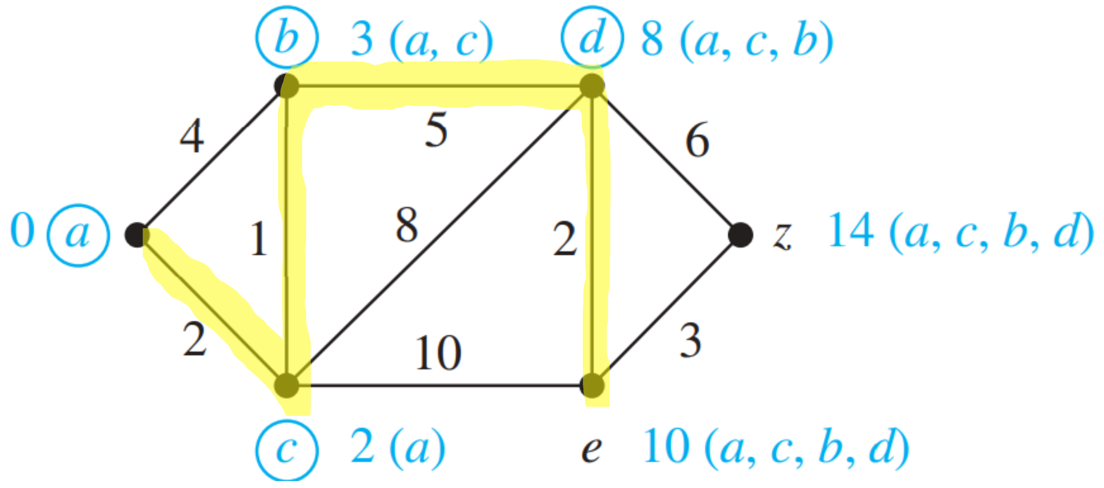
procedure *Dijkstra*(G: weighted connected simple graph, with
  all weights positive)
{G has vertices $a = v_0, v_1, \dots, v_n = z$ and lengths $w(v_i, v_j)$
  where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
  $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
  other labels are ∞, and S is the empty set}
while $z \notin S$
  $u :=$ a vertex not in S with $L(u)$ minimal
  $S := S \cup \{u\}$
  for all vertices v not in S
    if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
    {this adds a vertex to S with minimal label and updates the
    labels of vertices not in S}
return $L(z)$ {$L(z)$ = length of a shortest path from a to z}



- **Get path length from vertices adjacent to latest vertex added to S**
- **Add the vertex that creates the smallest path length**
- **S now contains {a, c, b, d}**

ALGORITHM 1 Dijkstra's Algorithm.

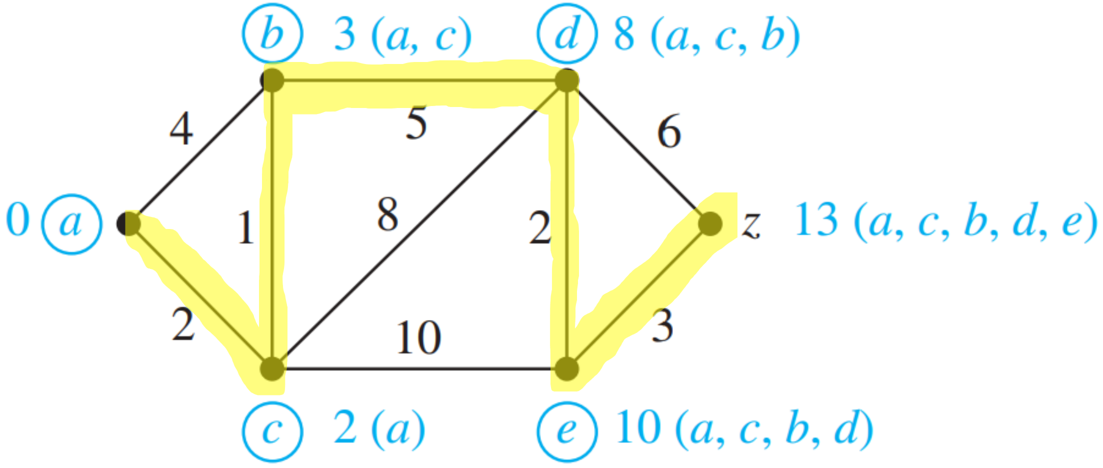
procedure *Dijkstra*(G: weighted connected simple graph, with all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$ where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
return $L(z)$ {$L(z)$ = length of a shortest path from a to z}



- **Get path length from vertices adjacent to latest vertex added to S**
- **Add the vertex that creates the smallest path length**
- **S now contains {a, c, b, d, e}**

ALGORITHM 1 Dijkstra's Algorithm.

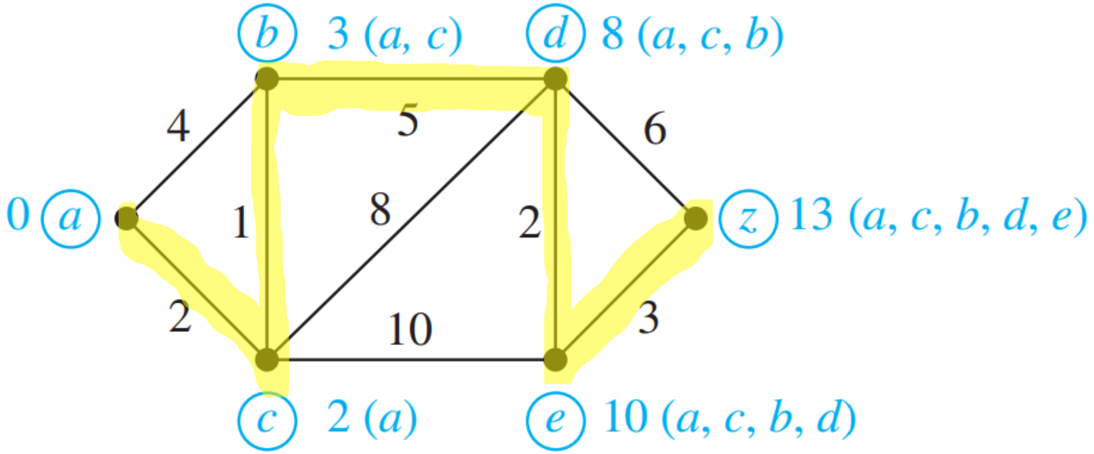
procedure *Dijkstra*(G: weighted connected simple graph, with all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$ where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
return $L(z)$ {$L(z)$ = length of a shortest path from a to z}



- **Get path length from vertices adjacent to latest vertex added to S**
- **Add the vertex that creates the smallest path length**
- **S now contains {a, c, b, d, e, z}**

ALGORITHM 1 Dijkstra's Algorithm.

procedure *Dijkstra*(G: weighted connected simple graph, with
all weights positive)
$\{G$ has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$
where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in $G\}$
for $i := 1$ **to** n
$L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
other labels are ∞, and S is the empty set}
while $z \notin S$
$u :=$ a vertex not in S with $L(u)$ minimal
$S := S \cup \{u\}$
for all vertices v not in S
if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
{this adds a vertex to S with minimal label and updates the
labels of vertices not in S}
return $L(z)$ $\{L(z) =$ length of a shortest path from a to $z\}$



- **S = {a, c, b, d, e, z}**
- **Shortest path from a to z is a, c, b, d, e, z with length 13**

This is a greedy algorithm.

Local Optimality:

- At each step, the algorithm selects the vertex with the shortest known distance among the vertices not yet included in the shortest path tree.

No Reevaluation of Choices:

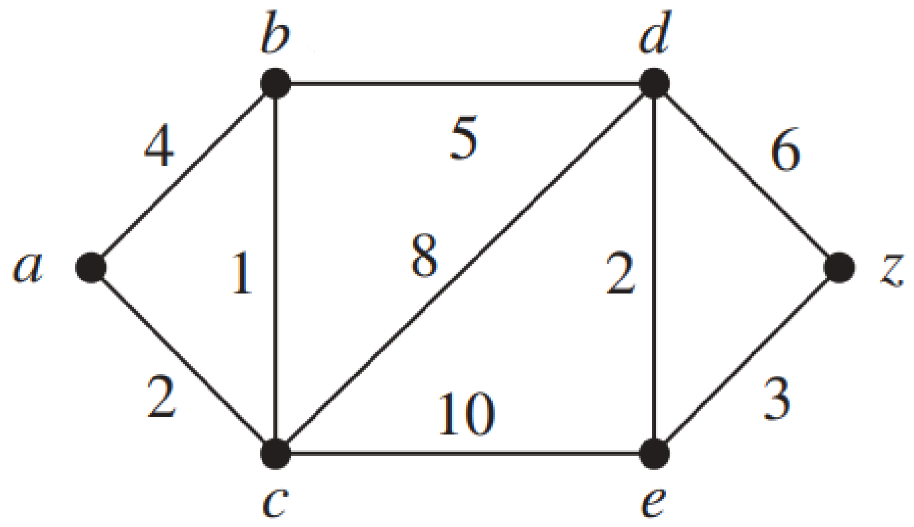
- Once a vertex is included in the shortest path tree (added to the set S), its distance label is not reconsidered. The algorithm does not revisit vertices; it moves forward, always choosing the locally optimal path.

Short-Term Choices:

- It optimizes for the shortest path from the source to each vertex individually. It does not look ahead but focuses on immediate gains at each step.

Greedy Choice Property:

- Makes globally optimal solution by selecting locally optimal choices.



Limitations/Cons

- No negative weights
- No loops or cycles
- O(n^2) operations (additions and comparisons)
  - Bad for larger graphs
- Used for static graphs
  - Cannot change graph as algorithm is running

Applications of Dijkstra's Algorithm

MANAGING DATA AND NETWORKS
- used by routers to determine the optimal path for transmitting data
- helps in load balancing and ensuring efficient data transfer
- ensures efficient data transmission and minimizes delays

NAVIGATION AND ROUTING
- Google Maps or other navigation apps
- optimize traffic flow, finding the most efficient routes for vehicles, reducing congestion and travel time

OTHER
- robots use it to find the most efficient path
- character navigation in game development
- optimizing the layout of components on a circuit board, minimizing the length of connections and reducing signal delays

References

- Rosen, Kenneth H. “10.6 Shortest Path Problems.” *Discrete Mathematics and Its Applications*, McGraw-Hill, New York, 2019, pp. 743–751.
- dipesh99kumar. “Applications of Dijkstra’s Shortest Path Algorithm.” GeeksforGeeks, GeeksforGeeks, 23 Sept. 2022, www.geeksforgeeks.org/applications-of-dijkstras-shortest-path-algorithm/.