Computer Science 331
Breadth-First Search

Mike Jacobson

Department of Computer Science
University of Calgary

Lecture #32

Outline

Breadth-First Search

Algorithm to search a graph in *breadth-first order*

- visit all neighbours of a node before going deeper

Given a graph G and *source vertex* s, the algorithm finds

- the vertices that are reachable from s by following edges (in their "forward" direction if the graph is directed)
- the distance (number of edges) of each of these vertices from s
- a shortest path from s to each vertex
- a *tree* with root s including vertices reachable from s

Idea

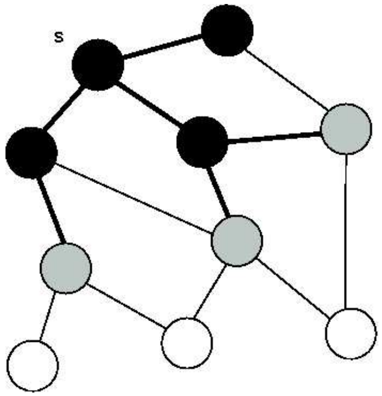
Begin with s; expand the boundary between "discovered" and "undiscovered" vertices uniformly across the breadth of the boundary

As in DFS, Vertices are coloured during the search

- All vertices are initially **white**, s is almost immediately coloured **grey**.
- All white vertices are "undiscovered."
- "Discovered" vertices are either grey or black. Vertices on the boundary between discovered and undiscovered vertices are **grey**. Other discovered vertices are **black**.

Unlike DFS, when a grey vertex t is processed, all white neighbours are recoloured grey; t is then coloured black.

Typical Search Pattern



Data and Data Structures

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

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

In order to ensure that the search is performed in a “breadth-first” way, a **queue** is used to store grey nodes

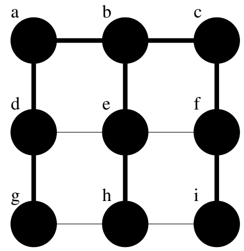
Pseudocode

BFS(G, s)
{Initialization}
for each vertex $u \in V$ **do**
  $colour[u] = \text{white}$ {mark all vertices as undiscovered}
  $d[u] = +\infty$
  $\pi[u] = \text{NIL}$
end for
$colour[s] = \text{grey}$ {start with source vertex s}
$d[s] = 0$ {path from s to itself has distance 0}
$\pi[s] = \text{NIL}$ {s is the root of the BFS tree (no parent)}
Initialize queue Q to be empty
enqueue(Q, s) {add first grey node s to the queue}

Pseudocode, Continued

while (Q is not empty) **do**
  $u = \text{dequeue}(Q)$
  for each $v \in Adj[u]$ **do**
    {examine neighbours of u}
    if $colour[v] == \text{white}$ **then**
      $colour[v] = \text{grey}$ {discover each undiscovered neighbour}
      $d[v] = d[u] + 1$ {shortest path: s to u followed by (u, v)}
      $\pi[v] = u$ {u is the predecessor on the shortest path}
      enqueue(Q, v) {examine neighbours of v}
    end if
  end for
  $colour[u] = \text{black}$ {all neighbours of u have been discovered}
end while
return π, d

Example



Q $\square$

	a	b	c	d	e	f	g	h	i
d	0	1	2	1	2	3	2	3	4
π	NIL	a	b	a	b	c	d	e	f

Partial Correctness of Breadth-First Search

The *shortest-path distance* $\delta(s, v)$ from s to v is the minimum number of edges on a path from s to v.

Theorem 1

Let $G = (V, E)$ be a directed or undirected graph, and suppose BFS is run on G from a given source vertex $s \in V$. Then each of the following properties is satisfied on termination of the algorithm (if it terminates):

- *The predecessor subgraph $G_p = (V_p, E_p)$ for the function π and vertex s is a tree containing all of (and only those) vertices that are reachable from s in G.*
- *For all $v \in V$, $d[v]$ is the length of a shortest path from s to v in G, and $d[v] = +\infty$ if and only if v is not reachable from s.*
- *For every $v \in V$ that is reachable from s, the path from s to v in G_p is also a shortest path from s to v in G.*

Useful Property of Distances

Lemma 2

Let $G = (V, E)$ be a directed or undirected graph, and let $s \in V$ be an arbitrary vertex. Then, for every edge $(u, v) \in E$, $\delta(s, v) \leq \delta(s, u) + 1$.

Proof.

If u is reachable from s :

- one path from s to v : shortest path to u followed by edge (u, v)
- shortest path to v is at most as long as this path ($\delta(s, u) + 1$)

Otherwise, $\delta(s, u) = \infty$ and the inequality holds. $\square$

Lemma: Distance Inequality

Lemma 3

Let $G = (V, E)$ be a directed or undirected graph, and suppose BFS is run on G from a given source vertex $s \in V$. Then, if BFS terminates, for each vertex $v \in V$, the value $d[v]$ calculated by the algorithm satisfies the inequality $d[v] \geq \delta(s, v)$.

Proof: induction on the number of enqueue operations

Proof of Distance Inequality

Proof.

Base case (s is enqueued):

- $d[s] = \delta(s, s) = 0$, and $d[v] = \infty \geq \delta(s, v)$ for all $v \in V - \{s\}$

Inductive step (white vertex v discovered during the search from u):

- $d[u] \geq \delta(s, u)$ by inductive hypothesis
- algorithm sets $d[v] = d[u] + 1 \geq \delta(s, u) + 1$
- thus, by Lemma 2, $d[v] \geq \delta(s, v)$
- v is coloured grey and enqueued
- v is never enqueued again (only new grey nodes are enqueued)

$d[v]$ never changes again (inductive hypothesis is maintained) □

Lemma: Enqueued Vertices

Lemma 4

Suppose that during the execution of BFS on a graph $G = (V, E)$, the queue Q contains vertices $\langle v_1, v_2, \ldots, v_r \rangle$, where v_1 is the head of Q and v_r is the tail of Q. Then $d[v_r] \leq d[v_1] + 1$ and $d[v_i] \leq d[v_{i+1}]$ for $1 \leq 2 \leq r - 1$.

Interpretation of Lemma:

- second inequalities: d values of vertices in Q are increasing
- first inequality: there are at most two distinct d values for all vertices in Q

Proof: induction on the number of queue operations

Proof (Enqueued Vertices Lemma)

Proof.

Base case (Q contains only s) holds trivially

Inductive step (Lemma holds after enqueuing or dequeing a vertex):

1. if v_1 is dequeued, v_2 becomes the new head
   - by the inductive hypothesis $d[v_1] \leq d[v_2]$
   - thus $d[v_r] \leq d[v_1] + 1 \leq d[v_2] + 1$
2. if v_{r+1} is enqueued
   - vertex u previously removed, so $d[v_1] \geq d[u]$ by hypothesis
   - thus $d[v_{r+1}] = d[u] + 1 \leq d[v_1] + 1$
   - by hypothesis $d[v_r] \leq d[u] + 1$, so $d[v_r] \leq d[v_{r+1}]$

Thus, after either operation, the lemma holds. □

Lemma: Distance and Queue Order

Lemma 5

Suppose that vertices v_i and v_j are enqueued during the execution of BFS, and that v_i is enqueued before v_j. Then $d[v_i] \leq d[v_j]$ at the time v_i is enqueued.

Proof.

Follows from Lemma 4, and the fact that each vertex only receives a finite d value once. □

Lemma: Correctness of Distance

Lemma 6

If a vertex v is enqueued at any point during the execution of the algorithm, then v is reachable from s. Furthermore, the value $d[v]$ that is set immediately before v is enqueued is equal to $\delta(s, v)$.

See handout for complete proof.

Outline of Proof (Correctness of Distance Lemma)

Assume v is the vertex with smallest $\delta(s, v)$ value for which $d[v]$ is incorrect

- By Lemma 3 $d[v] \geq \delta(s, v) \Rightarrow d[v] > \delta(s, v)$

Suppose u precedes v on the shortest path from s to v.

- $\delta(s, u) < \delta(s, v)$, so by our choice of v we have $d[u] = \delta(s, u)$
- Thus $d[v] > \delta(s, v) = \delta(s, u) + 1 = d[u] + 1$

Proceed by arguing that when u is dequeued, the inequality

$$d[v] > d[u] + 1$$

is violated.

Proof Outline (continued)

When u is dequeued, its neighbour v is either white, grey, or black:

1. if white, then algorithm sets $d[v] = d[u] + 1$ (contradiction)
2. if black, v was already removed from the queue and by Lemma 5 $d[v] \leq d[u]$ (contradiction)
3. if grey, v was coloured after removing another vertex w before u :
   - $d[v] = d[w] + 1$, but by Lemma 5 $d[w] \leq d[u]$
   - thus $d[v] \leq d[u] + 1$ (contradiction)

In all three cases, we have a contradiction to the inequality $d[v] > d[u] + 1$

Thus, we must have $d[v] = \delta(s, v)$ as required □

Lemma: Completeness of Predecessor Subgraph

Lemma 7

Suppose the BFS algorithm is run with a graph $G = (V, E)$ and vertex $s \in V$ as input. If the algorithm terminates then, on termination, the predecessor subgraph for the function π and vertex s includes all of the vertices in G (and, only those vertices) that are reachable from s.

Proof.

- By Lemma 6, all $v \in V$ that are enqueued are reachable from s
- Algorithm sets $\pi[v] = u$ when v is enqueued.
- Thus, all vertices in the predecessor subgraph ($\pi[v] \neq$ NIL) are reachable from s. □

Proof of Theorem 1 (partial correctness of BFS)

Proof.

First point follows from Lemma 7.

Second point follows from Lemma 6

The third point holds because:

- if $\pi[v] = u$, then $d[v] = d[u] + 1$ (from the pseudocode)
- shortest path from s to $\pi[v]$ followed by the edge $(\pi[v], v)$ has minimal length $d[u] + 1$ □

Efficiency

Theorem 8

Let $G = (V, E)$ be a directed or undirected graph, and suppose BFS is run on G from a given source vertex $s \in V$. Then the algorithm terminates after performing $O(|V| + |E|)$ operations.

Proof.

- Each vertex is enqueued and dequeued at most once — $O(|V|)$
- Adjacency list of each vertex is scanned once — total for all vertices is $\Theta(|E|)$
- cost of initialization is $O(|V|)$

Thus, total cost is $O(|V| + |E|)$. □

References

Text, Section 12.4: A similar version of the algorithm that does not compute and return the distances of vertices from the input node.

Introduction to Algorithms, Section 22.3: More details about the version of the algorithm presented here.