CS213/293 Data Structure and Algorithms 2023

Lecture 16: Graphs - Breadth-first search

Instructor: Ashutosh Gupta

IITB India

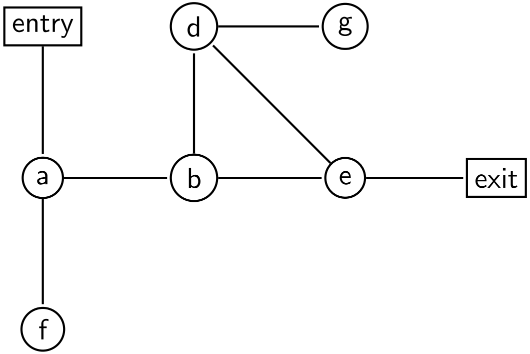
Compile date: 2023-10-25

Topic 16.1

Breadth-first search (BFS)

Solving a maze

What is a good way of solving a maze?



- Every choice point is a vertex
- Paths connecting the points are edges
- Problem: find exit node

Breadth-first search

Definition 16.1

A breadth-first search(BFS) traverses a connected component in the following order.

- *BFS starts at a vertex, which is at level 0.*
- *BFS traverses the unvisited adjacent vertices of level $n-1$ vertices, which are the vertices at level n.*

The above traversal defines a spanning tree of the graph.

In the algorithm, we need to keep track of the already visited vertices and visit vertices at lower level first.

Algorithm: BFS for search

Algorithm 16.1: BFS(Graph $G = (V, E)$, vertex r, Value x)

```
Queue Q;
set $visited := \{r\}$;
$Q.enqueue(r)$;
while not $Q.empty()$ do
    $v := Q.dequeue()$;
    if $v.label == x$ then
        return $v$
    for $w \in G.adjacent(v)$ do
        if $w \notin visited$ then
            $visited := visited \cup \{w\}$;
            $Q.enqueue(w)$
```

A vertex can be in three possible states

- Not visited
- Visited and in queue
- Visited and not in queue

Exercise 16.1

How do we maintain visited set?

Example : BFS

Initially: $Q = [entry]$

After visiting entry: $Q = [a]$

After visiting a: $Q = [f, b]$

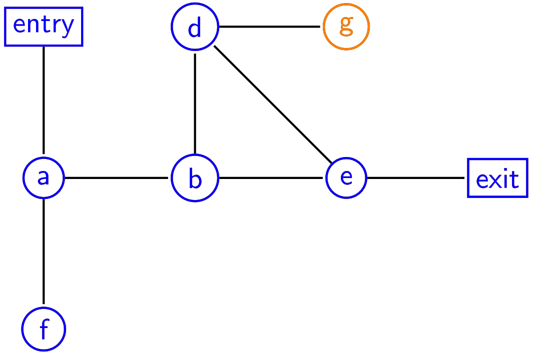
After visiting f: $Q = [b]$

After visiting b: $Q = [e, d]$

After visiting e: $Q = [d, exit]$

After visiting d: $Q = [exit, g]$

After visiting exit: return is triggered



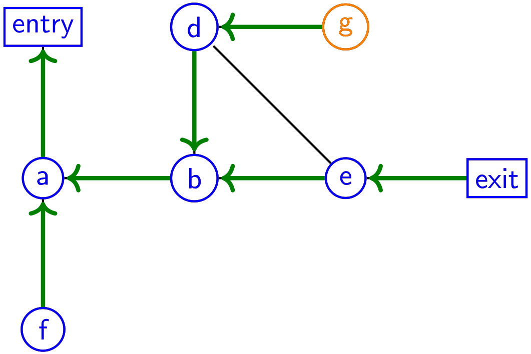
We not only want to find the exit node but also the path to the exit node.

Algorithm: BFS for path to the found node

Algorithm 16.2: BFS(Graph $G = (V, E)$, vertex r, Value x)

```
Queue Q;
$visited := \{r\}$;
$Q.enqueue(r)$;
while $not\ Q.empty()$ do
    $v := Q.dequeue()$;
    if $v.label == x$ then
        return $v$
    for $w \in G.adjacent(v)$ do
        if $w \notin visited$ then
            $visited := visited \cup \{w\}$;
            $Q.enqueue(w)$;
            $w.parent = v$
```

Example : BFS with parent relation



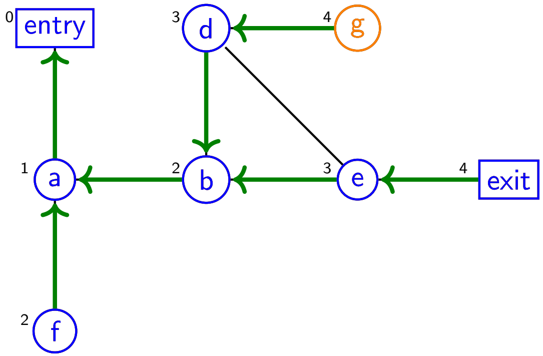
Algorithm: BFS for rooted spanning tree

We do not stop at some node but traverse entire graph.

Algorithm 16.3: BFSSPANNING(Graph $G = (V, E)$, vertex r)

```
Queue Q;
$visited := \{r\}$;
$Q.enqueue(r)$;
$v.level := 0$;
while not $Q.empty()$ do
    $v := Q.dequeue()$;
    for $w \in G.adjacent(v)$ do
        if $w \notin visited$ then
            $visited := visited \cup \{w\}$;
            $Q.enqueue(w)$;
            $w.parent := v$;
            $w.level := v.level + 1$
```

Example : Spanning tree from BFS



Topic 16.2

Analysis of BFS

Running time of BFS

- For Each node there is an enqueue and a dequeue. Therefore, $O(|V|)$ queue operations.
- For each node adjacent nodes are enumerated. Therefore, the inner loop will have $O(|E|)$ iterations.

Therefore, the running time is $O(|V| + |E|)$

Pattern in content of Q

Theorem 16.1

Q will have nodes with level sequence $k...k\underbrace{(k+1)...(k+1)}_{\text{possibly empty}}$ for some k.

Proof.

We prove it by induction.

Base case:

Initially, Q has level sequence 0.

Induction step:

Let us suppose at a given time the level sequence is $k...k\underbrace{(k+1)...(k+1)}$.

We will dequeue a k and and possibly add $k+1$. Hence proved. □

Level difference of adjacent nodes.

Theorem 16.2

For any edge $\{v, v'\} \in E$, $|v.level - v'.level| \leq 1$.

Proof.

Let us suppose v was added to visited set before v'.

Due to the previous theorem, Vertices will enter Q with increasing level.

v will receive a level on entering Q.

We have two possible cases.

- v' entered Q at the iteration for dequeue of v. Therefore, $v'.level = v.level + 1$.
- v' entered Q before dequeue of v. $v'.level$ must be either $v.level + 1$ or $v.level$. (Why?)

Commentary: In the last case, v is in Q when v is entering, therefore $v'.level$ cannot enter Q with value greater than $v.level + 1$.

BFS finds shortest path

Theorem 16.3

For each $v \in V$, the path from v to r via parent field is a shortest path.

Proof.

Since $r.level = 0$, the path from v to r has $v.level$ edges.

Due to the previous theorem, no edge can reduce *level* more than one, all paths to r from v cannot be shorter than $v.level$. □

Topic 16.3

Finding connected components

Detect a component

Algorithm 16.4: BFSConnected(Graph $G = (V, E)$, Vertex r, int id)

```
Queue Q;
$visited := \{r\}$;
$Q.enqueue(r)$;
$r.component := id$;
while not $Q.empty()$ do
    $v := Q.dequeue()$;
    for $w \in G.adjacent(v)$ do
        if $w \notin visited$ then
            $visited := visited \cup \{w\}$;
            $Q.enqueue(w)$;
            $w.component := id$
```

Find all connected components

Algorithm 16.5: CC(Graph $G = (V, E)$)

```
1 for $v \in V$ do
2     $v.component := 0$
3 $componentId := 1$;
4 while $r \in V$ such that $r.component == 0$ do
5     BFSConnected($G, r, componentId$);
6     $componentId := componentId + 1$;
```

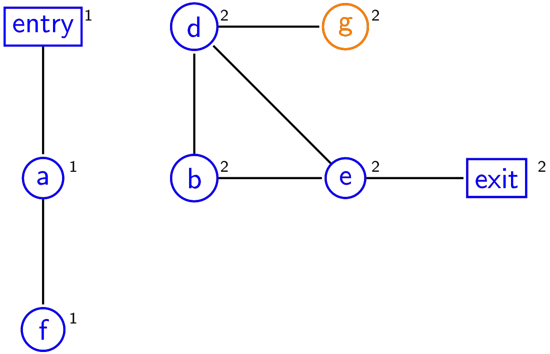
Exercise 16.2

a. *What is the cost of evaluating the condition at line 4?*

b. *What is the running time of the above procedure?*

Commentary: We should not evaluate the condition at line 4 from start. We should try to reuse the previous runs of the condition.

Example: find all connected components



Topic 16.4

Checking bipartite graph

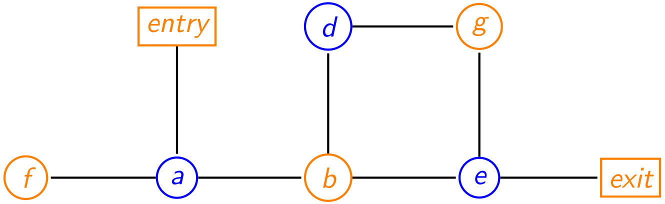
Bipartite graphs

Definition 16.2

A graph $G = (V, E)$ is bipartite if there are V_1 and V_2 such that $V = V_1 \uplus V_2$ and for all $e \in E$, $e \not\subseteq V_1$ and $e \not\subseteq V_2$.

Example 16.1

The following is a bipartite graph. Where $V_1 = \{entry, f, b, g, exit\}$ and $V_2 = \{a, d, e\}$.



Theorem 16.4

A bipartite graph does not contain cycles of odd length. Done in tutorial.

Checking Bipartite graph

Algorithm 16.6: IsBipartite(Graph $G = (V, E)$)

```
1 Assume(graph is connected);
2 Choose $r \in V$;
3 BFSSpanning($G, r$);
4 if for each $\{v, v'\} \in E$ $v.level \neq v'.level$ then
5     return True;
6 return False;
```

Exercise 16.3

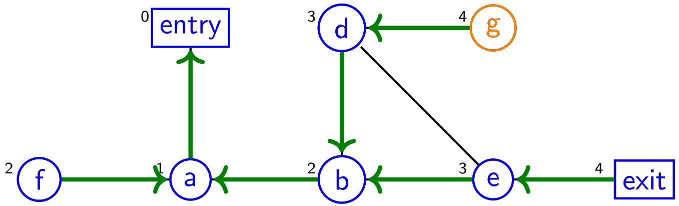
a. Modify the above algorithm to support not connected graph.

a. What is the cost of evaluating the condition at line 3?

b. What is the running time of the above procedure?

Example : IsBipartite

We ran BFS on the following graph.

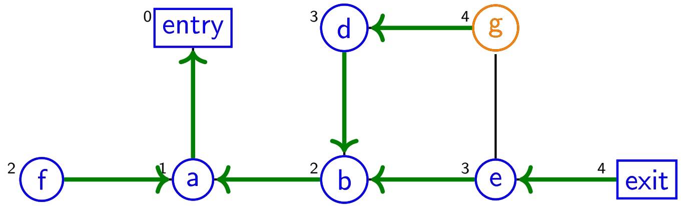


Since $d.level = b.level$, (d, e) edge causes the if condition to fail.

Therefore, the graph is not bipartite.

Example : IsBipartite another example

We ran BFS on the following graph.



BFS spanning tree edges will naturally satisfy the if condition.

Since $g.level \neq e.level$, the extra edge (g, e) also does not cause the if condition to fail.

Therefore, the graph is bipartite.

Correctness of IsBipartite

Theorem 16.5

If IsBipartite$(G = (V, E))$ *returns true, G is bipartite.*

Proof.

Let $V_1 = \{v | v.level\%2 = 0\}$ and $V_2 = \{v | v.level\%2 = 1\}$.

Due to the condition in IsBipartite, there are no edges connecting same level.

Due to theorem 16.2, we only have edges connecting neighbouring levels.

There are no edges that are inside V_1 or V_2. $\square$

Correctness of IsBipartite

Theorem 16.6

If IsBipartite$(G = (V, E))$ *returns false,* G *is not bipartite.*

Proof.

Let $\{v_1, v_2\} \in E$ such that that $v_1.level = v_2.level$.

Let v be the least common ancestor of v_1 and v_2 in the spanning tree induced by the run of BFS.

$v_1..v...v_2 v_1$ is an odd cycle.

Therefore, G is not bipartite. □

Topic 16.5

Diameter of a graph

Diameter of a graph G

Definition 16.3

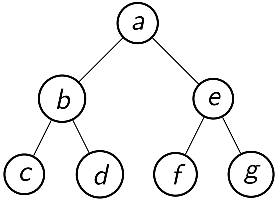
For a graph G, let the distance(v,v') be the length of a shortest path between v and v'.

Definition 16.4

For a graph $G = (V, R')$, $diameter(G) = max\{distance(v, v')|\{v, v'\} \in E\}$.

Diameter is only defined for a connected graph.

Example 16.2



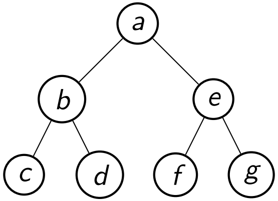
The diameter of the above graph is 4.

Can we use BFS for diameter?

Let us run $\text{BFSSpanning}(G = (V, E), r)$ for some $r \in V$.

Let *maxlevel* be the maximum level assigned to a node in the above graph.

Example 16.3



In the above graph, let $r = a$. The maxlevel is 2.

BFS diameter relation.

Theorem 16.7

Let maxlevel be the maximum level assigned to a node in $G = (V, E)$ *after running BFS from node* r. $maxlevel \leq diameter(G) \leq 2 * maxlevel$

Proof.

Since there are nodes *maxlevel* distance away from r, $maxlevel \leq diameter(G)$.

Let $v_1, v_2 \in V$. $distance(r, v_1) \leq maxlevel$ and $distance(r, v_2) \leq maxlevel$.

Therefore, $distance(r, v_1) + distance(r, v_2) \leq 2 * maxlevel$.

Therefore, $distance(v_1, v_2) \leq 2 * maxlevel$.

Therefore, $diameter(G) \leq 2 * maxlevel$.

□

All runs of BFS

Algorithm 16.7: DIAMETER(Graph $G = (V, E)$)

```
ASSUME(graph is connected);
$maxlevel := 0$;
for $r \in V$ do
    BFSSPANNING($G, r$);
    $maxlevel' :=$ maximum level assigned to a node;
    $maxlevel := max(maxlevel, maxlevel')$
return maxlevel;
```

Topic 16.6

Tutorial problems

Exercise: shortest path

Exercise 16.4

There are many variations of BFS to solve various needs. For example, suppose that every edge e=(u,v) also has a weight w(e) (say the width of the road from u to v). Assume that the set of values that w(e) can take is small. For a path p= (v1,v2,...,vk), let the weight w(p) be the minimum of the weights of the edges in the path. We would like to find a shortest path from a vertex s to all vertices v. If there are multiple such paths, we would like to find a path whose weight is maximum. Can we adapt BFS to detect this path?

Exercise: correctness of BFS

Exercise 16.5

Write an induction proof to show that if vertex r and v are connected in a graph G, then v will be visited in call $\textsc{BFSConnected}$*(Graph $G = (V, E)$, Vertex r, int id).*

Exercise: Graph with two kinds of edges

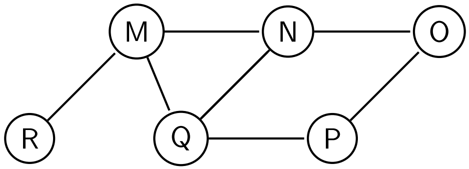
Exercise 16.6

Suppose that there is an undirected graph G(V,E) where the edges are colored either red or blue. Given two vertices u and v. It is desired to (i) find the shortest path irrespective of colour, (ii) find the shortest path, and of these paths, the one with the fewest red edges, (iii) a path with the fewest red edges. Draw an example where the above three paths are distinct. Clearly, to solve (i), BFS is the answer. How will you design algorithms for (ii) and (iii)?

Exercise: order of traversal (quiz 23)

Exercise 16.7

Is MNRQPO a possible BFS traversal for the following graph?



End of Lecture 16