Applications of graph traversal

(CLRS 22.2, 22.3)

Undirected graphs:

Concepts:

- paths, cycles
- connectivity
- shortest paths
- trees
- spanning trees/ spanning forest

Basic problems on undirected graphs:

Briefly describe algorithms to answer the following questions, and analyze the complexity of your algorithm. Assume the graph is given as an adjacency list.

1. Is G connected?

2. How many CCs are in G?

3. Compute the connected components of G, label each vertex with the id of its CC

4. Given two vertices, are they in the same CC?

5. Given two vertice u, v, find a path between u and v

6. Does G contain a cycle?

7. Compute a spanning tree (forest) for G.

8. Is G a tree?

9. Assume G is a connected undirected graph; given any two vertices u, v, find the shortest path between them.

10. Assume G is a connected undirected graph with vertices $v_1, v_2, ..., v_n$; describe how to compute a 2D-array $d[1..n][1..n]$ such that $d[i][j]$ represents the length (number of edges) of the shortest path from v_i to v_j.

11. All-pair connectivity: Given a graph, support queries of the form: are u, v connected? (a) with no-preprocessing, how fast can you answer a query?; (b) Pre-process the graph into an approptiate data structure in order to answer connectivity queries in $O(1)$ time.

12. All-pair shortest paths: Given a graph, support queries of the form: find the shortest path from u to v. (a) no -preprecessing; (b) Describe how a graph can be pre-processed in order to answer shortest path queries in $O(1)$ time.

13. Two-colorability: Is it possible that the vertices of a given graph be assigned one of two colors, such that no edge connects vertices of the same color? (Note: this is equivalent to the question: is G bipartite?))

Directed graphs (digraphs)

Concepts:

- Reachability
- Directed paths and directed cycles
- Strongly connected components (SCC)
- Directly acyclic graphs (DAGs) and topological ordering
- Transitive closure (TC)

Basic problems on directed graphs:

Briefly describe algorithms to answer the following questions, and analyze the complexity of your algorithm. Assume the graph is given as an adjacency list.

1. Find all verties reachable from a given vertex u.

2. Given a vertex u, compute all vertices v such that u is reachable from v.

3. Given two vertices u, v, is there a (directed) path from u to v? If so, find such a path.

4. Given two vertices u, v, is there a (directed) path from u to v? If so, find such a *shortest* such path.

5. Does G have a directed cycle?

6. Is G a DAG? (ie is G acyclic?)

7. All-pair reachability: Given a graph, support queries of the form: given u, v, is v reachable from u? (a) no pre-precessing; (b) with pre-precessing, in $O(1)$ time per query;

8. Are two vertices u, v in the same SCC?

9. Compute the SCCs of G (label each vertex with the id of its SCC).