Single-Source Shortest Paths
Dijkstra's algorithm

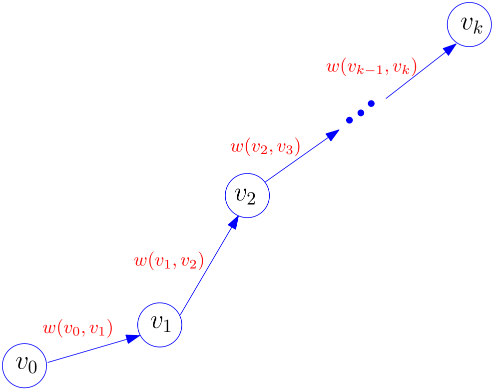
Armin Jamshidpey

Institute for Quantum Computing
University of Waterloo

Fall 2022

Review

- $G = (V, E)$ a directed graph with a weight function: $w : E \to \mathbb{R}$
- The weight of path $P = \langle v_0, \ldots, v_k \rangle$ is: $w(P) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$
- G has no negative-weight cycles



- The shortest path weight from u to v:

$$\delta(u, v) = \begin{cases} \min\{w(P) : u \overset{P}{\leadsto} v\} & \text{if there exists a path from u to v} \\ \infty & \text{otherwise} \end{cases}$$

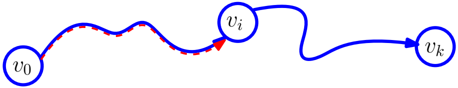
Review

Single-Source Shortest Path Problem

Input: $G = (V, E)$, $w : E \to \mathbb{R}$ and a source $s \in V$

Output: A shortest path from s to each $v \in V$

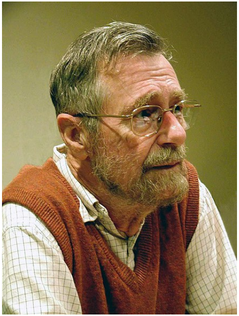
True/False: If $\langle v_0, v_1, \ldots, v_k \rangle$ is a shortest path from v_0 to v_k, then $\langle v_0, v_1, \ldots, v_i \rangle$ is a shortest path from v_0 to v_i, for any $0 \leq i \leq k$.



Padlet: https://padlet.com/arminjamshidpey/rlwrxbus8cosxrpg

Overview

- Explanation of Dijkstra's algorithm
- Pseudocode of the algorithm
- An example
- Complexity analysis
- Proof of correctness



E. Dijkstra (1930-2002)
Turing Award (1972)

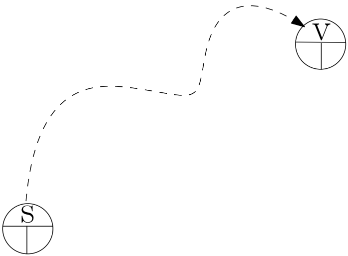
"Computer Science is no more about computers than astronomy is about telescopes."
-E. Dijkstra

Dijkstra's algorithm: Explanation

Dijkstra's algorithm is a greedy algorithm

Input: A weighted directed graph with non-negative edge weights

- For all vertices, maintain quantities
  - $d[v]$: a shortest-path estimate from s to v
  - $\pi[v]$: predecessor in the path (a vertex or NIL)

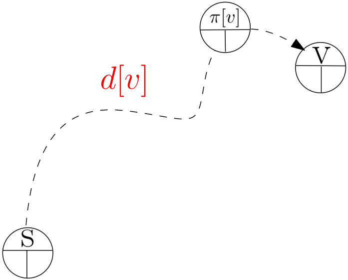


Dijkstra's algorithm: Explanation

Dijkstra's algorithm is a greedy algorithm

Input: A weighted directed graph with non-negative edge weights

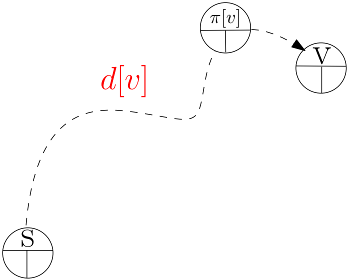
- For all vertices, maintain quantities
  - $d[v]$: a shortest-path estimate from s to v
  - $\pi[v]$: predecessor in the path (a vertex or NIL)



Dijkstra's algorithm: Explanation

Dijkstra's algorithm is a greedy algorithm

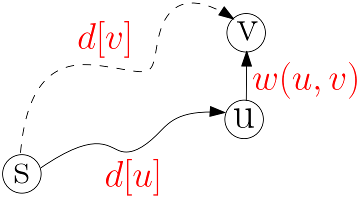
Input: A weighted directed graph with non-negative edge weights



- For all vertices, maintain quantities
  - $d[v]$: a shortest-path estimate from s to v
  - $\pi[v]$: predecessor in the path (a vertex or NIL)
- Initialize $C = \emptyset$, repeat the following until $C = V$:
  - Add $u \in V - C$ with smallest d-value to C
  - Update d-values of vertices v with $(u, v) \in E$:

$$d[v] \leftarrow \min\{d[v], d[u] + w(u, v)\}$$

  - Update $\pi[v]$ if $d[v]$ is changed



Dijkstra's algorithm: Explanation

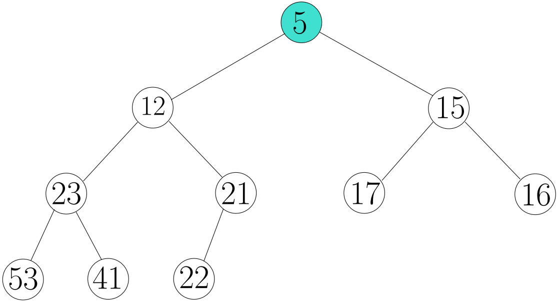
Which Abstract Data Type (ADT) should we use for vertices?

Padlet: https://padlet.com/arminjamshidpey/rlwrxbus8cosxrpg

Dijkstra's algorithm: Explanation

Which Abstract Data Type (ADT) should we use for vertices?

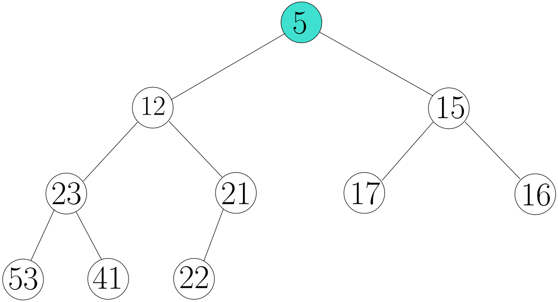
Padlet: https://padlet.com/arminjamshidpey/rlwrxbus8cosxrpg



Dijkstra's algorithm: Explanation

Which Abstract Data Type (ADT) should we use for vertices?

Padlet: https://padlet.com/arminjamshidpey/rlwrxbus8cosxrpg



Cost of operations of a binary min-heap (of size n):

- Insert: $O(\log n)$
- Extract-Min: $O(\log n)$
- Update-Key: $O(\log n)$

Dijkstra's algorithm: Pseudocode

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$

Dijkstra's algorithm: Pseudocode

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$

Dijkstra's algorithm: Pseudocode

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    $u \leftarrow$ EXTRACT-MIN(Q)
    $C \leftarrow C \cup \{u\}$

Dijkstra's algorithm: Pseudocode

```
Dijkstra(G, w, s)
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← Extract-Min(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

DIJKSTRA(G, w, s)

```
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← EXTRACT-MIN(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

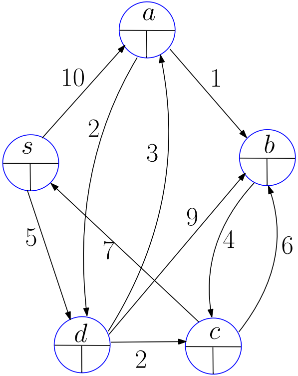


Figure: An example from CLRS

Dijkstra's algorithm: Example

Dijkstra(G, w, s)

```
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← Extract-Min(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

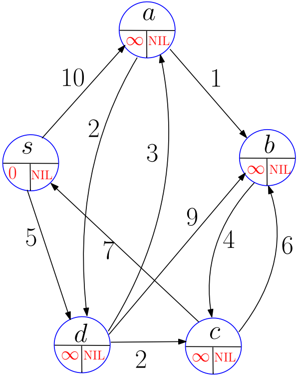


Figure: An example from CLRS

Dijkstra's algorithm: Example

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $C \leftarrow C \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $d[v] > d[u] + w(u, v)$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

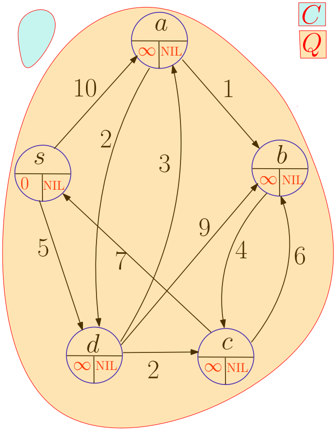


Figure: An example from CLRS

Dijkstra's algorithm: Example

$\text{DIJKSTRA}(G, w, s)$

for each vertex $v \in V[G]$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow \text{EXTRACT-MIN}(Q)$
  $C \leftarrow C \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $d[v] > d[u] + w(u, v)$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

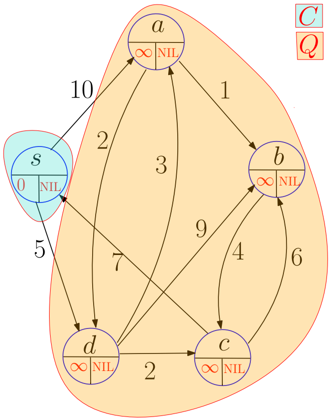


Figure: An example from CLRS

Dijkstra's algorithm: Example

DIJKSTRA(G, w, s)

```
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← EXTRACT-MIN(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

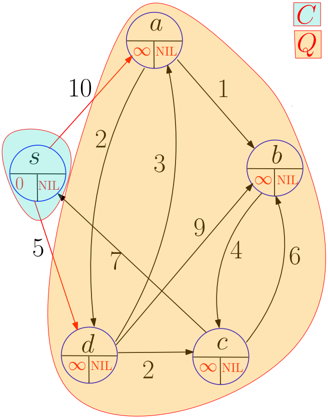


Figure: An example from CLRS

Dijkstra's algorithm: Example

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $C \leftarrow C \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $d[v] > d[u] + w(u, v)$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

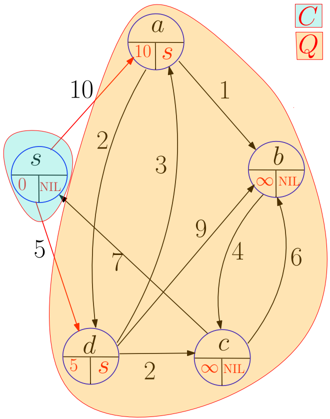


Figure: An example from CLRS

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    $u \leftarrow$ EXTRACT-MIN(Q)
    $C \leftarrow C \cup \{u\}$
    for each vertex $v \in Adj[u]$
        if $d[v] > d[u] + w(u, v)$
            $d[v] \leftarrow d[u] + w(u, v)$
            $\pi[v] \leftarrow u$

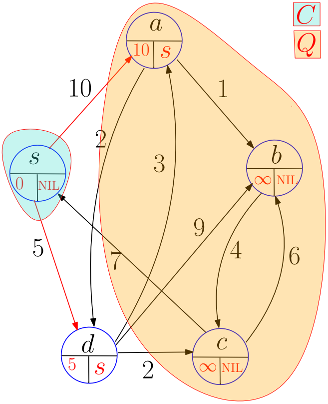


Figure: An example from CLRS

Dijkstra's algorithm: Example

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    $u \leftarrow$ EXTRACT-MIN(Q)
    $C \leftarrow C \cup \{u\}$
    for each vertex $v \in Adj[u]$
        if $d[v] > d[u] + w(u, v)$
            $d[v] \leftarrow d[u] + w(u, v)$
            $\pi[v] \leftarrow u$

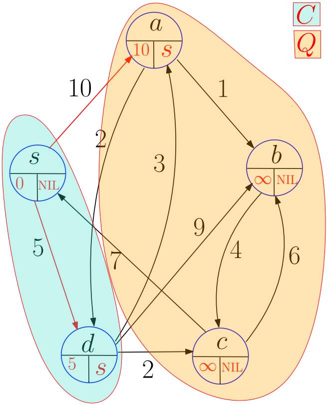


Figure: An example from CLRS

DIJKSTRA(G, w, s)

```
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← EXTRACT-MIN(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

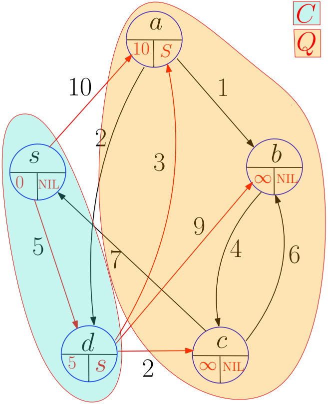


Figure: An example from CLRS

Dijkstra's algorithm: Example

```
DIJKSTRA(G, w, s)
  for each vertex v ∈ V[G]
        d[v] ← ∞
        π[v] ← NIL
  d[s] ← 0
  C ← ∅
  Q ← V[G]
  while Q ≠ ∅
        u ← EXTRACT-MIN(Q)
        C ← C ∪ {u}
        for each vertex v ∈ Adj[u]
              if d[v] > d[u] + w(u, v)
                    d[v] ← d[u] + w(u, v)
                    π[v] ← u
```

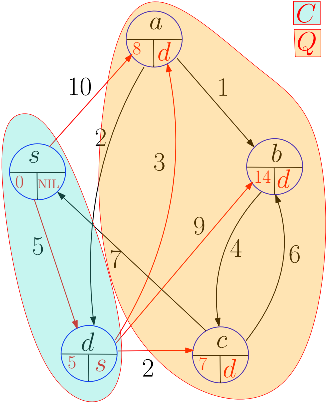


Figure: An example from CLRS

Dijkstra's algorithm: Example

```
DIJKSTRA(G, w, s)
for each vertex v ∈ V[G]
        d[v] ← ∞
        π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
        u ← EXTRACT-MIN(Q)
        C ← C ∪ {u}
        for each vertex v ∈ Adj[u]
                if d[v] > d[u] + w(u, v)
                        d[v] ← d[u] + w(u, v)
                        π[v] ← u
```

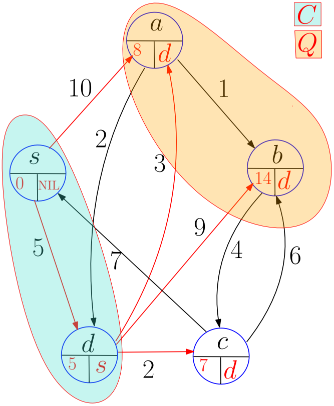


Figure: An example from CLRS

Dijkstra's algorithm: Example

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $C \leftarrow C \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $d[v] > d[u] + w(u, v)$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

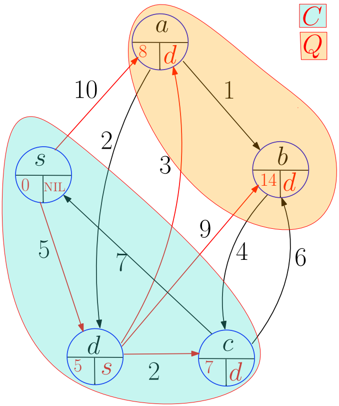


Figure: An example from CLRS

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $C \leftarrow C \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $d[v] > d[u] + w(u, v)$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

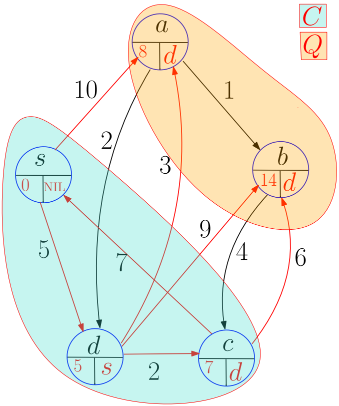


Figure: An example from CLRS

Dijkstra's algorithm: Example

Dijkstra(G, w, s)

```
for each vertex v ∈ V[G]
      d[v] ← ∞
      π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
      u ← Extract-Min(Q)
      C ← C ∪ {u}
      for each vertex v ∈ Adj[u]
            if d[v] > d[u] + w(u, v)
                  d[v] ← d[u] + w(u, v)
                  π[v] ← u
```

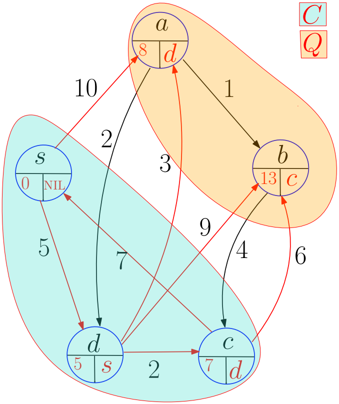


Figure: An example from CLRS

Dijkstra's algorithm: Example

```
DIJKSTRA(G, w, s)
for each vertex v ∈ V[G]
        d[v] ← ∞
        π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
        u ← EXTRACT-MIN(Q)
        C ← C ∪ {u}
        for each vertex v ∈ Adj[u]
                if d[v] > d[u] + w(u, v)
                        d[v] ← d[u] + w(u, v)
                        π[v] ← u
```

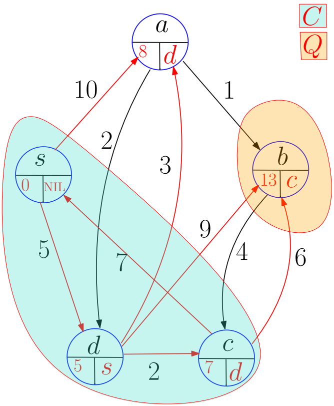


Figure: An example from CLRS

Dijkstra's algorithm: Example

$\textsc{Dijkstra}(G, w, s)$

```
for each vertex v ∈ V[G]
        d[v] ← ∞
        π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
        u ← Extract-Min(Q)
        C ← C ∪ {u}
        for each vertex v ∈ Adj[u]
                if d[v] > d[u] + w(u, v)
                        d[v] ← d[u] + w(u, v)
                        π[v] ← u
```

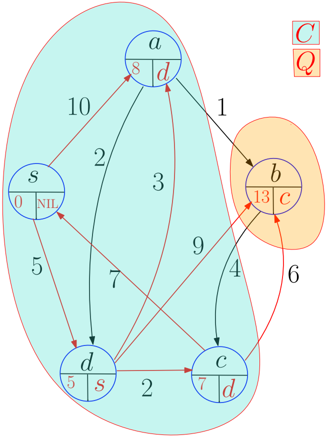


Figure: An example from CLRS

Dijkstra's algorithm: Example

DIJKSTRA(G, w, s)

```
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← EXTRACT-MIN(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

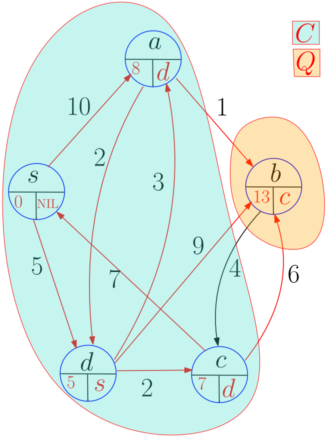


Figure: An example from CLRS

```
DIJKSTRA(G, w, s)
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← EXTRACT-MIN(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

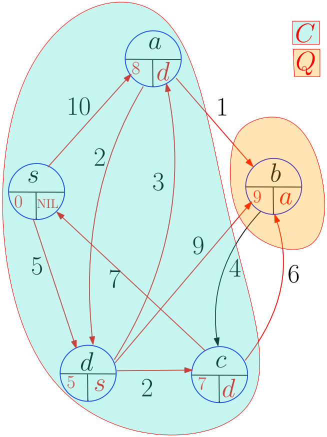


Figure: An example from CLRS

Dijkstra's algorithm: Example

```
DIJKSTRA(G, w, s)
for each vertex v ∈ V[G]
    d[v] ← ∞
    π[v] ← NIL
d[s] ← 0
C ← ∅
Q ← V[G]
while Q ≠ ∅
    u ← EXTRACT-MIN(Q)
    C ← C ∪ {u}
    for each vertex v ∈ Adj[u]
        if d[v] > d[u] + w(u, v)
            d[v] ← d[u] + w(u, v)
            π[v] ← u
```

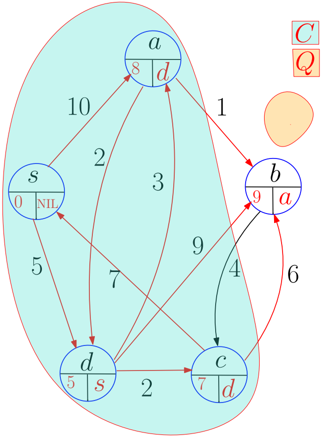


Figure: An example from CLRS

Dijkstra's algorithm: Example

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $C \leftarrow C \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $d[v] > d[u] + w(u, v)$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

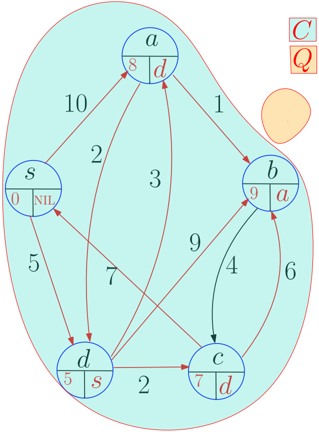


Figure: An example from CLRS

Dijkstra's algorithm: Example

$\text{DIJKSTRA}(G, w, s)$

for each vertex $v \in V[G]$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    $u \leftarrow \text{EXTRACT-MIN}(Q)$
    $C \leftarrow C \cup \{u\}$
    for each vertex $v \in Adj[u]$
        if $d[v] > d[u] + w(u, v)$
            $d[v] \leftarrow d[u] + w(u, v)$
            $\pi[v] \leftarrow u$

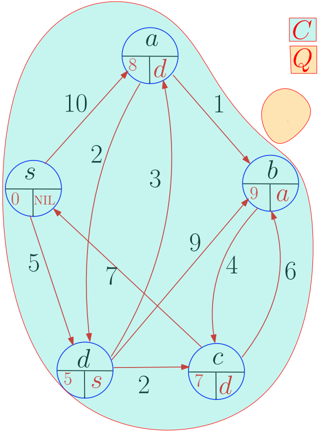


Figure: An example from CLRS

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
  $d[v] \leftarrow \infty$
  $\pi[v] \leftarrow \text{NIL}$
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
  $u \leftarrow$ EXTRACT-MIN(Q)
  $C \leftarrow C \cup \{u\}$
  for each vertex $v \in Adj[u]$
    if $d[v] > d[u] + w(u, v)$
      $d[v] \leftarrow d[u] + w(u, v)$
      $\pi[v] \leftarrow u$

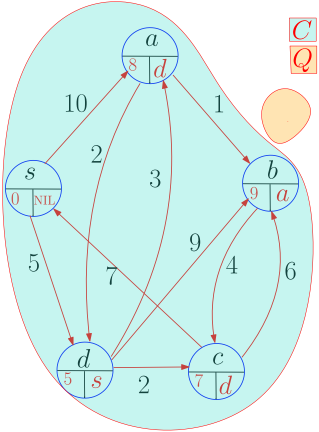


Figure: An example from CLRS

DIJKSTRA(G, w, s)

for each vertex $v \in V[G]$
    $d[v] \leftarrow \infty$
    $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$
$C \leftarrow \emptyset$
$Q \leftarrow V[G]$
while $Q \neq \emptyset$
    $u \leftarrow$ EXTRACT-MIN(Q)
    $C \leftarrow C \cup \{u\}$
    for each vertex $v \in Adj[u]$
        if $d[v] > d[u] + w(u, v)$
            $d[v] \leftarrow d[u] + w(u, v)$
            $\pi[v] \leftarrow u$

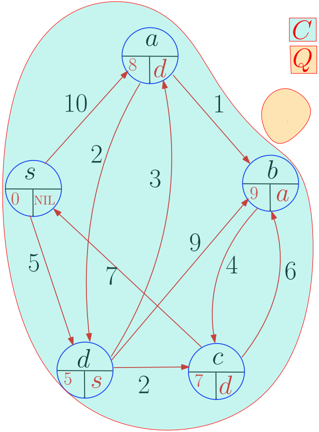


Figure: An example from CLRS

Dijkstra's algorithm: Complexity analysis

```
DIJKSTRA(G, w, s)
 1  for each vertex v ∈ V[G]
 2        d[v] ← ∞
 3        π[v] ← NIL
 4  d[s] ← 0
 5  C ← ∅
 6  Q ← V[G]
 7  while Q ≠ ∅
 8        u ← EXTRACT-MIN(Q)
 9        C ← C ∪ {u}
10        for each vertex v ∈ Adj[u]
11              if d[v] > d[u] + w(u, v)
12                    d[v] ← d[u] + w(u, v)
13                    π[v] ← u
```

line	Array Imp.	Heap Imp.
1		
2		
3	$O(\lvert V\rvert)$	
4		
5		
6	$O(\lvert V\rvert)$	
7	$O(\lvert V\rvert)$ iterations	
8	$O(\lvert V\rvert^2)$	
9		
10	$O(\lvert E\rvert)$ iterations	
11		
12	$O(\lvert E\rvert)$	
13		
Tot.	$O(\lvert V\rvert^2)$	

Dijkstra's algorithm: Complexity analysis

```
Dijkstra(G, w, s)
1   for each vertex v ∈ V[G]
2       d[v] ← ∞
3       π[v] ← NIL
4   d[s] ← 0
5   C ← ∅
6   Q ← V[G]
7   while Q ≠ ∅
8       u ← Extract-Min(Q)
9       C ← C ∪ {u}
10      for each vertex v ∈ Adj[u]
11          if d[v] > d[u] + w(u, v)
12              d[v] ← d[u] + w(u, v)
13              π[v] ← u
```

line	Array Imp.	Heap Imp.
1		
2		
3	$O(\|V\|)$	$O(\|V\|)$
4		
5		
6	$O(\|V\|)$	$O(\|V\|)$
7	$O(\|V\|)$ iter.	$O(\|V\|)$ iter.
8	$O(\|V\|^2)$	$O(\|V\| \log \|V\|)$
9		
10	$O(\|E\|)$ iter.	$O(\|E\|)$ iter.
11		
12	$O(\|E\|)$	$O(\|E\| \log \|V\|)$
13		
Tot.	$O(\|V\|^2)$	$O\Big((\|V\| + \|E\|) \log \|V\|\Big)$

Note: The of number of iterations on lines 7 and 10, are already considered in the cost of lines 8-9 and 11-13

Dijkstra's algorithm: Proof of Correctness

Claim: For each vertex $v \in V$, we have $d[v] = \delta(s, v)$ at the time when v is added to set C. To prove the claim, we follow these steps:

Dijkstra's algorithm: Proof of Correctness

Claim: For each vertex $v \in V$, we have $d[v] = \delta(s, v)$ at the time when v is added to set C. To prove the claim, we follow these steps:

1. **Proof by contradiction:** Assume the claim is **not correct** and $u \in V$ is the first vertex for which $d[u] \neq \delta(s, u)$ when it is added to C.

Dijkstra's algorithm: Proof of Correctness

Claim: For each vertex $v \in V$, we have $d[v] = \delta(s, v)$ at the time when v is added to set C. To prove the claim, we follow these steps:

1. **Proof by contradiction:** Assume the claim is **not correct** and $u \in V$ is the first vertex for which $d[u] \neq \delta(s, u)$ when it is added to C.
2. Time t: Beginning of the iteration in which u is added to C.

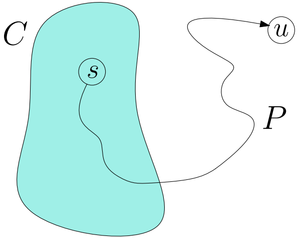
Dijkstra's algorithm: Proof of Correctness

Claim: For each vertex $v \in V$, we have $d[v] = \delta(s, v)$ at the time when v is added to set C. To prove the claim, we follow these steps:

1. **Proof by contradiction:** Assume the claim is **not correct** and $u \in V$ is the first vertex for which $d[u] \neq \delta(s, u)$ when it is added to C.

2. Time t: Beginning of the iteration in which u is added to C.

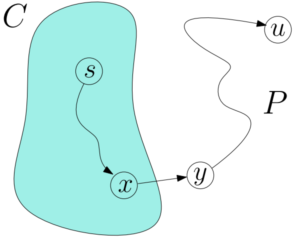
3. Use a shortest path P, from s to u (needs justification)



C covers a part of P at time t

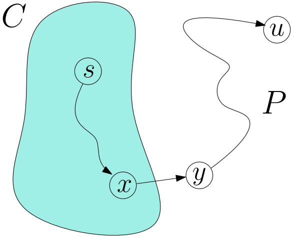
Dijkstra's algorithm: Proof of Correctness

4. On P, find y (with predecessor x), the first vertex in $V - C$.



Dijkstra's algorithm: Proof of Correctness

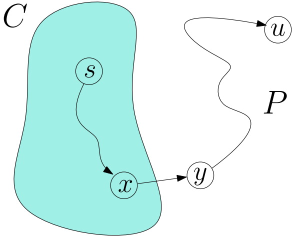
4. On P, find y (with predecessor x), the first vertex in $V - C$.



5. At time t, we have $d[y] = \delta(s, y)$. (needs Justification)

Dijkstra's algorithm: Proof of Correctness

4. On P, find y (with predecessor x), the first vertex in $V - C$.

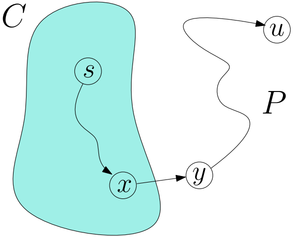


5. At time t, we have $d[y] = \delta(s, y)$. (needs Justification)
6. Fact:

$$d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u] \quad (1)$$

Dijkstra's algorithm: Proof of Correctness

4. On P, find y (with predecessor x), the first vertex in $V - C$.



5. At time t, we have $d[y] = \delta(s, y)$. (needs Justification)
6. Fact:

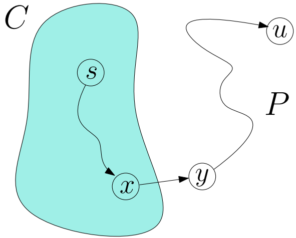
$$d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u] \tag{1}$$

7. Another fact: at time t the algorithm chose u. Hence

$$d[u] \leq d[y] \tag{2}$$

Dijkstra's algorithm: Proof of Correctness

4. On P, find y (with predecessor x), the first vertex in $V - C$.



5. At time t, we have $d[y] = \delta(s, y)$. (needs Justification)
6. Fact:

$$d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u] \tag{1}$$

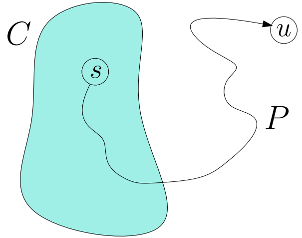
7. Another fact: at time t the algorithm chose u. Hence

$$d[u] \leq d[y] \tag{2}$$

8. From equations (1) and (2) we conclude

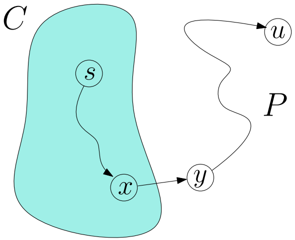
$$d[y] = \delta(s, y) = \delta(s, u) = d[u]$$

3. Use a shortest path P, from s to u (needs justification)
   Proof:



C covers a part of P at time t

5. At time t, we have $d[y] = \delta(s, y)$.
Proof:



CS341 Critical Incident Questionnaire

https://docs.google.com/forms/d/1R0EuR500MfIx3BZnum2Lad8ezU61gzcBLaClsW0dInc/edit