Algorithms: Applications of BFS

Suppose we have a graph $G = (V, E)$. A given graph could have few edges, or lots of edges, or anything in between. Let's think about the range of possible relationships between V and E.

1 The smallest possible value of $|E|$ is __________________________.

2 $|E|$ is $O\Big(\qquad \Big)$ because _________________________________.

3 When G is a tree, $|E|$ is $\Theta\Big(\qquad \Big)$ because _________________.

Now, recall from last class that we showed breadth-first search (BFS) can be implemented to run in $\Theta(|V| + |E|)$ time.

4 In terms of Θ, how fast does BFS run, as a function of $|V|$, when G is a tree?

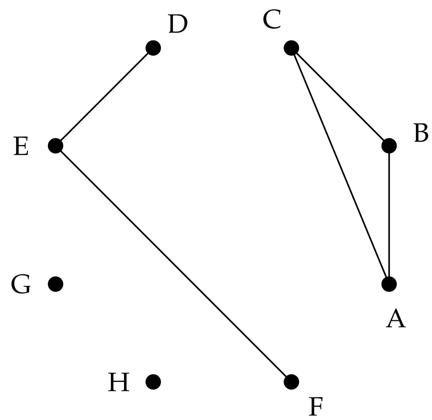
5 How fast does BFS run, as a function of $|V|$, when G is very dense, *i.e.* it contains some constant fraction (say, half) of all possible edges?

A first application of BFS

6 Describe an algorithm to find the connected components of a graph G.

Input: a graph $G = (V, E)$
Output: a set of sets of vertices, `Set<Set<Vertex>>`, where each set contains the vertices in some (maximal) connected component. That is, all the vertices within each set should be connected; no vertex should be connected to vertices in any other set; and every vertex in V should be contained in exactly one of the sets.

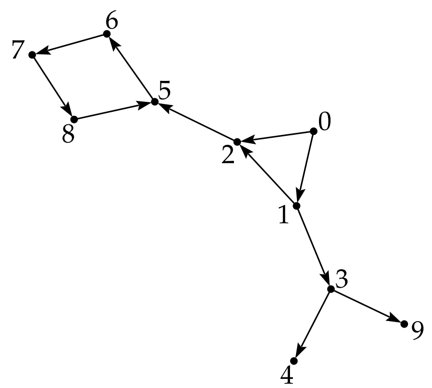
For example, given the graph below, the algorithm should return $\{\{D, E, F\}, \{C, B, A\}, \{G\}, \{H\}\}$.

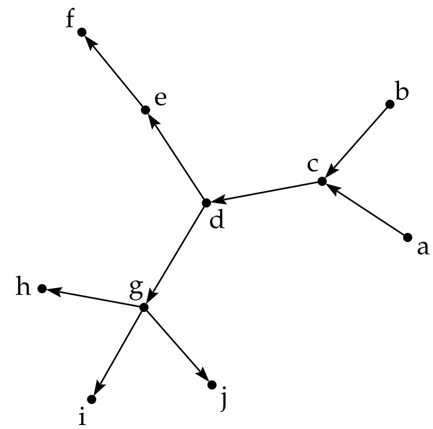


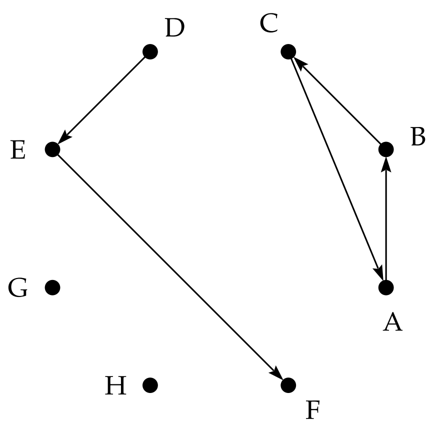
Describe your algorithm (using informal prose or pseudocode) and analyze its asymptotic running time.

A second application of BFS

Model 1: Directed graphs







$$G = (V, E)$$
$$V = \{\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \eta, \theta\}$$
$$E = \{(\alpha, \delta), (\theta, \eta), (\beta, \alpha), (\zeta, \delta), (\epsilon, \eta), (\gamma, \alpha)\}$$

Key: α = alpha, β = beta, γ = gamma, δ = delta, ϵ = epsilon, ζ = zeta, η = eta, θ = theta

- The *indegree* of vertex C is 1. The *outdegree* of vertex C is also 1. The *indegree* of vertex 5 is 2. The *outdegree* of vertex g is 3.
- $\{C, B, A\}$ is a *strongly connected component*. So is $\{5, 6, 7, 8\}$. $\{D, E, F\}$ is a *weakly connected component* but not a strongly connected one.
- b, c, d, e, f is a path. $0, 1, 2, 5, 6$ is a path. So is D, E, F. $0, 1, 2, 5, 8$ is not a path. Neither is F, E, D.

7 What is the difference between directed graphs and the (undirected) graphs we saw on a previous activity?

8 The previous activity defined graphs as consisting of a set V of vertices and a set E of edges, where each edge is a set of two vertices. How would you modify this definition to allow for directed graphs?

9 For each of the following graph terms/concepts, say whether you think its definition needs to be modified for directed graphs; if so, say what the new definition should be.

1 *vertex*

2 *degree*

3 *path*

4 *cycle*

10 What (if anything) about our implementation of BFS needs to be modified for BFS to work sensibly on directed graphs?

Definition 1. A directed graph $G = (V, E)$ is *strongly connected* if for any two vertices $u, v \in V$ there is a (directed) path from u to v, *and also* from v to u.

11 Describe a brute force algorithm for determining whether a given directed graph G is strongly connected.

12 Analyze the running time of your algorithm. Express your answer using Θ.

Model 2: Reverse graphs and strong connectivity

Definition 2. Given a directed graph G, its *reverse graph* G^{rev} is the graph with the same vertices and edges, except with all the edges reversed.

Theorem 3. *A directed graph $G = (V, E)$ is strongly connected if and only if given any $s \in V$,*

- *all vertices are reachable from s in G, and*
- *all vertices are reachable from s in G^{rev}.*

13 Based on the above theorem, describe an algorithm to determine whether a given directed graph $G = (V, E)$ is strongly connected, and analyze its running time.

14 Can you give an informal, intuitive explanation why the theorem is true? (*Hint*: if all vertices are reachable from s in G^{rev}, what does it tell us about G?)