Breadth First Search

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

This lecture will introduce **breadth first search** (BFS) for traversing a graph. We will assume directed graphs because the extension to undirected graphs is straightforward. To make our discussion concrete, we will consider a concrete problem: **single source shortest path** (SSSP) with unit weights, which can be elegantly solved by BFS.

Shortest Path

Let $G = (V, E)$ be a directed graph.

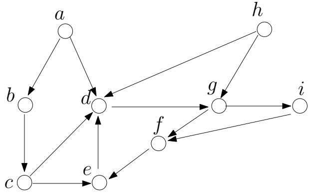
A **path** in G is a sequence of edges $(v_1, v_2), (v_2, v_3), ..., (v_\ell, v_{\ell+1})$, where $v_1, v_2, ..., v_{\ell+1}$ are distinct vertices, and ℓ is an integer at least 1. The value ℓ is called the **length** of the path. The path is said to be **from** v_1 **to** $v_{\ell+1}$.

- Sometimes, we will also denote the path as $v_1 \to v_2 \to ... \to v_{\ell+1}$.

Given two vertices $u, v \in V$, a **shortest path** from u to v is a path of the minimum length from u to v.

If there is no path from u to v, then v is **unreachable** from u.

Example



There are several paths from a to g:

- $a \to b \to c \to d \to g$ (length 4)
- $a \to b \to c \to e \to d \to g$ (length 5)
- $a \to d \to g$ (length 2)

The last one is a shortest path.

Note that h is unreachable from a.

Single Source Shortest Path (SSSP) with Unit Weights

Let $G = (V, E)$ be a directed graph, and s be a vertex in V. The goal of the **SSSP problem** is to find, for every other vertex $t \in V \setminus \{s\}$, a shortest path from s to t, unless t is unreachable from s.

Next, we will describe the BFS algorithm to solve the problem in $O(|V| + |E|)$ time.

At first glance, this may look surprising because the total length of all the shortest paths may reach $\Omega(|V|^2)$, even when $|E| = O(|V|)$ (can you give such an example?)! So shouldn't the algorithm need $\Omega(|V|^2)$ time just to output all the shortest paths in the worst case?

The answer, interestingly, is no. As will see, BFS encodes all the shortest paths in a BFS tree compactly, which uses only $O(|V|)$ space and can be output in $O(|V| + |E|)$ time.

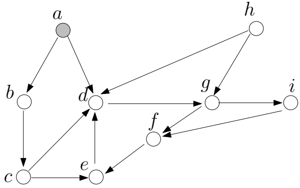
BFS

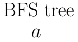
At the beginning, color all vertices in the graph **white** and create an empty BFS tree T.

Create a queue Q. Insert the source vertex s into Q and color it **gray** (which means "in the queue"). Make s the root of T.

Example

Suppose that the source vertex is a.





$Q = (a)$.

BFS

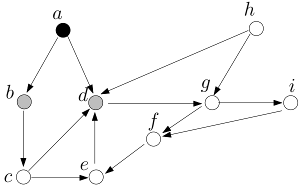
Repeat the following until Q is empty.

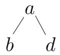
1. De-queue from Q the first vertex v.
2. For every out-neighbor u of v that is still white:
   2.1 En-queue u into Q, and color u **gray**.
   2.2 Make u a child of v in the BFS tree T.
3. Color v **black** (meaning that v is done).

> BFS behaves like "spreading a virus", as we will see from our running example.

Running Example

After de-queueing a:

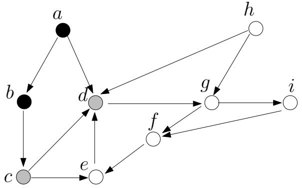


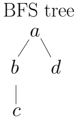


$Q = (b, d)$.

Running Example

After de-queueing b:

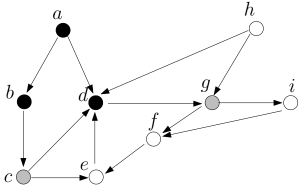


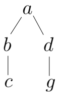


$Q = (d, c)$.

Running Example

After de-queueing d:

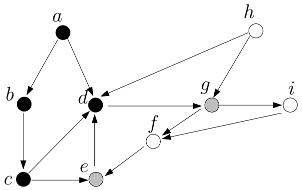


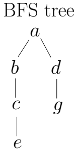


$Q = (c, g)$.

Running Example

After de-queueing c:



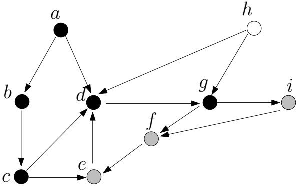


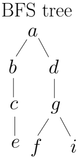
$Q = (g, e)$.

Note: d is not en-queued again because it is black.

Running Example

After de-queueing g:

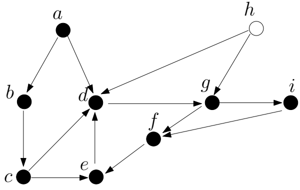


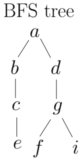


$Q = (e, f, i)$.

Running Example

After de-queueing e, f, i:



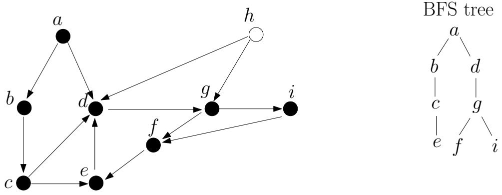


$Q = ()$.

This is the end of BFS. Note that h remains white — we can conclude that it is not reachable from a.

Running Example

Where are the shortest paths?



The shortest path from *a* to any vertex, say, *x* is simply the path from *a* to node *x* in the BFS tree!

- The proof will be left as an exercise.

Time Analysis

When a vertex v is de-queued, we spend $O(1 + d^+(v))$ time processing it, where $d^+(v)$ is the out-degree of v.

Clearly, every vertex enters the queue at most once.

The total running time of BFS is therefore

$$O\left(\sum_{v \in V} (1 + d^+(v))\right) \quad = \quad O(|V| + |E|).$$