Data Structures and Algorithms

Practice-sheet

DFS Traversal

Note: Each graph is represented as adjacency list unless mentioned otherwise. Furthermore, n denotes the number of vertices and m denotes the number of edges.

1. Let $G = (V, E)$ be an undirected connected graph on n vertices and m edges given in the adjacency lists representation. A DFS traversal has been performed on the graph. Observe that there will be n distinct DFS calls during this traversal. Furthermore, there will be distinct times at which each of these DFS calls start and finish for any vertex. So each vertex $v \in V$ can be assigned two unique numbers $s(v)$ and $f(v)$ from the range $[0, 2n-1]$. The number $s(v)$ is the time at which DFS call for v was invoked and the number $f(v)$ is the time at which DFS call for v finishes. You are given two arrays S and F of size n storing $s(v)$ and $f(v)$ for each vertex v. For example, if r is the vertex from where we start the DFS traversal, then $S[r] = 0$ and $F[r] = 2n - 1$.

   - Find out the relationship, if any, among $\{S[u], S[v], F[u], F[v]\}$ for any two vertices u and v.
   - Given the graph G, and arrays S and F, design an $O(m + n)$ time algorithm to compute the DFS tree associated with S and F.
   - Suppose you are given only the arrays S and F; the graph G is not given at all. How can you construct the DFS tree in $O(n \log n)$ time ?

2. Suppose there is a connected graph $G = (V, E)$. You execute BFS traversal from a vertex v and get a rooted tree T. You execute a DFS tree and get the same rooted tree. What inference can you draw about the graph ?

3. Modify the DFS algorithm such that it detects and produces any cycle (if exists) in a given undirected graph. How efficient can you make the algorithm ?

4. We know that there may be many possible DFS trees for a graph depending upon the start vertex and the order in which we explore the neighbors of each vertex.

   Given a connected graph $G = (V, E)$, you are given a rooted tree T such that each edge of this tree is present in E. Design an efficient algorithm to determine if T is a DFS tree of G.

   Hint: make use of the solutions of some problems mentioned above.

5. You are given an undirected graph $G = (V, E)$ which is connected. G may have many cycles in it. You want to assign a direction to each edge in the graph so that the resulting directed graph does not have a cycle. Design an efficient algorithm for this task.

6. Let $\delta(u, v)$ denote the distance between u and v in an undirected graph $G = (V, E)$, where length of each edge is 1. The diameter of a connected graph is defined as maximum distance in the graph. In precise words, diameter is $\max_{u,v \in V} \delta(u, v)$. It

takes $O(mn)$ time to compute diameter of an undirected graph. Interestingly, for some special graphs, diameter can be computed much faster. One such family of graphs is the family of tree graphs.

You are given a rooted tree T on n vertices. Let G_T be the undirected graph obtained by removing directions from the edges of T. G_T is a tree graph. Use the ideas from BFS/DFS to design an $O(n)$ time algorithm for computing diameter of the graph G_T. (this problem was discussed in the tutorial).

1 Optional (and an exploratory) problem

This problem is meant for those students whose expectation from the course is more than just a good grade. You may pursue it after the course if you have more free time then. A connected graph G is said to be a Eulerian graph if the following property holds. Starting from any vertex v, it is possible to make a tour on this graph which terminates at v such that each edge is visited **exactly once** (though a vertex may be visited multiple times). The corresponding tour is called an Euler tour of G. See Figure **??** for a better understanding. You have to design an $O(m+n)$ time algorithm to output an Euler tour of a graph G, if it exists.

You might like to study Eulerian graph on Wikipedia or other sources on the web before solving this problem. You may use the following characterization of an Eulerian graph. A connected graph is Eulerian if and only if the number of edges incident on each vertex is even. Make use of this characterization of an Eulerian graph and use DFS traversal. **A sketch of the algorithm:** find a cycle C in the graph, remove all the edges of cycle from the graph and recursively find Euler tour in various components formed after removal of all edges of C, then stitch these tours using cycle C suitably to get an Euler tour of the original graph. If you want to test your coding skills, you are encouraged to code the algorithm.

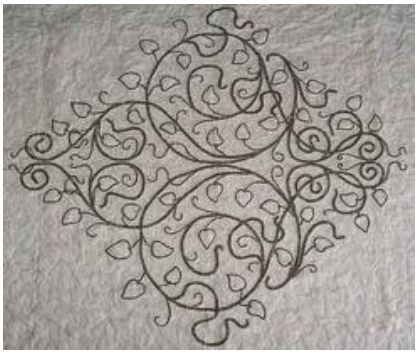

Figure 1: Design of an algorithm sometimes involves fine intricacies which expose a beautiful structure underlying the problem.