Lecture 11

Instructor: Subrahmanyam Kalyanasundaram

23rd September 2019

Plan

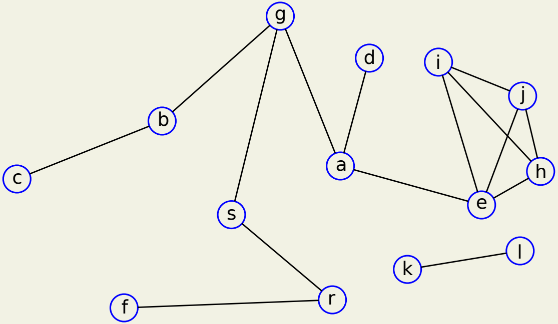
- Complete the proof of correctness of BFS
- After that, we see shortest path in weighted graphs

Algorithm 1 Breadth-first Search from vertex s

Color all vertices WHITE.
For all $u \in V$, $d[u] \leftarrow \infty$, $\pi[u] \leftarrow$ NIL.
$d[s] \leftarrow 0$, color$[s] \leftarrow$ GRAY.
Initialize queue $Q \leftarrow \emptyset$.
ENQUEUE(Q, s)
while $Q \neq \emptyset$ **do**
    $u \leftarrow$ DEQUEUE(Q)
    for each $v \in \mathcal{N}(u)$ **do**
        if color(v) =WHITE **then**
            color$[v] \leftarrow$ GRAY
            $d[v] \leftarrow d[u] + 1$
            $\pi[v] \leftarrow u$
            ENQUEUE(Q, v)
        end if
    end for
    color$[u] \leftarrow$ BLACK.
end while

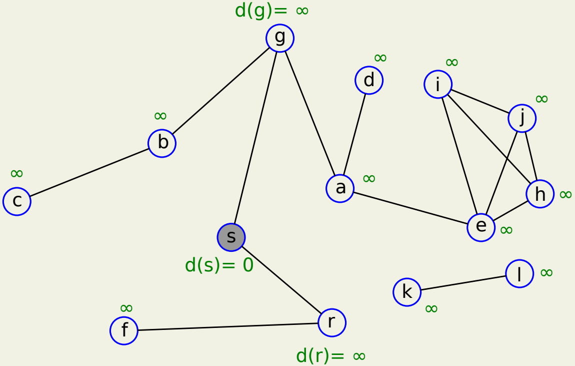
Breadth-first Search

Queue: $\emptyset$



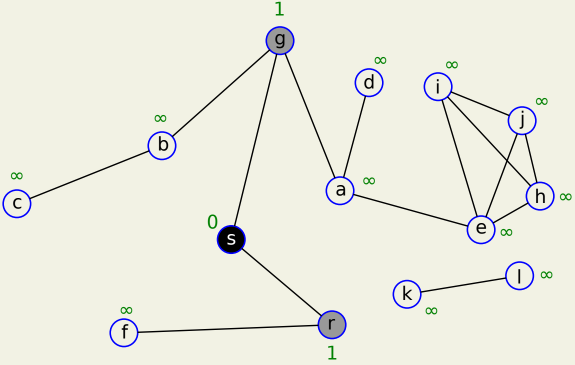
Breadth-first Search

Dequeued vertex: $\boxed{}$ Queue: $\boxed{s}$



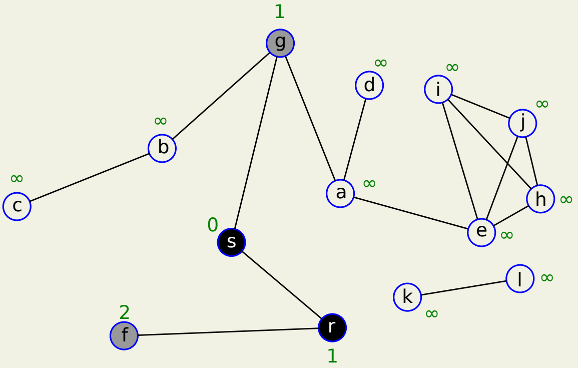
Breadth-first Search

Dequeued vertex: | s | Queue: | r | g |



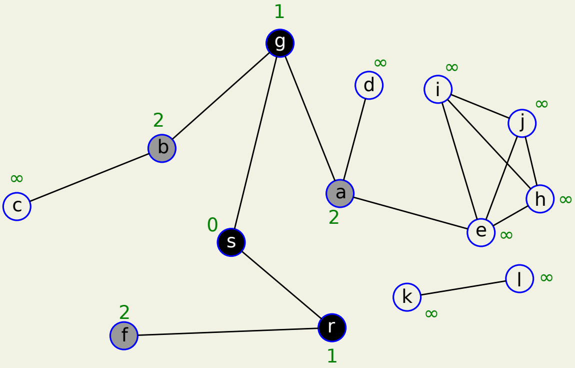
Breadth-first Search

Dequeued vertex: $\boxed{r}$ Queue: $\boxed{g \mid f}$



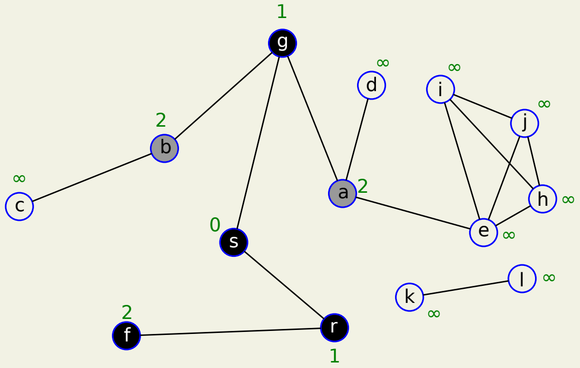
Breadth-first Search

Dequeued vertex: | g | Queue: | f | a | b |



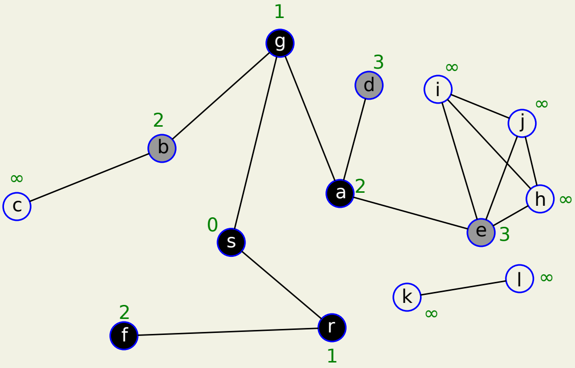
Breadth-first Search

Dequeued vertex: $\boxed{f}$ Queue: $\boxed{a \mid b}$



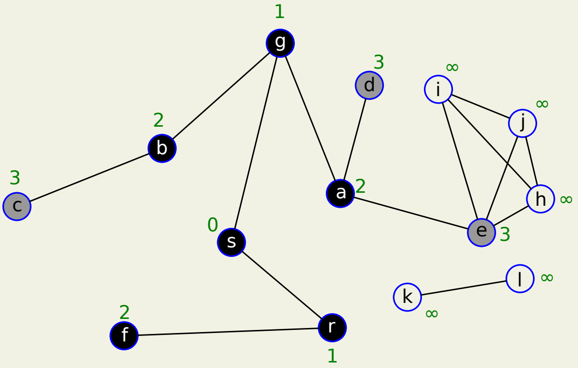
Breadth-first Search

Dequeued vertex: | a | Queue: | b | e | d |



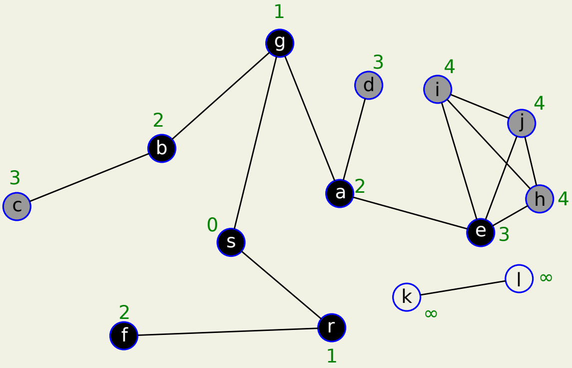
Breadth-first Search

Dequeued vertex: | b | Queue: | e | d | c |



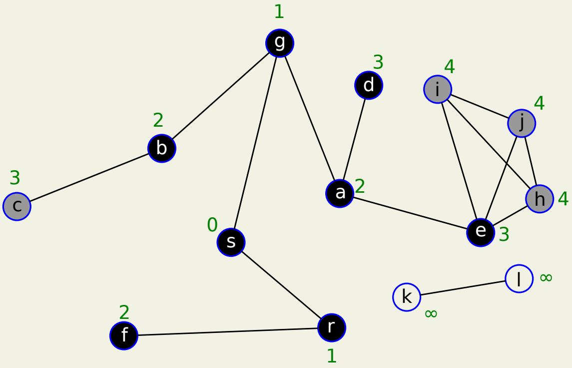
Breadth-first Search

Dequeued vertex: | *e* | Queue: | *d* | *c* | *j* | *h* | *i* |



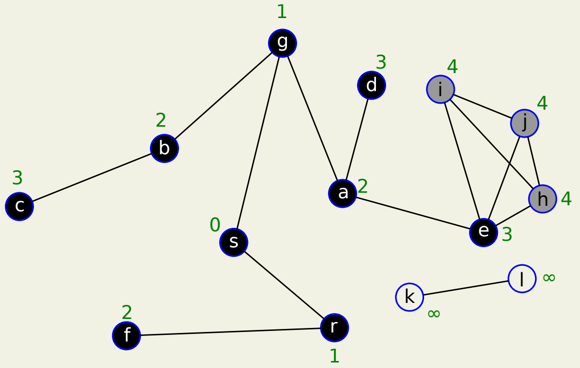
Breadth-first Search

Dequeued vertex: d Queue: | c | j | h | i |



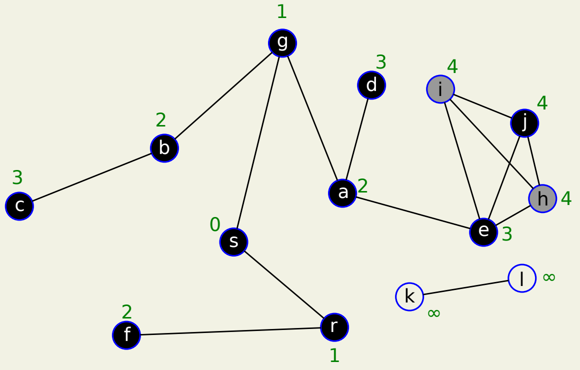
Breadth-first Search

Dequeued vertex: c Queue: j h i



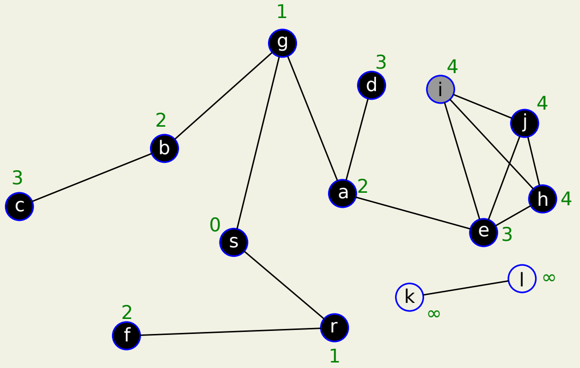
Breadth-first Search

Dequeued vertex: $\boxed{j}$ Queue: $\boxed{h \mid i}$



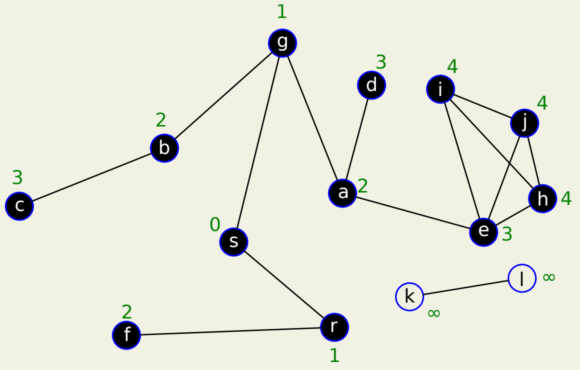
Breadth-first Search

Dequeued vertex: $\boxed{h}$ Queue: $\boxed{i}$



Breadth-first Search

Dequeued vertex: $\boxed{i}$ Queue: $\emptyset$



Algorithm 2 Breadth-first Search from vertex s

Color all vertices WHITE.
For all $u \in V$, $d[u] \leftarrow \infty$, $\pi[u] \leftarrow$ NIL.
$d[s] \leftarrow 0$, color$[s] \leftarrow$ GRAY.
Initialize queue $Q \leftarrow \emptyset$.
ENQUEUE(Q, s)
while $Q \neq \emptyset$ **do**
    $u \leftarrow$ DEQUEUE(Q)
    for each $v \in \mathcal{N}(u)$ **do**
        if color(v) =WHITE **then**
            color$[v] \leftarrow$ GRAY
            $d[v] \leftarrow d[u] + 1$
            $\pi[v] \leftarrow u$
            ENQUEUE(Q, v)
        end if
    end for
    color$[u] \leftarrow$ BLACK.
end while

Time Complexity of BFS

- Each enqueue/dequeue takes $O(1)$ time.
- Total queue operations take $O(|V|)$ time.
- Each list in the adj. list is scanned once. This requires total $\Theta(|E|)$. This is assuming the graph is provided using adjacency list.
- Initialization required $\Theta(|V|)$.
- Total running time is $O(|V| + |E|)$.

Time Complexity of BFS

- Each enqueue/dequeue takes $O(1)$ time.
- Total queue operations take $O(|V|)$ time.
- Each list in the adj. list is scanned once. This requires total $\Theta(|E|)$. This is assuming the graph is provided using adjacency list.
- Initialization required $\Theta(|V|)$.
- Total running time is $O(|V| + |E|)$.

- **Note:** The colors can be omitted. Instead, check if $d[v] = \infty$

Correctness of BFS

Notation: Let $\delta(s, v)$ denote the minimum number of edges on a path from s to v.

Theorem

Let $G = (V, E)$ be a graph. When BFS is run on G from vertex $s \in V$:

1. Every vertex that is reachable from s gets discovered.
2. On termination, $d[v] = \delta(s, v)$ for all v.

We will first show (2).

Proof of correctness

Proof

Suppose, for the sake of contradiction, (2) does not hold.
Let v be the vertex with smallest $\delta(s, v)$ such that $d[v] \neq \delta(s, v)$.
Claim 1: $d[v] \geq \delta(s, v)$
Choose a *shortest* path from s to v.
Let u be the vertex immmediately preceding v.
Then $\delta(s, v) = \delta(s, u) + 1 = d[u] + 1$.
So we have:

$$d[v] > \delta(s, v) = \delta(s, u) + 1 = d[u] + 1$$

Proof of correctness

Proof cont...

We have:

$$d[v] > \delta(s, v) = \delta(s, u) + 1 = d[u] + 1$$

Consider the time step when u is dequeued.

- Case 1: v was white.
  The algo sets $d[v] = d[u] + 1$.
  This contradicts the eq above.
- Case 2: v is black.
  Then, v was dequeued before u.
  Claim 2: If v was dequeued before u, then $d[v] \leq d[u]$.

Proof of correctness

Proof cont...

- Case 3: v was gray.
  Vertex v was colored gray after dequeuing some vertex w earlier.
  So $d[v] = d[w] + 1$.
  By Claim 2, $d[w] \leq d[u]$ since w was dequeued before u.
  This gives: $d[v] = d[w] + 1 \leq d[u] + 1$.

Exercise

Show (1) using (2). That is, given that $d[v] = \delta(s, v)$, show that every vertex reachable from s gets discovered.

Proof of correctness

Claim 3

Let $(u, v) \in E$. Then we have:

$$\delta(s, v) \leq \delta(s, u) + 1$$

Proof

If u is reachable from s, then:
Take the shortest path from s to u. Then take the edge (u, v).
This gives a path from s to v.
The shortest path from s to v can only be shorter than the above path.

$\square$

Proof of correctness

Claim 1

$\forall v \in V, d[v] \geq \delta(s, v)$

Proof

Induction on the number of enqueue operations.
Hypothesis: same as claim.
Base case: The time when the first vertex enqueued.
The first vertex enqueued is s. At this time we have:

- $\forall v \in V \setminus \{s\}, d[v] = \infty$
- $d[s] = \delta(s, s) = 0.$

Hence the claim holds for the base case.

Proof of correctness

Proof

Hypothesis: $\forall v \in V, d[v] \geq \delta(s, v)$
Step: A white (undiscovered) vertex v gets discovered while we are visiting a vertex u with $(u, v) \in E$.
From induction, we have: $d[u] \geq \delta(s, u)$.
The algorithm assigns $d[v] \leftarrow d[u] + 1$. So:

$$\begin{aligned} d[v] &= d[u] + 1 \\ &\geq \delta(s, u) + 1 \\ &\geq \delta(s, v) \end{aligned}$$

Last inequality follows from Claim 3.

□

Proof of correctness

Claim 2

If v was dequeued before u, then $d[v] \leq d[u]$.

We will show a stronger claim:

Claim 4

If at some point, the queue contained $v_1, v_2, \ldots, v_r$ where v_1 was the head. Then:

(a) $d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r]$

(b) $d[v_r] \leq d[v_1] + 1$

Proof of Claim 2:

Write down vertices in the order they went through the queue.

By claim 4 (a), the calculated d values for them are non-decreasing.

Vertex v will appear before u in this order.

Hence claim 2 follows. $\square$

Proof of correctness

Claim 4

If queue contains $v_1, v_2, \ldots, v_r$ where v_1 is the head. Then:

(a) $d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r]$

(b) $d[v_r] \leq d[v_1] + 1$

Proof

Induction on number of queue operations.

Hypothesis: Same as claim. We show that the claim holds after every enqueue and dequeue.

Base case: The first queue operation - enqueuing s.

The claim trivially holds.

Proof of correctness

Claim 4

If queue contains $v_1, v_2, \ldots, v_r$ where v_1 is the head. Then:

(a) $d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r]$

(b) $d[v_r] \leq d[v_1] + 1$

Proof

Step:

- **Dequeue:** After v_1 is dequeued, v_2 is the new head.
  Part (a): From induction,
  $d[v_1] \leq d[v_2] \leq d[v_3] \leq \cdots \leq d[v_r]$.
  Hence (a) holds.
  Part (b): From induction, $d[v_r] \leq d[v_1] + 1$. And so:

$$\begin{aligned} d[v_r] &\leq d[v_1] + 1 \\ &\leq d[v_2] + 1 \end{aligned}$$

Proof of correctness

Proof

- **Enqueue:** When a vertex v is enqueued:
  It was enqueued because:
  - it was undiscovered so far.
  - it was present in the adjacency list of a vertex u that was just dequeued.

  Since u was the previous head of the list, from induction we have:
  - $d[u] \leq d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r]$.
  - $d[v_r] \leq d[u] + 1$.

  We assign $d[v] \leftarrow d[u] + 1$ and then enqueue v. Hence, we have:
  - $d[v_r] \leq d[u] + 1 = d[v]$
  - $d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r] \leq d[v]$.

$\square$

Loop Invariant

Claim 4

If queue contains $v_1, v_2, \ldots, v_r$ where v_1 is the head. Then:

(a) $d[v_1] \leq d[v_2] \leq \cdots \leq d[v_r]$

(b) $d[v_r] \leq d[v_1] + 1$

Claim 4 is actually a loop invariant!

Another loop invariant

The queue Q consists of the set of GRAY vertices.

Weighted Graphs

A weighted graph is a graph $G = (V, E)$ with a weight function:

$$w : E \to \mathbb{Z}$$

The weight of an edge $(u, v) \in E$ is $w((u, v))$.
For this lecture, we look at directed weighted graphs with weight function $w : E \to \mathbb{Z}^+$.

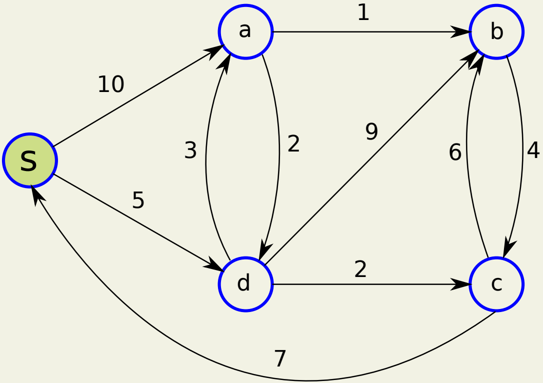
Shortest path in weighted graphs

Input:

- Graph $G = (V, E)$
- Weight function $w : E \to \mathbb{Z}^+$
- Source vertex $s \in V$.

Goal: Compute the shortest path from s to all reachable vertices.

Example graph

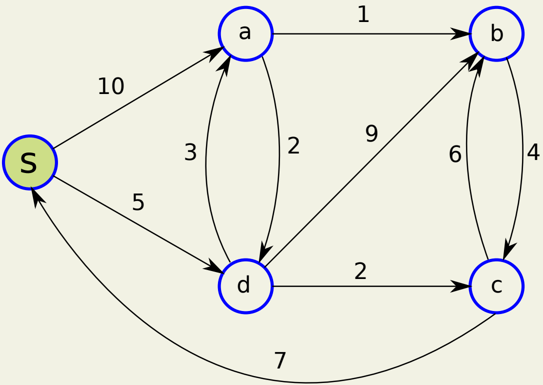


Dijkstra's Algorithm Pseudocode

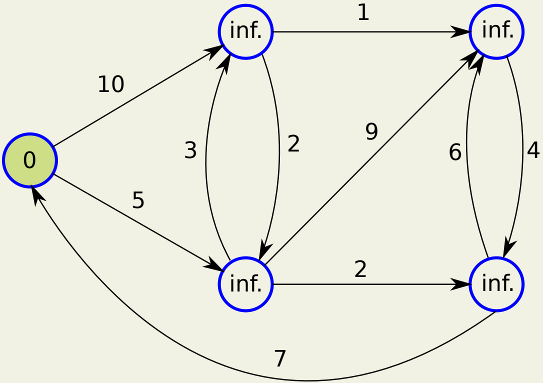
Algorithm 3 Dijkstra's algorithm

```
For all u ∈ V, d[u] ← ∞, π[u] ← NIL
d[s] ← 0
Initialize min-priority queue Q ← V
S ← ∅
while Q ≠ ∅ do
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each v ∈ 𝒩(u) do
        if d[u] + w(u, v) < d[v] then
            d[v] ← d[u] + w(u, v)
            DECREASE-KEY(v, d[v]).
            π[v] ← u
        end if
    end for
end while
```

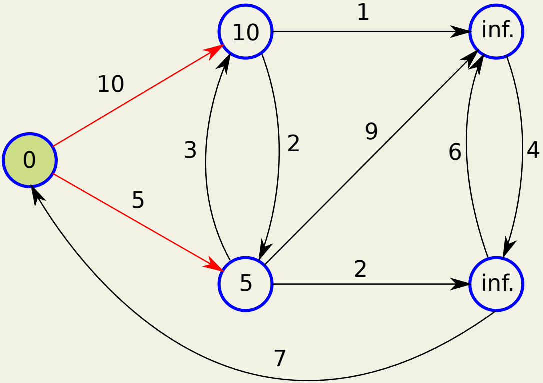
Example graph



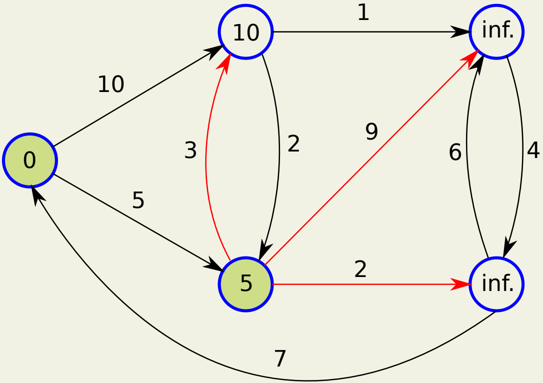
Example graph



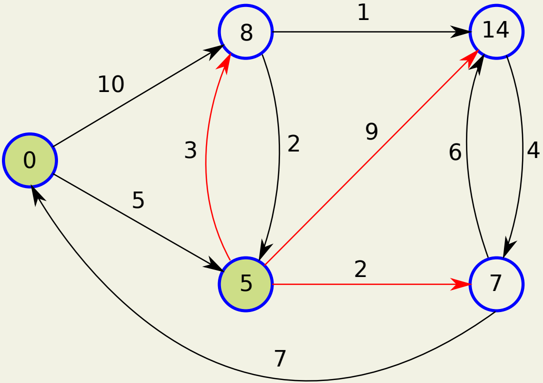
Example graph



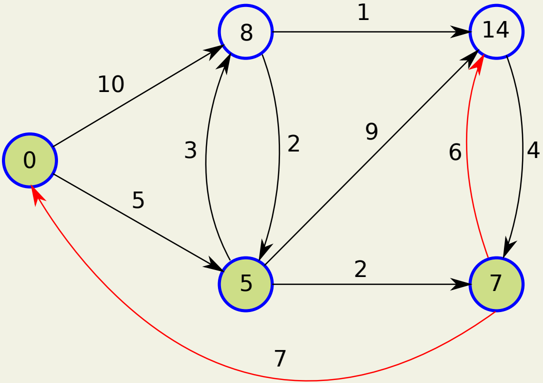
Example graph



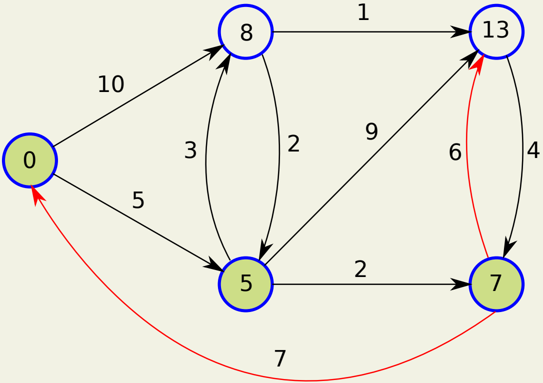
Example graph



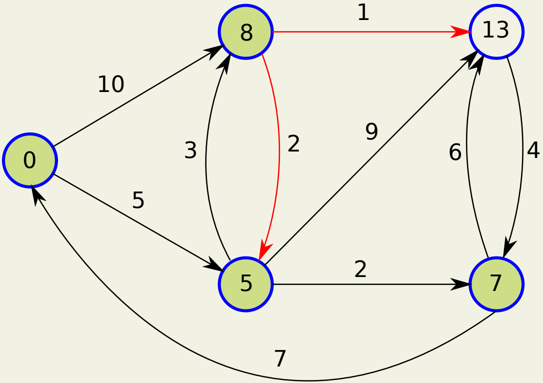
Example graph



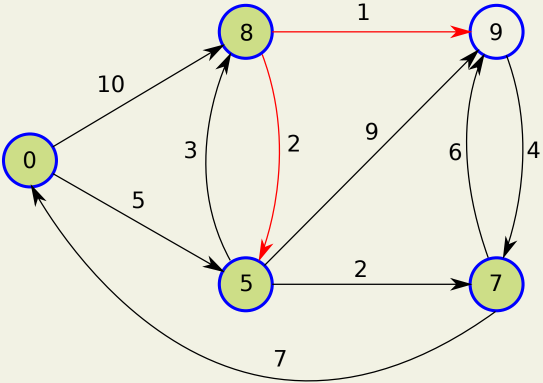
Example graph



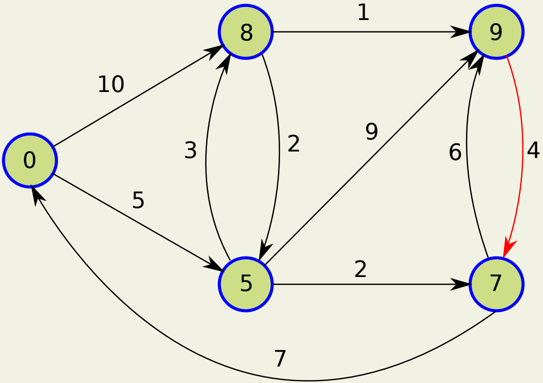
Example graph



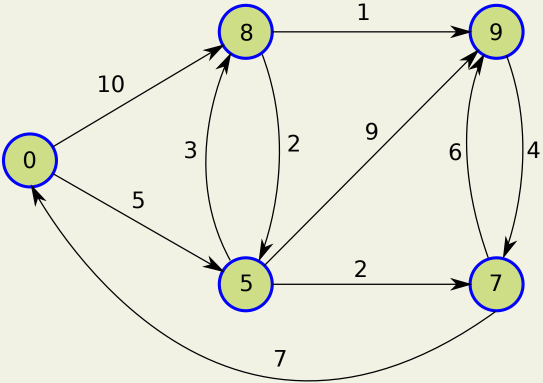
Example graph



Example graph



Example graph



Dijkstra's algorithm

"It is the algorithm for the shortest path, which I designed in about twenty minutes. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path. As I said, it was a twenty-minute invention."

-Edsger Dijkstra

Source: Wikipedia and "An Interview with Edsger W. Dijkstra". Communications of the ACM

Dijkstra's Algorithm Pseudocode

Algorithm 4 Dijkstra's algorithm

```
For all u ∈ V, d[u] ← ∞, π[u] ← NIL
d[s] ← 0
Initialize min-priority queue Q ← V
S ← ∅
while Q ≠ ∅ do
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each v ∈ N(u) do
        if d[u] + w(u, v) < d[v] then
            d[v] ← d[u] + w(u, v)
            DECREASE-KEY(v, d[v]).
            π[v] ← u
        end if
    end for
end while
```

Time Complexity of Dijkstra's

- Initialization: $O(|V|)$
- We need to do $|V|$ Extract-Min's and $|E|$ Decrease-Key's
- Depends on the implementation of the priority queue.

Time Complexity of Dijkstra's

- Initialization: $O(|V|)$
- We need to do $|V|$ Extract-Min's and $|E|$ Decrease-Key's
- Depends on the implementation of the priority queue.

- Array: Extract-Min takes $O(|V|)$ and Decrease-Key takes $O(1)$
- Heap: Extract-Min and Decrease-Key both take $O(\log |V|)$
- Fibonacci Heap: Decrease-Key takes $O(1)$ amortized time

Proof of Correctness

Theorem

At the end of Dijkstra's algorithm, we have:

$$\forall u \in V, d[u] = \delta(s, u)$$

Proof

Loop Invariant:
At the start of each iteration, we have $\forall v \in S, d[v] = \delta(s, v)$.
Init: At the start of the first iteration, $S = \emptyset$.
Maintenance: Let $u \in V$ be the first vertex for which $d[u] \neq \delta(s, u)$.
If u is not reachable from s, then $d[u] = \delta(s, u) = \infty$, so u must be reachable. Why?
If $u = s$, then the claim holds. So assume $u \neq s$.

Proof of Correctness

Take a shortest path σ from s to u.
Let y be the first vertex on σ that is outside S.
Let $x \in S$ be the vertex on σ just before y.
So the path σ looks like:

$$s \overset{\sigma_1}{\rightsquigarrow} x \to y \overset{\sigma_2}{\rightsquigarrow} u$$

Claim 1: $d[y] = \delta(s, y)$.

Proof of Correctness

$$\sigma = s \overset{\sigma_1}{\rightsquigarrow} x \to y \overset{\sigma_2}{\rightsquigarrow} u$$

Claim 1: $d[y] = \delta(s, y)$.
Since y appears before u in σ, we have $\delta(s, y) \leq \delta(s, u)$.
Claim 2: $d[u] \geq \delta(s, u)$.
Thus:

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

Although y and u were in $V \setminus S$, Extract-Min returned u.
This means $d[u] \leq d[y]$. Hence:

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

$\square$

Proof of Correctness

Claim 1

$$\sigma = s \overset{\sigma_1}{\rightsquigarrow} x \to y \overset{\sigma_2}{\rightsquigarrow} u$$

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

Proof

From loop invariant, for all vertices that were added to S before u, we computed the correct shortest distance.
So $d[x] = \delta(s, x)$.
We updated $d[y]$ when we added x to S.
Now we note a *convergence* property:
Let $s \rightsquigarrow x \to y$ be a shortest path, and $d[x] = \delta(s, x)$.
Then, relaxing the edge (x, y) sets $d[y] = \delta(s, y)$.

$\square$

Proof of Correctness

Claim 2

$d[u] \geq \delta(s, u)$

Proof

Induction on number of times d is updated after initialization.

Base case: Immediately after init, $\forall v$, $d[v] = \infty$ except $d[s] = 0$. So the claim holds.

Step: Assume claim for up to k many updates on d.

The value of $d[u]$ is updated when:

- We visit a vertex v and there exists edge (v, u).
- $d[u] > d[v] + w((v, u))$.

Proof of Correctness

Claim 2

$d[u] \geq \delta(s, u)$

Proof

Induction on number of times d is updated after initialization.

Base case: Immediately after init, $\forall v$, $d[v] = \infty$ except $d[s] = 0$. So the claim holds.

Step: Assume claim for up to k many updates on d.

The value of $d[u]$ is updated when:

- We visit a vertex v and there exists edge (v, u).
- $d[u] > d[v] + w((v, u))$.

The new $d[u] = d[v] + w((v, u))$.

The hypothesis holds for vertex v: $d[v] \geq \delta(s, v)$. So:

$$d[u] = d[v] + w((u, v)) \geq \delta(s, v) + w((u, v)) \geq \delta(s, u)$$