Lecture 13: Breadth-first Search
CSC2100 Data Structure

Yufei Tao

CSE department, CUHK

April 3, 2011
In this lecture, we will discuss *breadth-first search*. This is a very fundamental graph algorithm, which underlies the solutions to many graph problems.
1 Problem

2 Breadth-first search
   - Rationale
   - Pseudocode

3 Analysis
   - Running time
   - Access order
Problem (Reachability)

Given an undirected graph $G$ and a vertex $s$ in $G$, output all the vertices in $G$ that can be reached from $s$.

Example

Answer: $\{s, a, b, c, d, e, f, g\}$. 
BFS overview

- The *breadth-first search* (BFS) algorithm traverses all the vertices reachable from *s*.
- It outputs those vertices in *ascending* order of their distances to *s*.
  - Namely, first vertices that are one-hop away from *s*, then vertices that are 2-hops away, etc.
- Every vertex (reachable from *s*) will be output exactly once.
Let us get an idea of the algorithm from an example.

- At the beginning, color all the vertices white (which means “not touched yet”).
- Initiate an empty queue $Q$ (a linked list with the first-in-first-out property).
Output $s$, and color it **black** (which means “done”).

- Insert all the neighbors of $s$ into $Q$, and color them **grey** (which means “in the queue”). Now $Q = \{a, b, c\}$.
Breadth-first search

Analysis Summary

Rationale

**BFS example (cont.)**

- Pop out the first vertex $a$ of $Q$.
- Output $a$ and color it black.
- Insert all the **white neighbors** of $a$ into $Q$, and color them **grey**. Only $d$ is en-queued; and $Q = \{b, c, d\}$. 

---

Example

- Diagram showing the BFS with vertices $s, a, b, c, d, e, f, g, h, i$ and edges connecting them. The vertices are color-coded to illustrate the black, grey, and white states as per the BFS algorithm.
The rest of the algorithm simply repeats the above until \( Q \) is empty. Let us see one more step:

- Pop out the first vertex \( b \) of \( Q \).
- Output \( b \) and color it black.
- Insert all the white neighbors of \( b \) into \( Q \), and color them grey. \( e \) and \( f \) are en-queued; and \( Q = \{c, d, e, f\} \).
Algorithm $BFS(s)$
1. color all the vertices white
2. initialize an empty queue $Q$
3. for each neighbor $v$ of $s$
4. insert $v$ in $Q$; $color[v] = \text{grey}$
5. output $s$; $color[s] = \text{black}$
6. while $Q$ is not empty
7. $u = \text{top of } Q$; remove $u$ from $Q$
8. for each neighbor $v$ of $u$
9. if $color[v] = \text{white}$
10. insert $v$ in $Q$; $color[v] = \text{grey}$
11. output $u$; $color[u] = \text{black}$
Let us assume that the input graph $G$ is stored with an adjacency list.

- Coloring all vertices white (at the beginning of BFS) takes $O(|V|)$ time, where $V$ is the set of vertices in $G$.
- Then, every edge in $E$ (the set of edges in $G$) is processed at most twice.

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

Proof of the access order

We will prove that BFS outputs the vertices in $G$ (reachable from $s$) in ascending order of their distances from $s$.

- Let $V_i$ ($i \geq 0$) be the set of vertices that are $i$-hops away from $s$.
- The next lemma essentially shows that BFS outputs all the vertices of $V_i$ before outputting any vertex in $V_{i+1}$, for any possible $i$. 
Lemma

For any $i$, when BFS finishes outputting all the vertices of $V_i$, $Q$ contains all and only the vertices of $V_{i+1}$.

Proof

We prove the lemma by induction. The basic step with $i = 0$ is trivial, noticing that $V_0 = \{s\}$. Next, assuming that the lemma is correct up to $i \leq k$, we show its correctness for $i = k + 1$.

At the moment when all the vertices of $V_k$ have been output, the inductive assumption implies:

- every vertex of $V_{k+1}$ is in $Q$;
- $Q$ does not have any other vertex;
- $V_0, ..., V_{k-1}$ have been output.
Proof (cont.)

It suffices to prove that when $V_{k+1}$ has been output completely, the entire $V_{k+2}$ is in $Q$, and $Q$ does not have any other vertex. This is true from the following:

- Consider any $u \in V_{k+1}$. BFS en-queues only the white neighbors of $u$. All these neighbors must be in $V_{k+2}$.
- Any vertex in $V_{k+2}$ must have at least a neighbor in $V_{k+1}$. 

□
Problem
Breadth-first search

Analysis

Summary

Playback of this lecture:

- Breadth-first search.
- Running time $O(|V| + |E|)$.

Remark: BFS can be extended to work on directed graphs in a straightforward manner.