Breadth first search

1. **Background**
2. **Data structures**
3. **Procedure**
4. **Results and interpretation**

1. Background

At start: we have a digraph *G = (V,E)* and a starting node (source node) *s* from the digraph.

Goal: we want to know all nodes that are reachable from *s*.

One way to do this is by performing a graph search.

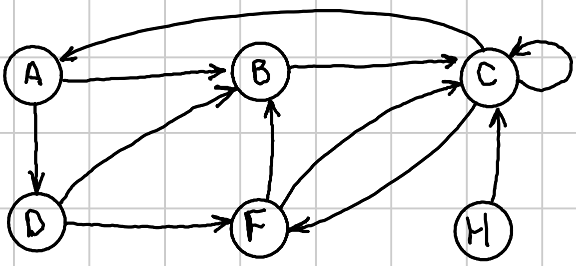
If all other nodes are reachable, then the search is also a traversal.

Reminders

- y is reachable from x when there is a directed path from x to y
- y is adjacent to x when edge $\cancel{(a,b)}$ (x,y) exists
- a path's length is the number of edges in the path

New: The distance $\delta(x, y)$ between from x to y is is the length of shortest path from x to y.

Example



x	A	B	C	D	F	H
adjacent to x	D, B	C	C, A, F	B, F	B, C	C
reachable from x	D, B, C, A, F	C, A, D, B, F	C, A, F, B, D	B, F, C, A, D	B, C, A, D, F	C, A, F, D, B

paths from D to C	(D, F, B, C)	(D, B, C)	(D, F, C)
length	3	2	2

$\delta(D, C) = 2$ $\square$

Results from Breadth-first-search (BFS):

- all reachable nodes from s
- the distance $\delta(s, y)$ of all reachable nodes from s
- a rooted tree whose root is s which includes all reachable nodes from s (the BF-tree)

2. Data structures

In a graph search a node is either discovered or undiscovered.

Q: What do we mean when we say a node x has been discovered?

A: We mean that a path from s to x has been found.

In any graph search

- start with all nodes undiscovered, except *s*
- progress is made by moving along edges and discovering nodes that have been undiscovered
- graph searches differ in the order in which they move along edges

In BFS the status of a node is monitored by assigning it a color:

- a white node is undiscovered
- a gray node has been discovered, but it may have adjacent nodes that are undiscovered (white)
- a black node has been discovered and all nodes adjacent to it have been discovered (either gray or black)

Note: progression of a node's color: white $\rightarrow$ gray $\rightarrow$ black

To perform BFS, for each node we have the following attributes:

- $x.d$ distance of node x from source s: $x.d = \delta(s, x)$
- $x.colour$ = color of node
- $x.\pi$ = parent of node x in BF tree
- $x.Adj$ set containing nodes that are adjacent to x

In BFS we maintain a queue of gray nodes.

A queue is a one-dimensional data structure that has two ends: the head (front end) and the tail (back end).

Let Q be a queue. There are two basic operations:

- ENQUEUE(Q, x): item x is put into Q at the tail
- DEQUEUE(Q): the item at the head of Q is removed and returned

A queue is said to function on a first-in-first-out (FIFO) basis.

Example

Start with empty queue: Q: tail $\boxed{}$ head

operation	queue
ENQUEUE(Q, 7)	tail $\boxed{7}$ head
ENQUEUE(Q, 2)	tail $\boxed{2}\boxed{7}$ head
DEQUEUE(Q)	tail $\boxed{2}$ head
ENQUEUE(Q, 4)	tail $\boxed{4}\boxed{2}$ head

□

3. Procedure

Description of BFS:

BREADTH–FIRST–SEARCH

1. Mark s as discovered. Mark the distance as $d = 1$.

2. Discover all nodes that are a distance of d from s.

3. Add 1 to d.

4. If there are still nodes that are reachable from s, then repeat steps 2 and 3.

Q: How do we know that there are still nodes that are reachable from *s*?

Pseudocode:

```
BREADTH–FIRST–SEARCH(s, G)
executes a breadth first search on graph G starting from
source node s

forEach node x in G
  x.color = white, x.d = ∞, x.π = NIL
end

/* Give source node appropriate values. */
s.color = gray, s.d = 0
▷ initialize a queue in Q
ENQUEUE(Q, s)
while Q is not empty
  x = DEQUEUE(Q)

  forEach node y in x.Adj
     if y.color == white then
        /* Node y is undiscovered. */
        y.color = gray, y.d = x.d + 1, y.π = x
        ENQUEUE(Q, y)
     end
  end
  x.color = black
end
```

Remarks

1. The entire graph *G* is given as an argument. This is intended to represent the set of vertices *V* and the adjacency sets *x.Adj* for each node.

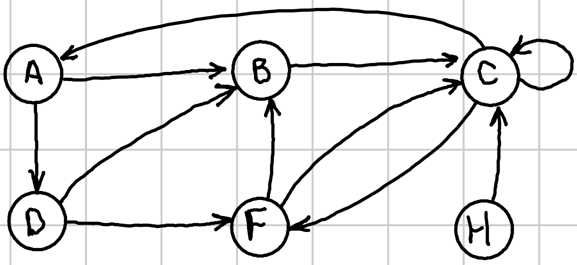
2. Initialization is done in forEach loop of line 5.

3. At line 12, *s* is the only element in *Q*. Hence in the first iteration of the while loop, *x* = *s*.

4. In each iteration of the while loop
   - *x* is removed from *Q*
   - all nodes adjacent to *x* are investigated in the forEach loop of line 16
   - *x* is eventually colored black

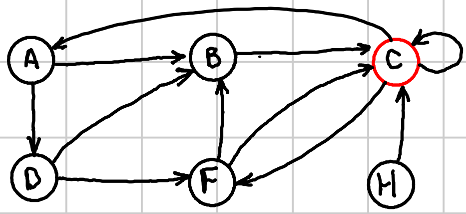
5. A node can only be added once to *Q*.

Example



Execute BREADTH-FIRST_SEARCH with *s* = *D*

item	while-loop iteration 0	1	2	3	4
Q	D	F B	C F	C	A
A.d	∞	∞	∞	∞	3
B.d	∞	1	1	1	1
C.d	∞	∞	2	2	2
D.d	0	0	0	0	0
F.d	∞	1	1	1	1
H.d	∞	∞	∞	∞	∞
A.π	NIL	NIL	NIL	NIL	C
B.π	NIL	D	D	D	D
C.π	NIL	NIL	B	B	B
D.π	NIL	NIL	NIL	NIL	NIL
F.π	NIL	D	D	D	D
H.π	NIL	NIL	NIL	NIL	NIL
A.color	white	white	white	white	gray
B.color	white	gray	black	black	black
C.color	white	white	gray	gray	black
D.color	gray	black	black	black	black
F.color	white	gray	gray	black	black
H.color	white	white	white	white	white



Final results:

x	A	B	C	D	F	H
$x.d$	3	1	2	0	1	∞
$x.\pi$	C	D	B	NIL	D	NIL
$x.color$	black	black	black	black	black	white

□

4. Results and interpretation

Q: How do we know that $x.d$ is the length of the shortest path from s to x, $\delta(s, x)$?

A: The nodes are put into the queue Q in increasing order of the their distance.

From example:

Q:	A	C	F	B	D
x.d	3	2	1	1	0

Q: How do we know what nodes are reachable from s?

A1: If x is reachable from s then $x.color$ is not white.

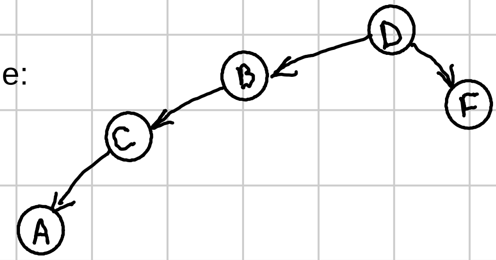
A2: If x is reachable from s then $x.d$ is not ∞.

Q: How can we construct the BF-tree?

A: The root is *s*. For each node *x*, we can construct the reverse path (from *x* to *s*) using parent attribute $x.\pi$.

x	A	B	C	D	F	H
$x.\pi$	C	D	B	NIL	D	NIL

BF-tree from example:



Remark: The paths from *s* to *x* is the shortest length path. (There may be other paths that are as short, but none that are shorter.)

Q: Can we use Breadth-first-search on an undirected graph?

A: Yes.

Remarks for undirected graph:
- if *x* belongs to *y.Adj*, then *y* belongs to *x.Adj*
- if *x* is reachable from *s*, then *s* is reachable from *x*
- *x.d* is the distance both from *s* to *x* and from *x* to *s*