



Design and analysis of algorithms

Lecture 14 & 15

Edyta Szymańska

`edyta@cc.gatech.edu.pl`

Breadth-First-Search (BFS)

BFS - visits nodes in order of increasing distance from a particular node s

Breadth-First-Search (BFS)

BFS - visits nodes in order of increasing distance from a particular node s

Idea:

expanding frontier “greedy”, one edge distance at a time

Breadth-First-Search (BFS)

BFS - visits nodes in order of increasing distance from a particular node s

Idea:

expanding frontier “greedy”, one edge distance at a time
It labels each node with *the shortest distance from s* , that is the number of edges in the shortest path from s to the node.

Breadth-First-Search (BFS)

BFS - visits nodes in order of increasing distance from a particular node s

Idea:

expanding frontier “greedy”, one edge distance at a time
It labels each node with *the shortest distance from s* , that is the number of edges in the shortest path from s to the node.

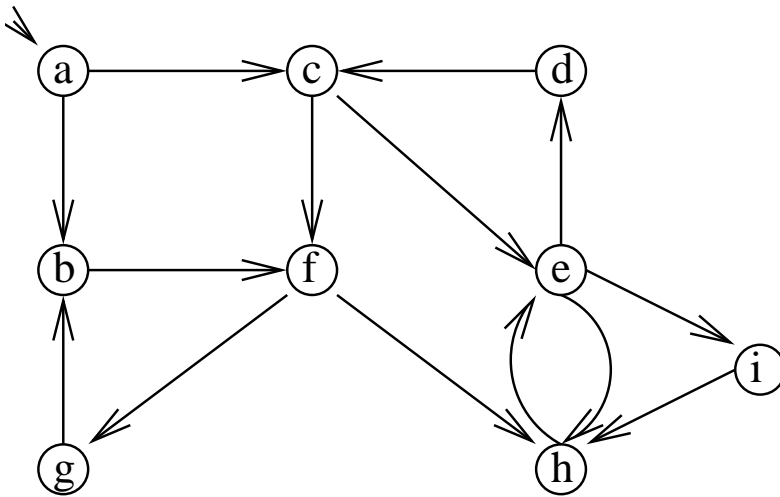
Data structure: FIFO queue (DFS - stack)

Breadth-First-Search (BFS)

```
BFS( $G = (V, E), s$ )
  initialize empty queue  $Q$ 
  for all  $u \in V - \{s\}$  do
     $d[u] := \infty$ {distance from  $s$ }
  end for
   $d[s] := 0$ 
  Insert( $s, Q$ )
  while  $Q \neq \emptyset$  do
     $u := \text{Remove}(Q)$ 
    for all  $v \in \text{Adj}[u]$  do
      if  $d[v] = \infty$  then
        Insert( $v, Q$ )
         $d[v] := d[u] + 1$ 
      end if
    end for
  end for
```

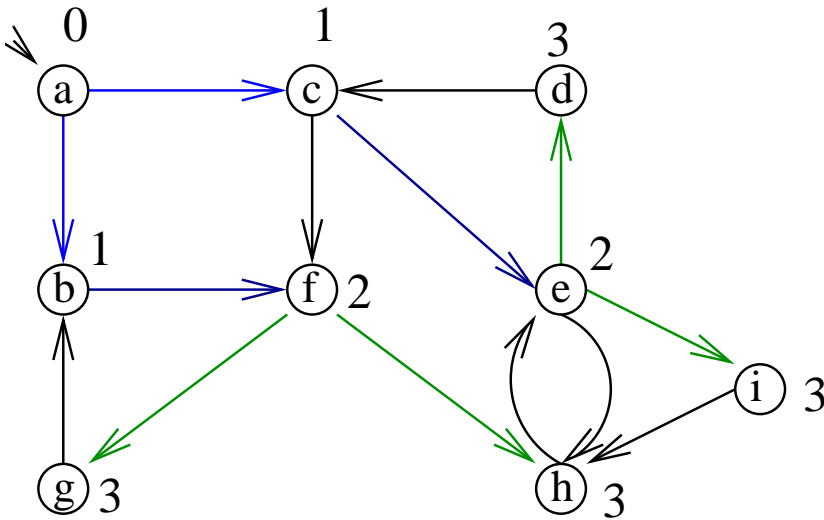
Breadth-First-Search (BFS)

Example:



Breadth-First-Search (BFS)

Example:



BFS - correctness

Proof similar to algorithm with weighted edges. We will do with Dijkstra's alg.

BFS - correctness

Proof similar to algorithm with weighted edges. We will do with Dijkstra's alg.

Note:

BFS may not reach all vertices but we are only interested in distances and do not restart BFS.

BFS - correctness

Proof similar to algorithm with weighted edges. We will do with Dijkstra's alg.

Note:

BFS may not reach all vertices but we are only interested in distances and do not restart BFS.

Time: $O(m + n)$

visits each vertex exactly once

does a constant amount of work per edge.

BFS - correctness

Proof similar to algorithm with weighted edges. We will do with Dijkstra's alg.

Note:

BFS may not reach all vertices but we are only interested in distances and do not restart BFS.

Time: $O(m + n)$

visits each vertex exactly once

does a constant amount of work per edge.

Notice that BFS can also be used to check *node-to-node connectivity* and to discover the connected component of G containing s .

BFS - applications to shortest paths

Dijkstra's algorithm: generalization of BFS to handle weighted graphs.

BFS - applications to shortest paths

Dijkstra's algorithm: generalization of BFS to handle weighted graphs.

Input:

directed graph $G = (V, E)$ and $w : E \rightarrow \mathbb{R}_+$

(BFS: $\forall e \ w(e) = 1$)

BFS - applications to shortest paths

Dijkstra's algorithm: generalization of BFS to handle weighted graphs.

Input:

directed graph $G = (V, E)$ and $w : E \rightarrow \mathbb{R}_+$

(BFS: $\forall e w(e) = 1$)

Data structure: PRIORITY QUEUE: the priority depends on our current estimate of a vertex distance from s

BFS - applications to shortest paths

Dijkstra's algorithm: generalization of BFS to handle weighted graphs.

Input:

directed graph $G = (V, E)$ and $w : E \rightarrow \mathbb{R}_+$

(BFS: $\forall e \ w(e) = 1$)

Data structure: PRIORITY QUEUE: the priority depends on our current estimate of a vertex distance from s

The algorithm will visit the vertex, which is the closest to s first.

BFS - applications to shortest paths

Dijkstra's algorithm: generalization of BFS to handle weighted graphs.

Input:

directed graph $G = (V, E)$ and $w : E \rightarrow \mathbb{R}_+$

(BFS: $\forall e w(e) = 1$)

Data structure: PRIORITY QUEUE: the priority depends on our current estimate of a vertex distance from s

The algorithm will visit the vertex, which is the closest to s first.

Dijkstra's algorithm

Dijkstra($G = (V, E, w), s$)

initialize empty queue Q

for all $v \in V - \{s\}$ **do**

$d[v] := \infty$ { distance from s }

$prev[v] := nil$ {the last vertex before v on the shortest path $s \rightsquigarrow v$ }

end for

$d[s] := 0$

Insert(s, Q)

while $Q \neq \emptyset$ **do**

$u := \text{Extract_MIN}(Q)$

 mark(u)

for all $v \in \text{Adj}[u]$ **do**

if v unmarked and $d[v] > d[u] + w(u, v)$ **then**

$d[v] := d[u] + w(u, v)$

$prev[v] := u$

 Insert(v, Q)

end if

end for

end while

Dijkstra's algorithm

Dijkstra($G = (V, E, w), s$)

initialize empty queue Q

for all $v \in V - \{s\}$ **do**

$d[v] := \infty$ { distance from s }

$prev[v] := nil$ {the last vertex before v on the shortest path $s \rightsquigarrow v$ }

end for

$d[s] := 0$

Insert(s, Q)

while $Q \neq \emptyset$ **do**

$u := \text{Extract_MIN}(Q)$

 mark(u)

for all $v \in \text{Adj}[u]$ **do**

if v unmarked and $d[v] > d[u] + w(u, v)$ **then**

$d[v] := d[u] + w(u, v)$

$prev[v] := u$

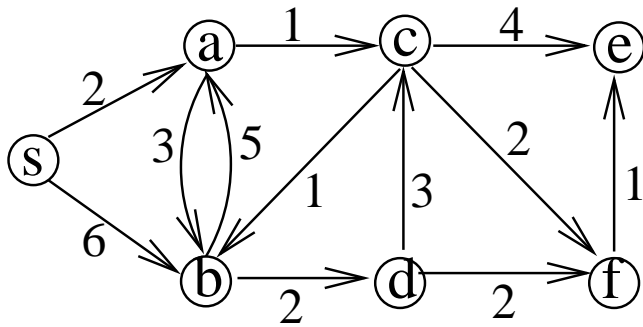
 Insert(v, Q)

end if

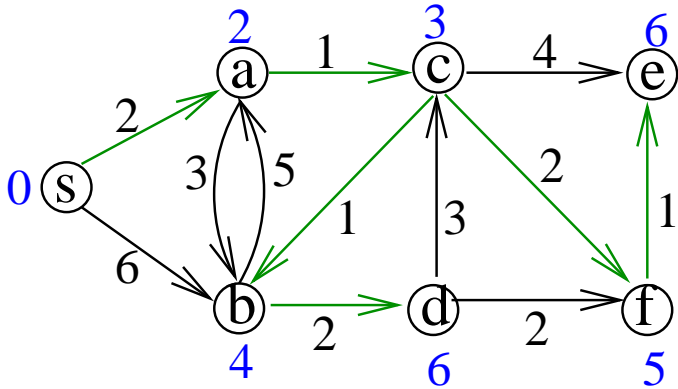
end for

end while

Dijkstra's algorithm - example



Dijkstra's algorithm - example



Dijkstra's algorithm - example



Dijkstra's algorithm - complexity

It performs:

- ⑥ *Insert* : $|E|$ times
- ⑥ *Extract_MIN* : $|V|$ times

Dijkstra's algorithm - complexity

It performs:

- ⑥ *Insert* : $|E|$ times
- ⑥ *Extract_MIN* : $|V|$ times

Q implementation

Dijkstra's algorithm - complexity

It performs:

- ⑥ *Insert* : $|E|$ times
- ⑥ *Extract_MIN* : $|V|$ times

Q implementation

<i>Q</i> implementation	Extract_MIN	Insert	Total
array	$O(n)$	$O(1)$	$O(n^2)$
binary heap	$O(\log n)$	$O(\log n)$	$O(m \log n)$
Fibonacci heap	$O(\log n)$	$O(1)$	$O(n \log n + m)$

Dijkstra's algorithm - correctness

Need to show that whenever u is marked, $d[u]$ is the shortest distance from s to u .

Dijkstra's algorithm - correctness

Need to show that whenever u is marked, $d[u]$ is the shortest distance from s to u .

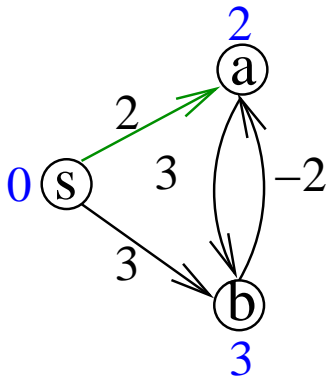
Dijkstra's algorithm



A problematic instance for Dijkstra's

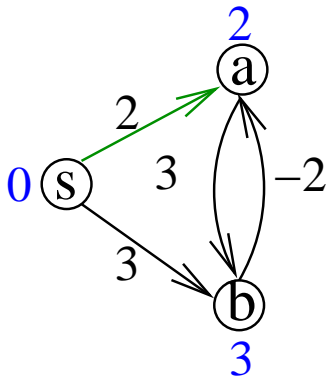
Dijkstra's algorithm

A problematic instance for Dijkstra's



Dijkstra's algorithm

A problematic instance for Dijkstra's



a new algorithm will handle this.