BFS, DFS, and the Proof of White Path Theorem

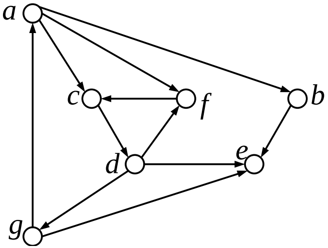
CSCI2100 Tutorial 11

Intruction

In this tutorial, we will first demonstrate BFS and DFS using concrete examples, and then prove the white path theorem.

Let's first go over the BFS algorithm through a running example on a directed graph.

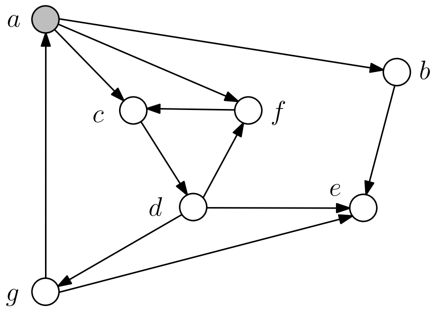
Input



Suppose we start from the vertex a, namely a is the root of BFS tree.

BFS

Firstly, set all the vertices to be white. Then, create a queue Q, en-queue the starting vertex a and color it gray. Create a BFS Tree with a as the root.

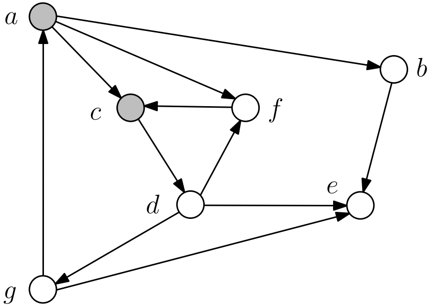


DFS Tree

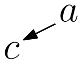
a

$Q = $ a

BFS

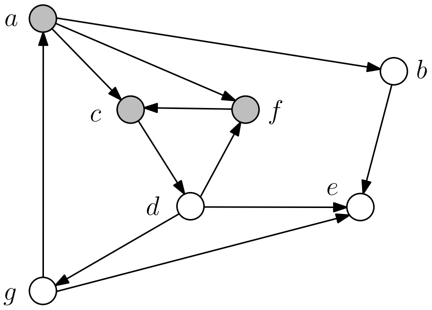


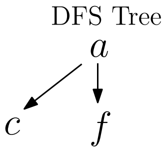
DFS Tree



$Q = \; a \;\; c$

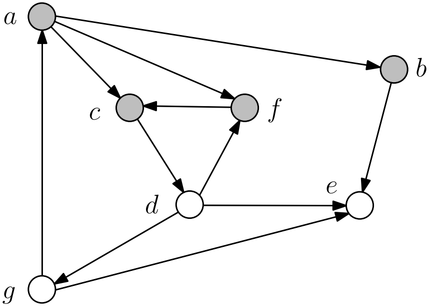
BFS

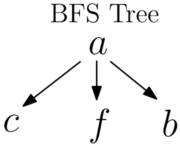




$Q = \quad a \ c \ f$

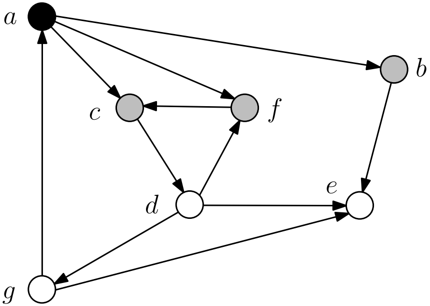
BFS



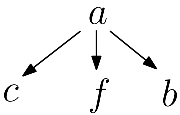


$Q = \quad a \ c \ f \ b$

BFS

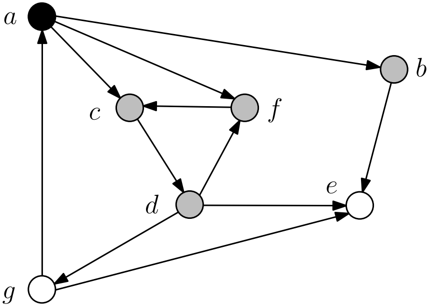


BFS Tree

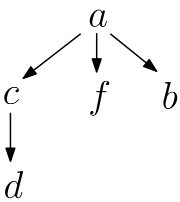


$Q =$ $c \;\; f \;\; b$

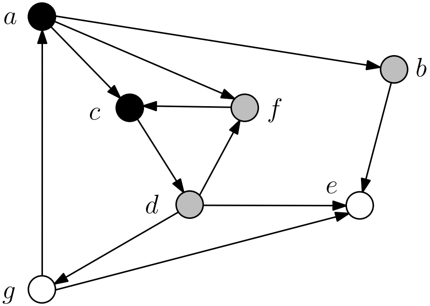
BFS



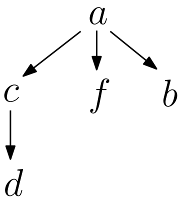
$Q = \; c \; f \; b \; d$



BFS

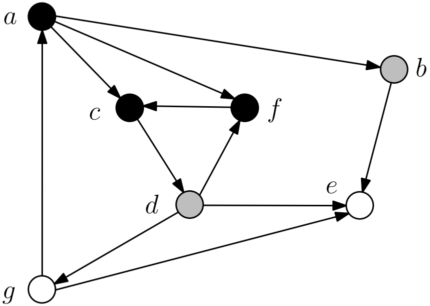


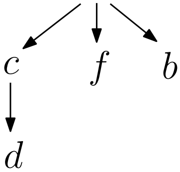
BFS Tree



$Q = \ f \ \ b \ \ d$

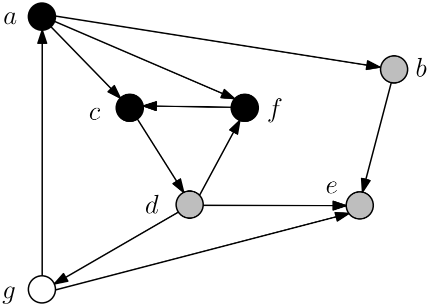
BFS

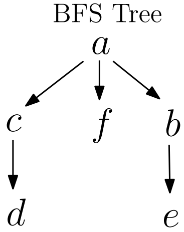




$Q =$ $b\ d$

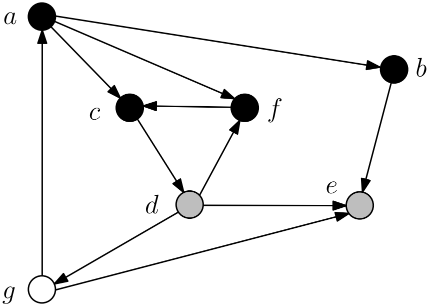
BFS

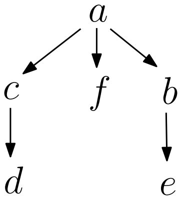




$Q = \underline{\overline{b\ d\ e\qquad\qquad}}$

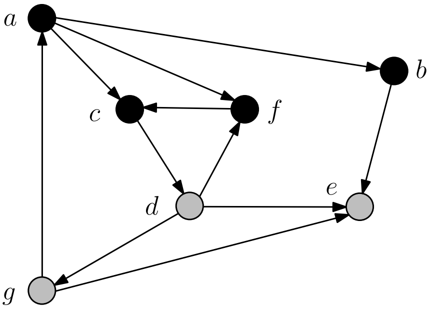
BFS

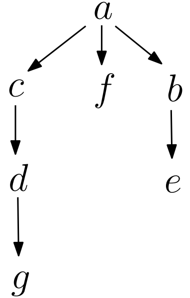




$Q = $ $d\ e$

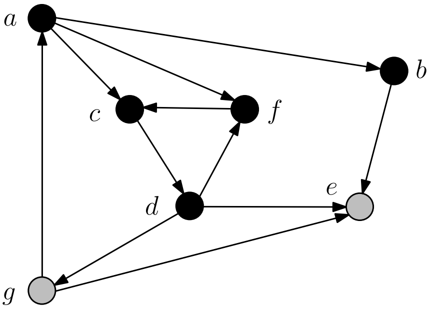
BFS

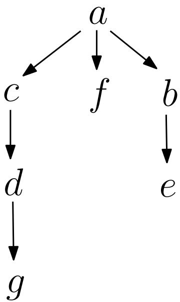




$Q = \; d \; e \; g$

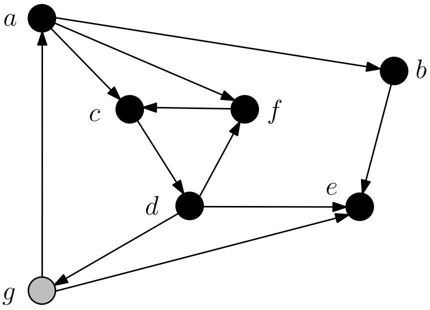
BFS

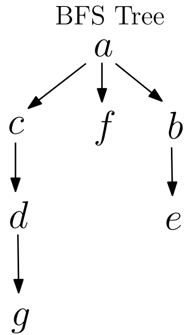




$Q =$ $e\ g$

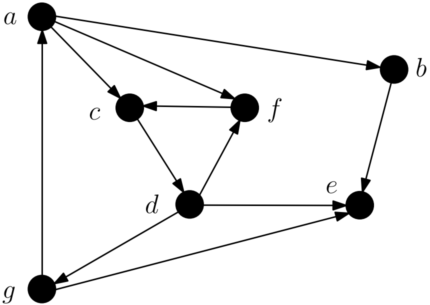
BFS

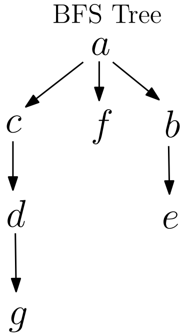




$Q = \; g$

BFS





$Q =$

Q is empty, algorithm terminated.

Single Source Shortest Path (SSSP) with Unit Weights

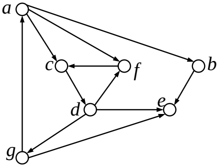
Input

A directed graph $G=(V,E)$. A vertex s in V as the starting point.

Goal

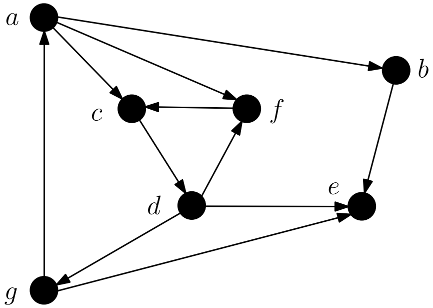
To find, for every other vertex $t \in V \setminus \{s\}$, a shortest path from s to t, unless t is unreachable from s.

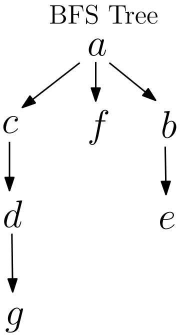
Example



a is assigned as the starting point.

First step: Do BFS on G using a as the starting point

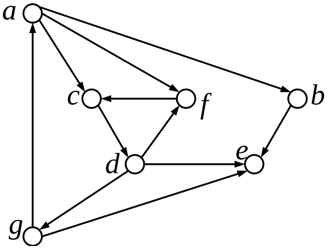




Follow the BFS Tree generated by the *BFS* algorithm, we can find the shortest paths required.

Let's first go over the DFS algorithm through a running example on a directed graph.

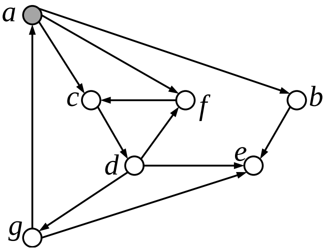
Input



Suppose we start from the vertex a, namely a is the root of DFS tree.

DFS

Firstly, set all the vertices to be white. Then, create a stack S, push the starting vertex a into S and color it gray. Create a DFS Tree with a as the root. We also maintain the time interval $I(u)$ of each vertex u.



DFS Tree

a

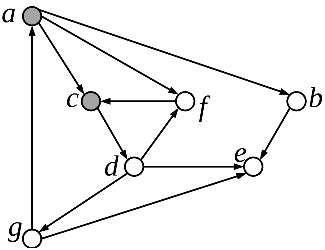
Time Interval

$I(a) = [1, \ ]$

$S = (a)$.

DFS

Top of stack: a, which has white out-neighbors b, c, f. Suppose we access c first. Push c into S.

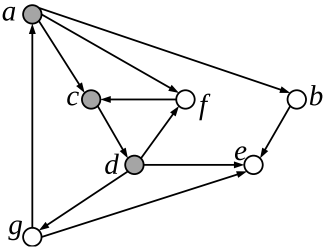


DFS Tree	Time Interval
a	$I(a) = [1,\ \ ]$
c	$I(c) = [2,\ \ ]$

$S = (a, c)$.

DFS

After pushing d into S:

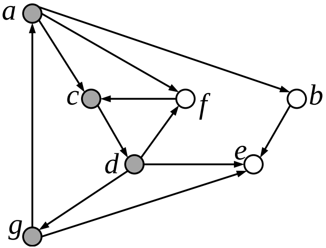


DFS Tree	Time Interval
a	$I(a) = [1,\ \ ]$
c	$I(c) = [2,\ \ ]$
d	$I(d) = [3,\ \ ]$

$S = (a, c, d)$.

DFS

Now d tops the stack. It has white out-neighbors e, f and g. Suppose we visit g first. Push g into S.

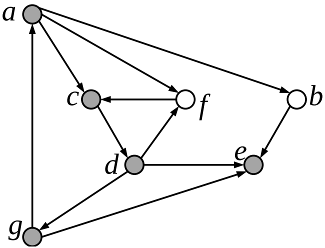


DFS Tree	Time Interval
a	$I(a) = [1,\ \ ]$
c	$I(c) = [2,\ \ ]$
d	$I(d) = [3,\ \ ]$
g	$I(g) = [4,\ \ ]$

$S = (a, c, d, g)$.

DFS

After pushing e into S:

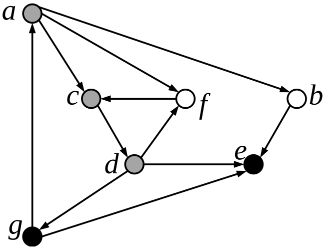


DFS Tree	Time Interval
a	$I(a) = [1, \ ]$
c	$I(c) = [2, \ ]$
d	$I(d) = [3, \ ]$
g	$I(g) = [4, \ ]$
e	$I(e) = [5, \ ]$

$S = (a, c, d, g, e)$.

DFS

e has no white out-neighbors. So pop it from S, and color it black. Similarly, g has no white out-neighbors. Pop it from S, and color it black.

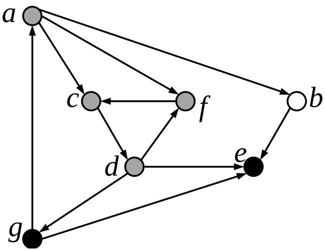


DFS Tree	Time Interval
a	$I(a) = [1,\ ]$
c	$I(c) = [2,\ ]$
d	$I(d) = [3,\ ]$
g	$I(g) = [4, 7]$
e	$I(e) = [5, 6]$

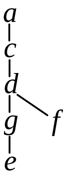
$S = (a, c, d)$.

DFS

Now d tops the stack again. It still has a white out-neighbor f. So, push f into S.



DFS Tree



Time Interval

$I(a) = [1, \ ]$

$I(c) = [2, \ ]$

$I(d) = [3, \ ]$

$I(g) = [4, 7]$

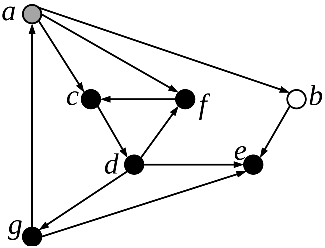
$I(e) = [5, 6]$

$I(f) = [8, \ ]$

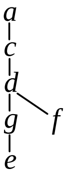
$S = (a, c, d, f)$.

DFS

After popping f, d, c:



DFS Tree



Time Interval

$I(a) = [1, \ ]$

$I(c) = [2, 11]$

$I(d) = [3, 10]$

$I(g) = [4, 7]$

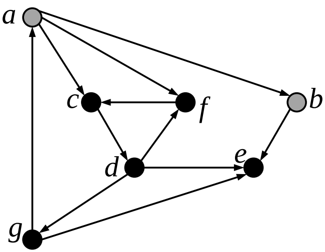
$I(e) = [5, 6]$

$I(f) = [8, 9]$

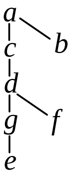
$S = (a)$.

DFS

Now a tops the stack again. It still has a white out-neighbor b. So, push b into S.



DFS Tree



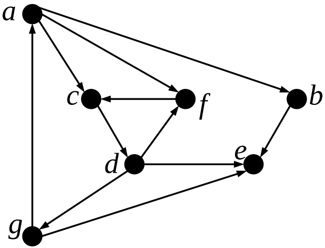
Time Interval

$I(a) = [1, \ ]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, \ ]$

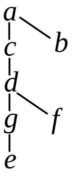
$S = (a, b)$.

DFS

After popping b and a:



DFS Tree



Time Interval

$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

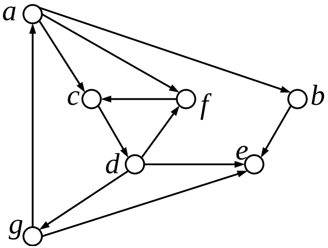
$S = ()$.

Now, there is no white vertex remaining, our algorithm terminates.

Cycle Detection

Problem Input:

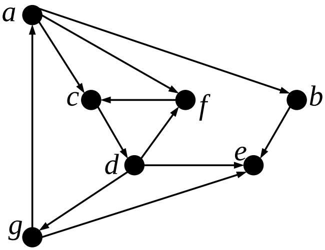
A directed graph.



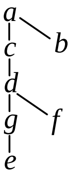
Problem Output:

A boolean indicating whether the graph contains a cycle.

First Step: DFS



DFS Tree

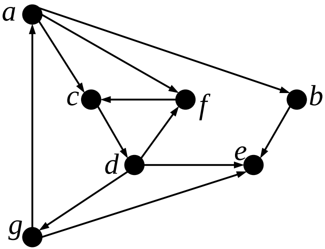


Time Interval

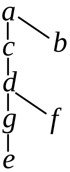
$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

Cycle Theorem: Let T be an arbitrary DFS-forest of graph G. G contains a cycle if and only if there is a back edge with respect to T.

Second Step: Try to Find Back Edge



DFS Tree



Time Interval

$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

Parenthesis Theorem: If u is a proper descendant of v in a DFS-tree of T, then $I(u)$ is contained in $I(v)$.

We proved the cycle theorem in the lecture. Recall that our proof relies on another theorem called the **white path theorem**, which we will establish in the rest of the tutorial.

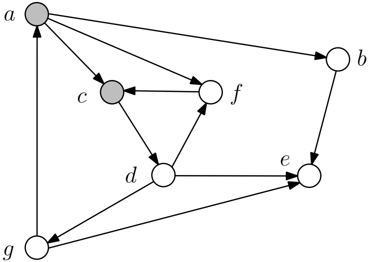
Proof of White Path Theorem

Recall:

White Path Theorem: Let u be a vertex in G. Consider the moment when u is pushed into the stack in the DFS algorithm. Then, a vertex v becomes a proper descendant of u in the DFS-forest if and only if the following is true:

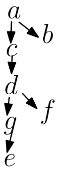
- We can go from u to v by travelling only on white vertices.

Example



DFS Tree

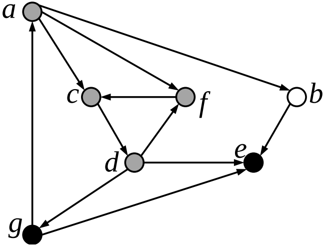
Final DFS Tree



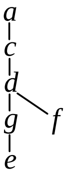
$S =$ $a \;\; c$

Lemma: Consider any vertex u in a DFS-tree. The root-to-u path in the tree is the same as the bottom-up vertex sequence in the stack at the moment when u enters the stack.

The proof is left to you.



DFS Tree



Time Interval

$I(a) = [1, \ ]$

$I(c) = [2, \ ]$

$I(d) = [3, \ ]$

$I(g) = [4, 7]$

$I(e) = [5, 6]$

$I(f) = [8, \ ]$

$S = (a, c, d, f)$.

Proof of White Path Theorem

White Path Theorem: Let u be a vertex in G. Consider the moment when u is pushed into the stack in the DFS algorithm. Then, a vertex v becomes a proper descendant of u in the DFS-forest if and only if the following is true:

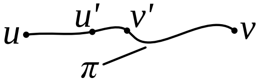
- We can go from u to v by traveling on only white vertices.

Proof: The "only-if direction" ($\Rightarrow$): Let v be a descendant of u in the DFS tree. Let π be the path from u to v in the tree. By the lemma on Slide 37, all the nodes on π entered the stack after u. Hence, π must be white at the moment when u enters the stack.

Proof of White Path Theorem

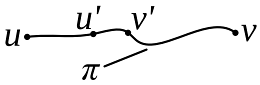
The "if direction" ($\Leftarrow$): When u enters the stack, there is a white path π from u to v. We will prove that all the vertices on π must be descendants of u in the DFS-forest.

Suppose that this is not true. Let v' be the first vertex on π — in the order from u to v — that is not a descendant of u in the DFS-forest. Clearly $v' \neq u$. Let u' be the vertex that precedes v' on π; note that u' is a descendant of u in the DFS-forest.



By the lemma on Slide 37, u' entered the stack after u.

Proof of White Path Theorem



Consider the moment when u' turns black (i.e., u' leaving the stack). Node u must remain in the stack currently (first in last out).

1. The color of v' cannot be white.

   Otherwise, v' is a white out-neighbor of u, which contradicts the fact that u' is turning black.

2. Hence, the color of v' must be gray or black.

   Recall that when u entered stack, v' was white. Therefore, v' must have been pushed into the stack while u was still in the stack. By the lemma on Slide 37, v' must be a descendant of u. This, however, contradicts the definition of v'.