Proof of Correctness for Bipartite Test Algorithm

Christopher Wood

12/21/11

1 Bipartite Test (BT) Algorithm

Input: The adjacency matrix for a graph, G.

Output: Boolean value indicating whether or not G is bipartite.

Description: The idea of the algorithm is to perform a depth-first-search (DFS) on the input graph G in order to visit every vertex in search of an odd cycle. The DFS can be applied recursively on the tree by selecting an initial start vertex, v_0, which is set to the current vertex, and then recursively visiting each of the neighbors of the current vertex until all vertices have been visited. At each vertex, the depth of the DFS traversal is assigned to the vertex in such a way so that v_0 would receive a label of 0. Using these visited labels, if at any point during the DFS a vertex is visited more than once, the even/odd numerical difference of the current DFS depth and the already-visited vertex label is compared. If this difference is even then the cycle must be even as well, and so the current DFS recursive path is stopped and the traversal continues as normal. If this difference is odd then the cycle is odd, at which point the algorithm terminates and returns false. Once all vertices in $V(G)$ have been visited and no odd cycle has been detected, the algorithm terminates and returns true. If there are no further traversal paths and yet all vertices in $V(G)$ have not been visited (indication of at least one isolated vertex), then the algorithm terminates and returns false.

2 Theorem

Applying the BT algorithm on a given graph G will indicate whether or not G is bipartite.

2.1 Proof

Let G be a graph on n vertices. If G is disconnected then performing a DFS on G will not cover all of the vertices $V(G)$, which means that there exists at least one isolated vertex and therefore the graph cannot be bipartite.

If G is connected, then performing a DFS on G will cover all vertices in $V(G)$. Furthermore, if performing a DFS on G does not yield any vertices that are visited more than once then G must be a tree, and therefore is also bipartite.

If G is connected and performing a DFS on G yields a vertex that has been visited once already in the DFS traversal then a cycle has been detected. This is because the DFS traversal recursively builds a single path, and if a vertex is repeated in a path with $n > 2$ vertices then that path must be a cycle. The length of this cycle can be determined by examining the even/odd numerical difference between the visited vertex label and the current DFS traversal depth. For the case of this discussion, let v_i be the twice-visited vertex. If this difference is even, then the number of vertices along the path between the first visit of v_i and the second visit of v_i in the DFS is odd, and taking v_i into account we are left with an even number of vertices along this path, which makes the cycle even. If this difference is odd, then the number of vertices along the path between the first visit of v_i and the second visit of v_i is even, and by taking v_i into account we are left with a path of odd order, which is an indication of an odd cycle.