CSI - 3105 Design & Analysis of Algorithms
Course 8

Jean-Lou De Carufel

Fall 2019

Algorithm *Dijkstra*(G, s)

for each vertex $v \in V$ **do**
    $d(v) = \infty$
end for
$d(s) = 0$
$S = \{\,\}$
$Q = V$
while $Q \neq \{\,\}$ **do**
    $u =$ vertex in Q for which $d(u)$ is minimum
    delete u from Q
    insert u into S
    for each edge (u, v) **do**
        $d(v) = \min\{d(v), d(u) + wt(u, v)\}$
    end for
end while

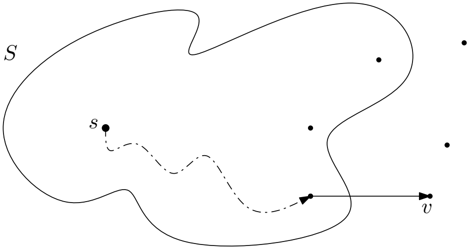
Dijkstra Algorithm is Correct

Theorem

Let $G = (V, E)$ be a weighted directed graph and $s \in V$ be a source vertex. Dijkstra algorithm finds the lengths of the shortest paths from s to all vertices in V.

Special Paths and Induction Hypotheses

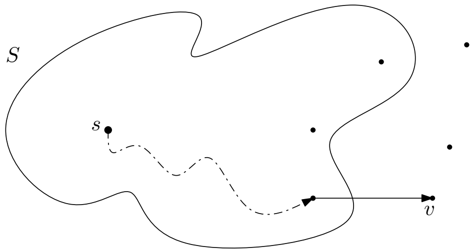
PROOF: We say that a path from s to a vertex v is *special* if all vertices on that path belong to S, except maybe v.



A *special path* from s to v.

Special Paths and Induction Hypotheses

PROOF: We say that a path from s to a vertex v is *special* if all vertices on that path belong to S, except maybe v.



A *special path* from s to v.

We prove by induction that

(a) if a vertex u is in S, then $d(u)$ gives the length of a shortest path from s to u and

(b) if a vertex u is not in S, then $d(u)$ gives the length of a shortest *special path* from s to u.

(a) if a vertex u is in S, then $d(u)$ gives the length of a shortest path from s to u and

(b) if a vertex u is not in S, then $d(u)$ gives the length of a shortest *special path* from s to u.

(a) if a vertex u is in S, then $d(u)$ gives the length of a shortest path from s to u and

(b) if a vertex u is not in S, then $d(u)$ gives the length of a shortest *special path* from s to u.

Base case:

(a) At the beginning, $S = \{\,\}$, so (a) is vacuously true.

(a) if a vertex u is in S, then $d(u)$ gives the length of a shortest path from s to u and
(b) if a vertex u is not in S, then $d(u)$ gives the length of a shortest *special path* from s to u.

Base case:

(a) At the beginning, $S = \{\,\}$, so (a) is vacuously true.
(b) Since $S = \{\,\}$, the other vertices simply cannot be reached by following a special path from s. Since d is initialized to ∞, then (b) also holds when the algorithm starts.

(a) if a vertex u is in S, then $d(u)$ gives the length of a shortest path from s to u and
(b) if a vertex u is not in S, then $d(u)$ gives the length of a shortest *special path* from s to u.

Base case:

(a) At the beginning, $S = \{\,\}$, so (a) is vacuously true.
(b) Since $S = \{\,\}$, the other vertices simply cannot be reached by following a special path from s. Since d is initialized to ∞, then (b) also holds when the algorithm starts.

Induction hypothesis: Assume that both (a) and (b) hold right before we add a new vertex v to S.

(a) if a vertex u is in S, then $d(u)$ gives the length of a shortest path from s to u and
(b) if a vertex u is not in S, then $d(u)$ gives the length of a shortest *special path* from s to u.

Base case:

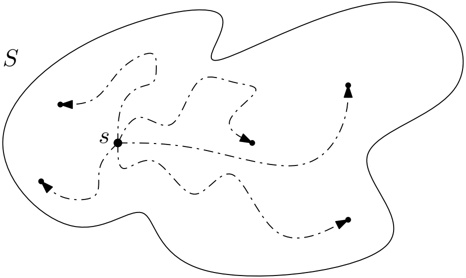
(a) At the beginning, $S = \{\,\}$, so (a) is vacuously true.
(b) Since $S = \{\,\}$, the other vertices simply cannot be reached by following a special path from s. Since d is initialized to ∞, then (b) also holds when the algorithm starts.

Induction hypothesis: Assume that both (a) and (b) hold right before we add a new vertex v to S.

We address induction steps (a) and (b) separately.

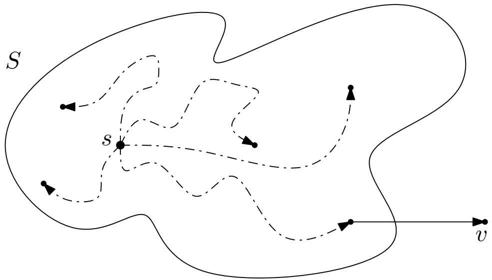
Induction Step (a)

Induction step (a): By the induction hypothesis (a), before the addition of v, we already know a shortest paths from s to all vertices that are in S. Adding v to S does not change these shortest paths.



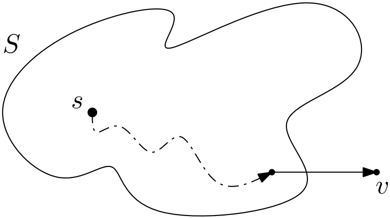
Induction Step (a)

Induction step (a): By the induction hypothesis (a), before the addition of v, we already know a shortest paths from s to all vertices that are in S. Adding v to S does not change these shortest paths.



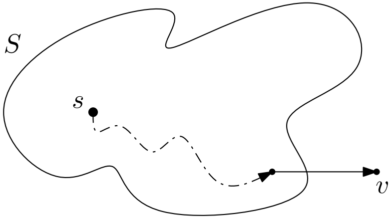
As for node v, it is about to be inserted in S. Before adding it to S, we must check that $d(v)$ gives the length of a shortest path from s to v.

Induction Step (a)



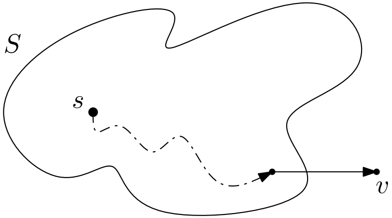
So we compare $d(v)$ to the lengths of all paths from s to v.

Induction Step (a)



So we compare $d(v)$ to the lengths of all paths from s to v.
There are two kinds of paths: (1) the paths that are special and (2) the ones that are not special.

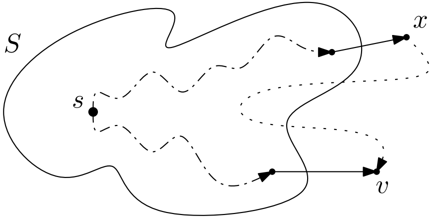
Induction Step (a)



So we compare $d(v)$ to the lengths of all paths from s to v.
There are two kinds of paths: (1) the paths that are special and (2) the ones that are not special.

(1) By the induction hypothesis (b), we already know that $d(v)$ is less than or equal to the length of any special path from s to v.

Induction Step (a)

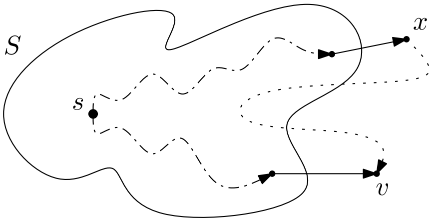


So we compare $d(v)$ to the lengths of all paths from s to v.
There are two kinds of paths: (1) the paths that are special and (2) the ones that are not special.

(1) By the induction hypothesis (b), we already know that $d(v)$ is less than or equal to the length of any special path from s to v.

(2) A non-special path from s to v is one which contains at least one vertex $x \neq v$ that is not in S.

Induction Step (a)



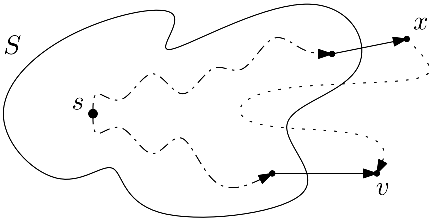
So we compare $d(v)$ to the lengths of all paths from s to v.
There are two kinds of paths: (1) the paths that are special and (2) the ones that are not special.

(1) By the induction hypothesis (b), we already know that $d(v)$ is less than or equal to the length of any special path from s to v.

(2) A non-special path from s to v is one which contains at least one vertex $x \neq v$ that is not in S. But such a path has length at least

$$d(x)$$

Induction Step (a)



So we compare $d(v)$ to the lengths of all paths from s to v.
There are two kinds of paths: (1) the paths that are special and (2) the ones that are not special.

(1) By the induction hypothesis (b), we already know that $d(v)$ is less than or equal to the length of any special path from s to v.

(2) A non-special path from s to v is one which contains at least one vertex $x \neq v$ that is not in S. But such a path has length at least

$$d(x) \geq d(v). \qquad \text{(do you see why?)}$$

Induction Step (a)

Hence, $d(v)$ is less than or equal to the length of any path from s to v,

Induction Step (a)

Hence, $d(v)$ is less than or equal to the length of any path from s to v, which means that $d(v)$ gives the length of a shortest path from s to v.

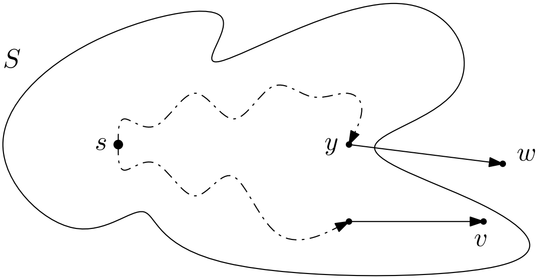
Induction Step (a)

Hence, $d(v)$ is less than or equal to the length of any path from s to v, which means that $d(v)$ gives the length of a shortest path from s to v.

So the induction step is complete for (a).

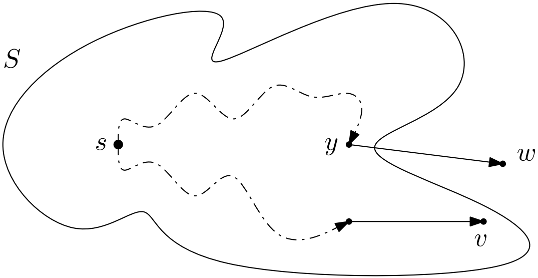
Induction Step (b)

Induction step (b): Consider a node $w \neq v$ which is not in S.



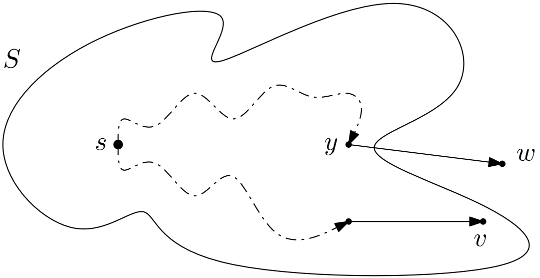
Induction Step (b)

Induction step (b): Consider a node $w \neq v$ which is not in S. Let y be the last vertex in S encountered on a shortest special path from s to w.



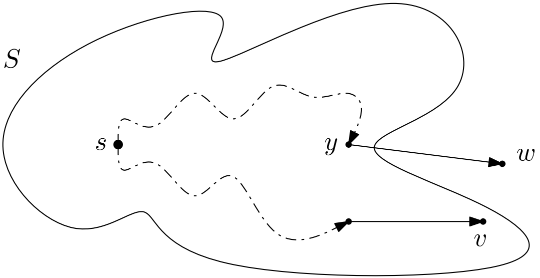
Induction Step (b)

Induction step (b): Consider a node $w \neq v$ which is not in S. Let y be the last vertex in S encountered on a shortest special path from s to w. We consider two cases: (1) $y \neq v$ or (2) $y = v$.



Induction Step (b)

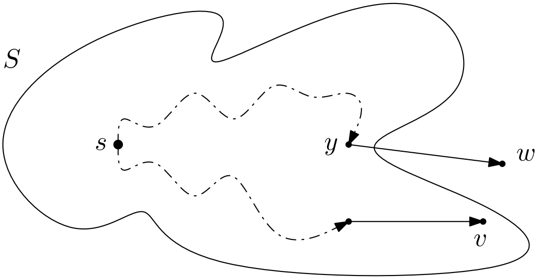
Induction step (b): Consider a node $w \neq v$ which is not in S. Let y be the last vertex in S encountered on a shortest special path from s to w. We consider two cases: (1) $y \neq v$ or (2) $y = v$.



(1) If $y \neq v$, then adding v to S does not change the value of $d(w)$.

Induction Step (b)

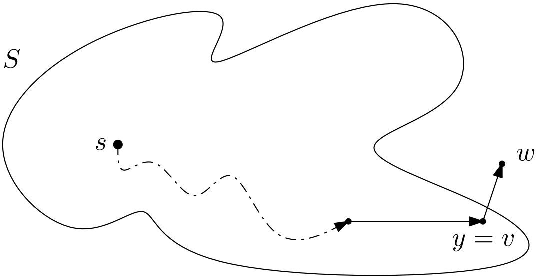
Induction step (b): Consider a node $w \neq v$ which is not in S. Let y be the last vertex in S encountered on a shortest special path from s to w. We consider two cases: (1) $y \neq v$ or (2) $y = v$.



(1) If $y \neq v$, then adding v to S does not change the value of $d(w)$.
Hence, by the induction hypothesis (b),
$d(w)$ gives the length of a shortest special path from s to w.

Induction Step (b)

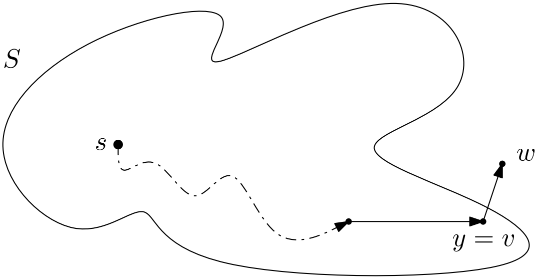
Induction step (b): Consider a node $w \neq v$ which is not in S. Let y be the last vertex in S encountered on a shortest special path from s to w. We consider two cases: (1) $y \neq v$ or (2) $y = v$.



(2) If $y = v$, then by doing $d(w) = \min\{d(w), d(v) + wt(v, w)\}$,

Induction Step (b)

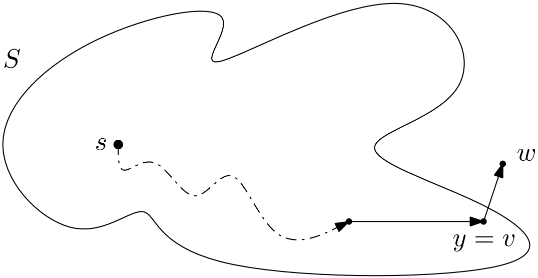
Induction step (b): Consider a node $w \neq v$ which is not in S. Let y be the last vertex in S encountered on a shortest special path from s to w. We consider two cases: (1) $y \neq v$ or (2) $y = v$.



(2) If $y = v$, then by doing $d(w) = \min\{d(w), d(v) + wt(v, w)\}$,
we ensure that after adding v to S,
$d(w)$ gives the length of a shortest special path from s to w.

Induction Step (b)

Induction step (b): Consider a node $w \neq v$ which is not in S. Let y be the last vertex in S encountered on a shortest special path from s to w. We consider two cases: (1) $y \neq v$ or (2) $y = v$.



(2) If $y = v$, then by doing $d(w) = \min\{d(w), d(v) + wt(v, w)\}$,

we ensure that after adding v to S,

$d(w)$ gives the length of a shortest special path from s to w. □

Question: Design a deterministic algorithm to solve the following problem.

input: A weighted directed graph $G = (V, E)$ together with a source vertex $s \in V$.

output: For each vertex $v \in V$, return TRUE if there is a *unique* shortest path from s to v and return FALSE if there is more than one shortest path from s to v.

What do you think of the following solution?

Initialize $unique[v] = \text{TRUE}$ for all vertices $v \in V$.

Change the for-loop (inside the while-loop) by

for each edge (u, v) **do**
    if $d[u] + wt(u, v) < d[v]$ **then**
        $d[v] = d[u] + wt(u, v)$
    else if $d[u] + wt(u, v) = d[v]$ **then**
        $unique[v] = \text{FALSE}$.
    end if
end for

What do you think of the following solution?

Initialize $unique[v] = \text{TRUE}$ for all vertices $v \in V$.

Change the for-loop (inside the while-loop) by

for each edge (u, v) **do**
  if $d[u] + wt(u, v) < d[v]$ **then**
    $d[v] = d[u] + wt(u, v)$
    $unique[v] = \text{TRUE}$
  else if $d[u] + wt(u, v) = d[v]$ **then**
    $unique[v] = \text{FALSE}$
  end if
end for

What do you think of the following solution?

Initialize $unique[v] = \text{TRUE}$ for all vertices $v \in V$.

Change the for-loop (inside the while-loop) by

for each edge (u, v) **do**
    if $d[u] + wt(u, v) < d[v]$ **then**
        $d[v] = d[u] + wt(u, v)$
        if $unique[u] = \text{FALSE}$ **then**
            $unique[v] = \text{FALSE}$
        end if
    else if $d[u] + wt(u, v) = d[v]$ **then**
        $unique[v] = \text{FALSE}$
    end if

What do you think of the following solution?

Initialize $unique[v] = \text{TRUE}$ for all vertices $v \in V$.

Change the for-loop (inside the while-loop) by

for each edge (u, v) **do**
  if $d[u] + wt(u, v) < d[v]$ **then**
    $d[v] = d[u] + wt(u, v)$
    if $unique[u] = \text{FALSE}$ **then**
      $unique[v] = \text{FALSE}$
    else
      $unique[v] = \text{TRUE}$
    end if
  else if $d[u] + wt(u, v) = d[v]$ **then**
    $unique[v] = \text{FALSE}$
  end if
end for

What if you want to know whether or not there are at least 3 different shortest paths?

What if you want to know the exact number of shortest paths from s to any vertex $v \in V$?