Data Structures and Algorithms : CS210/CS210A

Practice-sheet 5

Date: *1 October, 2014.*

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

1 Breadth First Search (BFS) Traversal

1. What is the running time of the BFS traversal algorithm if the underlying graph is represented by an adjacency matrix ?
2. We discussed BFS traversal from a vertex in an (undirected) graph. The same algorithm works for directed graphs as well. How will BFS traversal from a vertex in directed graph look like ? What relationship can you establish on the levels of the endpoints of an edge ?
3. Design an algorithm to count the number of all possible shortest paths tree rooted at a given vertex r in a graph $G = (V, E)$.

   Note: Computing all-possible BFS trees is more difficult and can be skipped for this course. Thanks to Abhishek Kumar Singh for making me realize it.
4. You are given an undirected connected graph $G = (V, E)$ and vertex $s \in V$. A single edge deletion may lead to increase in the distance from s to one or more vertices in the graph. Our aim is to find out all such edges in the graph G. Design an $O(m + n)$ time algorithm to compute all such edges.
5. Design an $O(n)$ time algorithm to determine if a given undirected graph has a cycle.
6. A graph is said to be connected if there is a single connected component in the graph. In the class, we designed an $O(m+n)$ time algorithm to determine if a given connected graph is bipartite. How can you extend the algorithm to handle a graph which is not connected (there are two or more connected components in the graph) ?
7. Suppose there is a connected graph where each edge has length which is a positive integer less than c. So we can define length of a path as the sum of the length of all edges lying on the path. Design an $O(mc + n)$ time algorithm to compute distance (length of the shortest path) from a given fixed vertex x to all other vertices in the graph. Your algorithm may use $O(mc + n)$ extra space if you wish.
8. **(slightly difficult problem)**

   You are given a graph $G = (V, E)$ and two vertices u and v. Furthermore, you are told that u and v belong to different connected components. The size of a connected component is defined as the number of edges present in that component. You have to determine whether the component of u is smaller than the component of v. Design a simple algorithm for this problem whose running time is of the order of the size of the smaller component. The algorithm should be easily implementable in practice.

9. Design an $O(mn)$ time algorithm which computes an $O(n^2)$ space data structure for a given graph $G = (V, E)$ such that given any two vertices $u, v \in V$, the data structure can output the shortest path between u and v. The time taken by the data structure to output the path has to be of the order of the number of vertices on the path.

10. (*This problem will make you realize that many problems can be modeled as graph problem in order to get an efficient solution*)

    The ministers of the cabinet were quite upset by the message from the Chief of Security stating that they would all have to change the four-digit room numbers on their offices.

    - It is a matter of security to change such things every now and then, to keep the enemy in the dark.
    - But look, I have chosen my number 1033 for good reasons. I am the Prime minister, you know!
    - I know, so therefore your new number 8179 is also a prime. You will just have to paste four new digits over the four old ones on your office door.
    - No, it's not that simple. Suppose that I change the first digit to an 8, then the number will read 8033 which is not a prime!
    - I see, being the prime minister you cannot stand having a non-prime number on your door even for a few seconds.
    - Correct! So I must invent a scheme for going from 1033 to 8179 by a path of prime numbers where only one digit is changed from one prime to the next prime.

    Now, the minister of finance, who had been eavesdropping, intervened.

    - No unnecessary expenditure, please! I happen to know that the price of a digit is one pound.
    - Hmm, in that case I need a computer program to minimize the cost. You don't know some very cheap software gurus, do you?
    - In fact, I do. You see, there is this programming contest going on...

    Help the prime minister to find the cheapest prime path between any two given four-digit primes! The first digit must be nonzero, of course. Here is a solution in the case above.
    1033
    1733
    3733
    3739
    3779
    8779
    8179

    The cost of this solution is 6 pounds. Note that the digit 1 which got pasted over in step 2 can not be reused in the last step a new 1 must be purchased.

2 Depth First Search (DFS) Traversal

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.

   - (Thanks to Abhishek Kumar Singh for sugesting this problem)
     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. Given a rooted tree on n vertices where the vertices are numbered from 0 to $n - 1$. Let r be the index of the root node. Design an $O(n)$ size data structure using which it takes $O(1)$ time to answer the following query for any $0 \leq i, j < n$.

   Is_Ancestor(i, j): Is i an ancestor of j ?

3. 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 ?

4. 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 ?

5. 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.

6. 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.

3 Slightly difficult problems

1. You are given a connected graph $G = (V, E)$. An edge is said to be a bridge if its removal disconnects the graph. Design an $O(m + n)$ time algorithm to compute all bridges in G.

2. Let $\delta(u, v)$ denote the distance between u and v in a graph $G = (V, E)$. 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.

4 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 1 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.

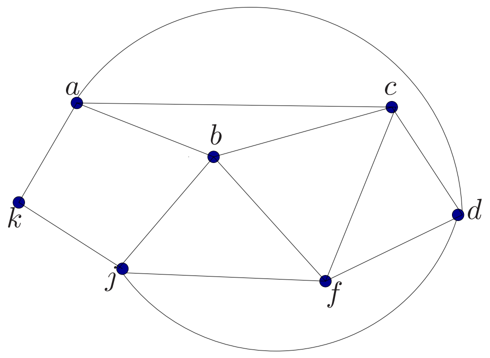

Figure 1: An Euler tour: $j - f - d - c - a - b - c - f - b - j - k - a - d - j$.

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