Connected Components and Correctness of BFS in SSSP

Yufei Tao's Teaching Team

In the lecture, we have discussed the steps of BFS for solving a special version of the SSSP problem. However, we have not proved the algorithm's correctness yet. This will be done today.

Single Source Shortest Path (SSSP) with Unit Weights

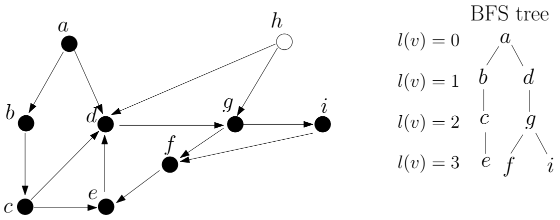
Let $G = (V, E)$ be a directed graph and s be a vertex in V. The goal of the **SSSP problem** is to find, for every other vertex $t \in V \setminus \{s\}$, a shortest path from s to t, unless t is unreachable from s.

Using BFS to Solve SSSP Problem

Run BFS algorithm starting from s on G, which returns a **BFS-tree** T.

For any $v \in V \setminus \{s\}$, the path from s to v in T as the shortest path from s to v in G. If the path does not exist, it means that s cannot reach v.

Using BFS to Solve SSSP Problem



For each vertex $v \in V$, let $\ell(v)$ denote the **level** of v in T, namely, the length of the path from s to v in T.

Proof of Correctness

We now prove the correctness of BFS, starting with a useful lemma.

Lemma 1: For any two vertices $u, v \in V$ such that $u \neq v$, if $\ell(u) < \ell(v)$, then u must be enqueued before v during the BFS.

Proof: We will prove this by induction.

Base Case. $\ell(v) = 1$. Hence, $\ell(u) = 0$, meaning that u is the source s. As s is enqueued at the very beginning of BFS, the base case holds.

Inductive Case.

Inductive assumption: For any two vertices u, v with $\ell(u) < \ell(v) \leq L - 1$ where $L \geq 2$, it always holds that u is enqueued before v.

Consider any vertices u and v satisfying $\ell(u) < \ell(v) = L$. If u is the root of T, then $u = s$ and is obviously enqueued before v. Next, we consider that u is not the root.

Let p_u and p_v be their parents in the BFS-tree T, respectively. We have $\ell(p_u) = \ell(u) - 1$ and $\ell(p_v) = \ell(v) - 1$. It follows that $\ell(p_u) < \ell(p_v) \leq L - 1$.

By the inductive assumption, p_u is enqueued before p_v. From the FIFO property of queue, p_u is dequeued before p_v. As u (resp., v) is enqueued right after p_u (resp., p_v) is dequeued, u is enqueued before v. □

We now prove the correctness of BBS.

Theorem: For any vertex $v \in V$, the path from s to v in T is a shortest path from s to v in G.

We will prove a stronger claim by induction:

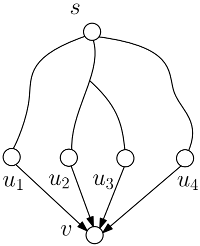
Claim: If a vertex $v \in V$ has shortest path distance L from s, then $\ell(v) = L$.

Base Case. $L = 0$ or 1.

- s is the only vertex with shortest path distance 0 from s. It is obvious that $\ell(s) = 0$.
- Every vertex v with shortest path distance 1 from s is an out-neighbor of s. Thus, v is enqueued when s is dequeued and must have $\ell(v) = 1$.

Inductive Case.

Inductive assumption: If a vertex v has shortest path distance $L \le k - 1$ from s where $k \ge 2$, then $\ell(v) = L$.

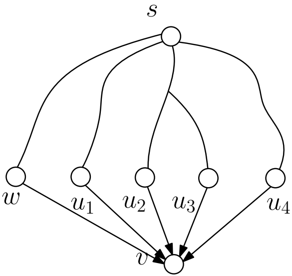


Let v be a vertex with shortest path distance k from s. Consider all the shortest paths from s to v and let U denote the set of predecessors of v on those paths. Furthermore, let u_1 denote the vertex in U that was enqueued the earliest during BFS. The shortest path distance from s to u_1 is $k - 1$.

By the inductive assumption, $\ell(u_1) = k - 1$. To prove $\ell(v) = k$, it suffices to prove that v is enqueued at the moment u_1 is dequeued, or equivalently:

> **Claim:** v is white when u_1 is dequeued.

We will prove this by contradiction.



Suppose that when u_1 is dequeued, v is not white. This means that v has already been added to the BFS-tree T when u_1 is dequeued. Define w as the parent of v in T (i.e., v is enqueued after w is dequeued).

By Lemma 1, We have $\ell(w) \leq \ell(u_1)$ as w is dequeued before u_1. We further have $\ell(w) \neq \ell(u_1)$; otherwise, w must belong to U, which contradicts the definition of u_1.

It follows that $\ell(w) < \ell(u_1)$. However, this means that the shortest path distance from s to w is less than $k - 1$. Thus, the shortest path distance from s to v is less than k, giving a contradiction.

□

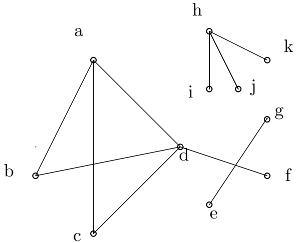
We have proved the correctness of BFS in solving the SSSP problem with unit weights on directed graphs. The algorithm is also correct when it runs on **undirected** graphs. The proof is similar and omitted.

Next, we will discuss **connected components**, an important concept in graph theory.

Let $G = (V, E)$ be an undirected graph.

A **connected component** of G is a set $S \subseteq V$ of vertices s.t.

- (connectivity) any two vertices in S are reachable from each other;
- (maximality) it is not possible to add another vertex to S while still satisfying the above requirement.



G

There are 3 CCs:
$\{a, b, c, d, f\}, \{g, e\}, \{h, i, j, k\}$

Lemma 2: Take an arbitrary vertex s. The CC covering s is the set R of vertices in G reachable from s.

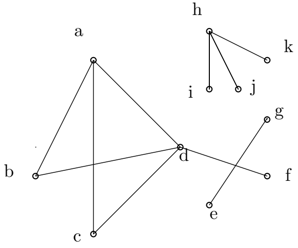
Proof: Let C be the CC covering s. By the connectivity property, we know that every vertex in CC is reachable from s. Hence, $C \subseteq R$.

If $C \subset R$, then R has at least one vertex u that does not appear in C. However, the existence of u violates the maximality property of C. $\square$

Next, we discuss how to find all the CCs of the input (undirected) graph $G = (V, E)$. As shown next, both BFS and DFS can be deployed for the purpose.

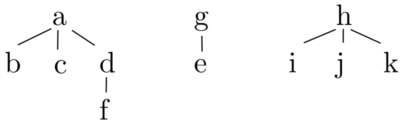
A BFS Solution

1. Run BFS on G starting from a white source vertex
2. Output the vertex set of the BFS-tree
3. If there is still a white vertex in G, repeat from 1



G

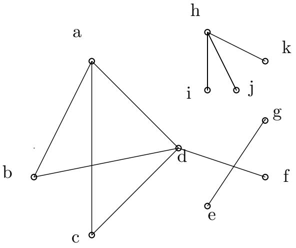
BFS-forest



Discuss: Why is the algorithm correct?

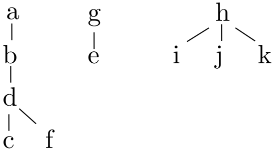
A DFS Solution

1. Run DFS on G starting from a white source vertex
2. Output the vertex set of the DFS-tree
3. If there is still a white vertex in G, repeat from 1



G

DFS-forest



Proof of correctness

Claim: The vertex set S of each DFS-tree is a CC of G.

Proof: We will prove the claim for the first DFS-tree produced. You can then think about how to prove the claim for the other DFS-trees.

Let s be the source vertex of DFS. We will show that the DFS-tree contains **all and only** the vertices reachable from s.

"All": Let v be a vertex reachable from s. At the beginning of DFS, there is a white path from s to v. By the white path theorem, v must be in the subtree of s, namely, in the DFS-tree.

"Only": Every vertex in the DFS-tree is clearly reachable from s (the tree itself gives a path). □