CPSC 365 / ECON 365: Algorithms Yale University

Discussion 2

Out: February 10, 2022 Discussed: February 11, 2022

1 Testing Bipartiteness

Design an algorithm to test if a graph is bipartite.

- If the graph is bipartite, the algorithm should output YES.
- If the graph is *not* bipartite (so it has an odd cycle), the algorithm should output NO along with an odd cycle in G.

2 Finding Shortest Cycle

Here's a proposal for how to find the length of the shortest cycle in an undirected unweighted graph: When a back edge, say (v, w), is encountered during a depth-first search, it forms a cycle with the DFS tree edges from w to v. The length of the cycle is $\texttt{level}[v] - \texttt{level}[w] + 1$, where the `level` of a vertex is its distance in the DFS tree from the root vertex. This suggests the following algorithm:

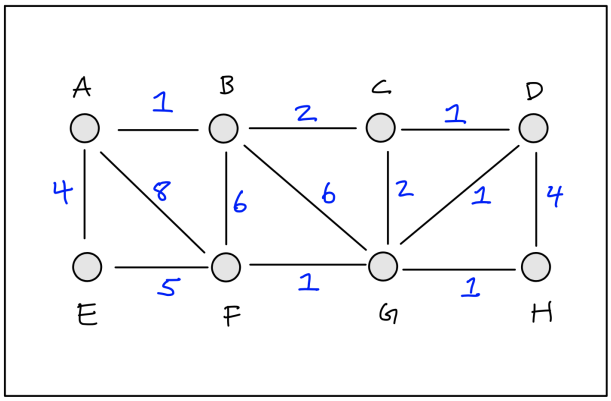
1. Do a depth-first search, keeping track of the level of each vertex.
2. Each time a back edge is encountered, compute the cycle length and save it if it is smaller than the shortest one previously seen.

Is the algorithm above correct? Either prove the algorithm above correctly computes the length of the shortest cycle; or show that the algorithm above does not always work by providing a counterexample as well as a brief explanation.

Optional: Suppose we use BFS rather than DFS. Does the algorithm work?

3 Dijkstra

Consider the following weighted, undirected graph:



1. Run Dijkstra's algorithm on the following graph starting from vertex A. Draw a table showing the intermediate distance values of all nodes at each iteration of the algorithm.
2. Draw the final shortest-path tree.