Computer Science 331

Depth-First Search

Mike Jacobson

Department of Computer Science
University of Calgary

Lecture #31

Outline

Depth-First Search

Algorithm to search a graph in *depth-first* order:

- Given a graph G and a vertex s, the algorithm finds the *depth-first tree*, that is, a tree with root s whose edges are chosen by searching as deeply down a path as possible before “backtracking.”

Reference: Text, Section 12.4, beginning on p.647, describes a similar version of the algorithm that also returns the order in which vertices are discovered and the order in which processing on vertices finishes.

Idea

Problem: graphs can have *cycles* and we need to avoid following cycles (resulting in infinite loops)

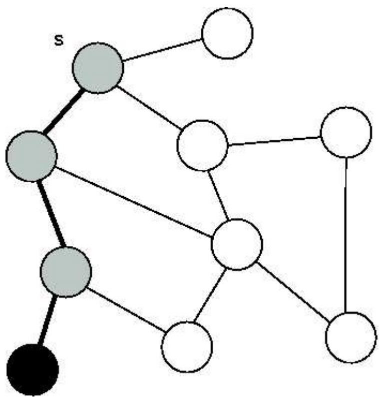
Solution: keep track of the nodes that have been visited already, so that we don’t visit them again

Details:

- initially all vertices are **white**
- carry out the following steps, beginning with node s.
  - Colour a node grey when a search from the node begins:
  - recursively search from each white neighbour (reachable by following an edge in the “forward” direction)
  - end the search by colouring the node black.

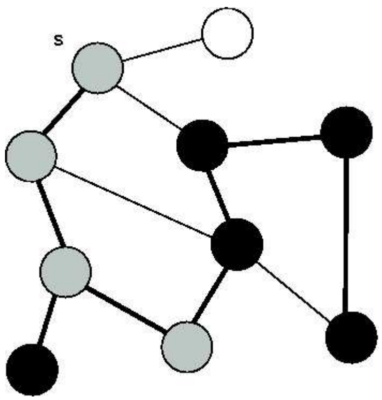
Typical Search Pattern

Pattern Near Beginning of Search:



Typical Search Pattern

Pattern Farther Along in Search:



Data and Pseudocode

The following information is maintained for each $u \in V$:

- $colour[u]$: Colour of u
- $\pi[u]$: Parent of u in tree being constructed

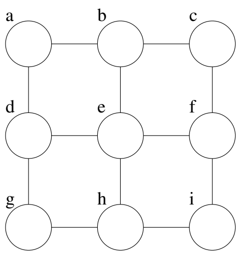
DFS(G, s)
  $\{$Initialization — all nodes initially white (undiscovered)$\}$
  for each vertex $u \in V$ **do**
    $colour[u] = \text{white}$
    $\pi[u] = \text{NIL}$
  end for
  $\{$Visit all vertices reachable from $s\}$
  DFS-Visit(s)
  return π

Pseudocode, Continued

DFS-Visit(u)
  $colour[u] = \text{grey}$
  for each $v \in Adj[u]$ **do**
    if $colour[v] == \text{white}$ **then**
      $\pi[v] = u$
      DFS-Visit(v)
    end if
  end for
  $colour[u] = \text{black}$

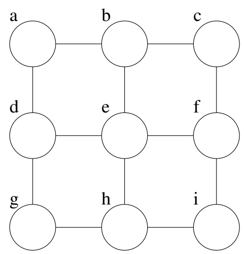
Example

Step 1



	a	b	c	d	e	f	g	h	i
π	NIL								

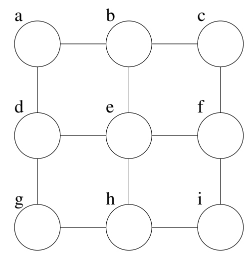
Step 2



	a	b	c	d	e	f	g	h	i
π	NIL								

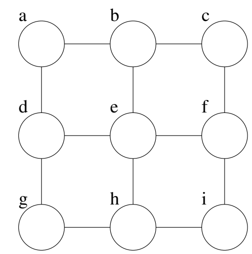
Example, continued

Step 3



	a	b	c	d	e	f	g	h	i
π	NIL								

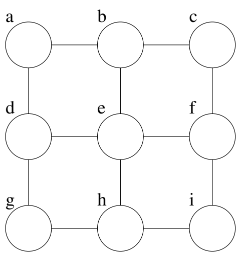
Step 4



	a	b	c	d	e	f	g	h	i
π	NIL								

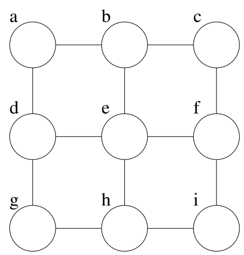
Example, continued

Step 5



	a	b	c	d	e	f	g	h	i
π	NIL								

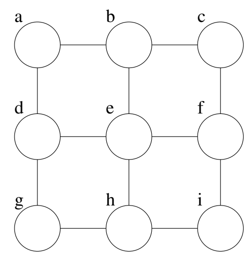
Step 6



	a	b	c	d	e	f	g	h	i
π	NIL								

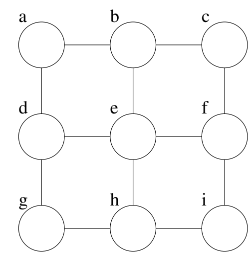
Example, continued

Step 7



	a	b	c	d	e	f	g	h	i
π	NIL								

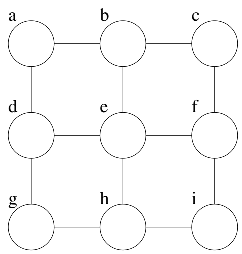
Step 8



	a	b	c	d	e	f	g	h	i
π	NIL								

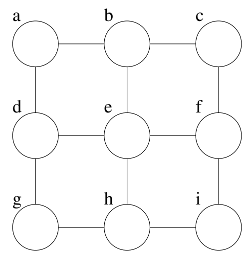
Example, continued

Step 9



	a	b	c	d	e	f	g	h	i
π	NIL								

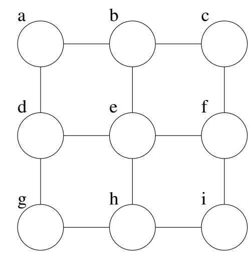
Step 10



	a	b	c	d	e	f	g	h	i
π	NIL								

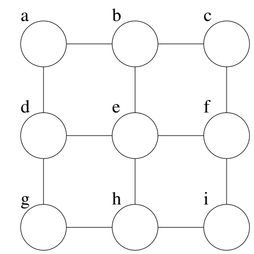
Example, continued

Step 11



	a	b	c	d	e	f	g	h	i
π	NIL								

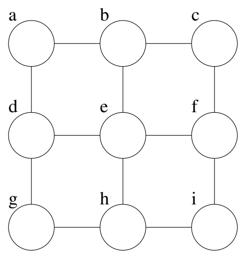
Step 12



	a	b	c	d	e	f	g	h	i
π	NIL								

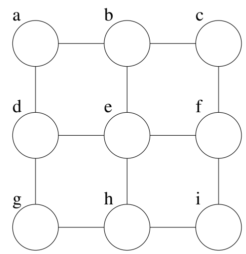
Example, continued

Step 13



	a	b	c	d	e	f	g	h	i
π	NIL								

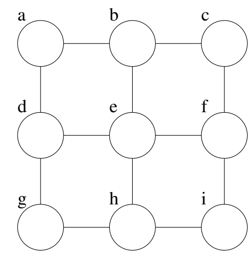
Step 14



	a	b	c	d	e	f	g	h	i
π	NIL								

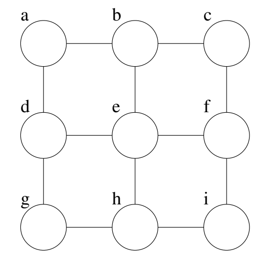
Example, continued

Step 15



	a	b	c	d	e	f	g	h	i
π	NIL								

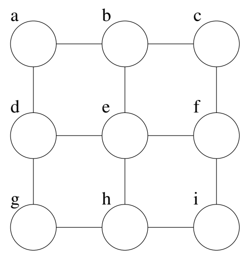
Step 16



	a	b	c	d	e	f	g	h	i
π	NIL								

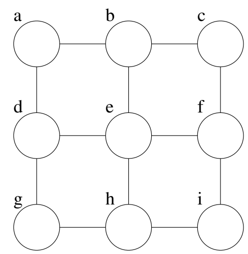
Example, continued

Step 17



	a	b	c	d	e	f	g	h	i
π	NIL								

Step 18



	a	b	c	d	e	f	g	h	i
π	NIL								

Specification of Requirements

A more formal "specification of requirements" for DFS can now be supplied.

Pre-Condition: $G = (V, E)$ is a graph and $s \in V$

Post-Condition:

- The predecessor graph $G_p = (V_p, E_p)$ corresponding to the function π and vertex s is the depth-first tree with root s.
- The graph G has not been changed.

Recall that

- $V_p = \{s\} \cup \{v \in V \mid \pi[v] \neq \text{NIL}\}$
- $E_p = \{(\pi[v], v) \mid v \in V \text{ and } \pi[v] \neq \text{NIL}\}$

Behaviour of DFS-Visit

Let $u \in V$.

- If DFS-Visit is ever called with input u then $colour[u] = \text{white}$ immediately *before* this function is called with this input, and $colour[u] = \text{black}$ on termination, if this function terminates.

The following notation will be useful when discussing properties of this algorithm.

- Consider the *colour* function just **before** DFS-Visit is called with input u. Let
  - $V_u = \{v \in V \mid colour[v] = \text{white}\}$,
  - $G_u = (V_u, E_u)$ be the induced subgraph of G corresponding to the subset V_u.

Behaviour of DFS-Visit

Additional Useful Notation:

- Consider the function π immediately **after** this call to DFS-Visit terminates (if it terminates at all).
  - Let $\pi_u : V_u \to V_u \cup \{\text{NIL}\}$ such that, for a node $v \in V_u$,

$$\pi_u(v) = \begin{cases} \pi(v) & \text{if } v \neq u, \\ \text{NIL} & \text{if } v = u. \end{cases}$$

  - Let $G_{p,u} = (V_{p,u}, E_{p,u})$ be the predecessor subgraph of G_u corresponding to the function π_u and the vertex u.

Behaviour of DFS-Visit

Theorem 1

Suppose that this execution of DFS-Visit terminates. Then

- $G_{p,u}$ *is a depth-first tree for the graph* G_u *and the vertex* u*.*
- *The graph* G *has not been changed by this execution of DFS-Visit.*
- *If* $v \in V_u$ *then* $colour[v] = black$ *if* $v \in V_{p,u}$*, and* $colour[v] = white$ *otherwise, immediately after termination*
- *If* $v \in V$ *but* $v \notin V_u$ *then neither* $colour[v]$ *nor* $\pi[v]$ *have been changed by this execution of DFS-Visit.*
- $\pi[u]$ *has not been changed by this execution of DFS-Visit.*

Method of proof.

Induction on $|V_u|$. □

Partial Correctness of DFS

Theorem 2

If DFS is executed with an input graph G *and vertex* $s \in G$ *(so that the given pre-condition is satisfied) then either the post-condition is satisfied on termination of this algorithm or the algorithm does not terminate at all.*

Method of Proof.

Notice that this follows by inspection of the code, using the result about DFS-Visit that has just been established. □

Running Time

Theorem 3

Suppose $G = (V, E)$ *is a directed or undirected graph, and suppose DFS is run on* G *and a vertex* $v \in S$*. Then the algorithm terminates after* $O(|V| + |E|)$ *operations.*

Sketch of Proof.

□

Iteration in Depth-First Order

Some applications require that the vertices in a graph that are reachable from a vertex s be accessed in “depth-first” order.

To list the nodes in this order, modify the given algorithms as follows:

- Delete references to the array π (this is no longer needed)
- Visit a node as soon as it is coloured **grey**

The worst-case cost is in $\Theta(|V| + |E|)$ once again.