ESO207: Data Structures and Algorithms (Practice Problems – II)

Question 1: Cycle Detection in a graph

(a) Using BFS design an O(n) time algorithm to determine if a given undirected graph has a cycle.

(b) Using DFS design an O(m + n) time algorithm to determine if a directed graph has a cycle.

Question 2: On size of connected component

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.

Question 3: Topological ordering in DAG

Given a directed acyclic graph of n vertices and m edges, design an $O(n+m)$ time algorithm to compute its topological ordering (see textbook for the definition of topological sorting).

Question 4: Structure for answering ancestor descendent query

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 $0 \leq i, j < n$.

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

Question 5: Determine if a tree is DFS tree

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.

Question 6: Constructing a DAG

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.

Question 7: Euler tour of a graph

(Difficult) A connected graph G is said to be a Eulerian graph if starting from any vertex v, it is possible to make a walk on the 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. Design an $O(m + n)$ time algorithm to output an Euler tour of a graph G, if it exists.

(Hint: You may try using DFS along with 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.)

Question 8: Computing k-smallest elements

Given an array A storing n numbers. Design an $O(n + klogn)$ time algorithm to compute k-smallest elements from A. You may modify the array A if you wish.

Question 9: Comparison with k-th smallest element

You are given a binary heap H of size n, a number x, and a positive integer k. Design an $O(k)$ time algorithm to determine if x is smaller than k-th smallest element in H. You may use $O(k)$ extra space.

Question 10: Another scheduling problem

There are n jobs. There are n PCs and one supercomputer. Each job consists of two stages: first it needs to be preprocessed on the supercomputer, and then it needs to be finished on one of the PCs. Let us say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC. Since there are n PCs available, the finishing of the jobs can be performed in parallel – all jobs can be processed at the same time on their respective PCs. However, the supercomputer can only work on a single job at a time. So we need to find out the order in which to feed the jobs to the supercomputer. Our aim is to minimize the completion time of the schedule which is defined as the earliest time at which all jobs will have finished processing on the PCs.

Design a greedy strategy/algorithm that finds the order in which the jobs should be scheduled on the supercomputer so that completion time achieved is as small as possible.

Question 11: Subsequence detection

Sequence A is said to be subsequence of another sequence B if there is a way to delete zero or more elements from A so that the resulting sequence turns out to be B. For example $\langle a, b, a, a, c, b, d\rangle$ is a subsequence of $\langle c, a, a, b, a, b, a, b, c, a, c, b, b, d\rangle$. Let S and $S?$ be two sequences of characters. Let m and n be respectively the lengths of sequences S and $S?$. You may assume that S and $S?$ are given in the form of arrays. Getting inspiration from greedy approach, design an $O(m+n)$ time algorithm to detect whether S is a subsequence of $S?$. Prove correctness of the algorithm as well.

Question 12: Efficient construction of committee

The manager of a large student union on campus comes to you with the following problem. She is in charge of a group of n students, each of whom is scheduled to work one shift during the week. There are different jobs associated with these shifts (tending the main desk, helping with package delivery, rebooting cranky information kiosks, etc.), but we can view each shift as a single contiguous interval of time. There can be multiple shifts going on at once. She is trying to choose a subset of these n students to form a supervising committee that she can meet once a week. She considers such a committee to be complete if for every student not on the committee, that students shift overlaps (at least partially) the shift of some student who is on the committee. In this way, each students performance can be observed by at least one person who is serving on the committee.

Give an efficient algorithm that takes the schedule of n shifts and produces a complete supervising committee containing as few students as possible. You must also prove the correctness of the algorithm.

Question 13: MST with changing edge weights

There is a connected graph $G = (V, E)$, with edge costs that are all distinct. You are given a minimum spanning tree T for G. Let v be a leaf node in T. Weight of one of the edges incident on v in the graph has decreased. This might lead to update T. You have to determine if T has to be updated, and in case, the MST is updated, you have to compute the new minimum spanning tree in $O(n)$ time only. (Note that there could be $O(n)$ edges incident on v in the original graph.)

Question 14: Exploring a simple hash function

Consider the hash function $h(i) = i \mod n$. Assume the universe size, $m > n^2$.

(a) Describe a set $S \subset U$ of size n for which the function h is too bad: all elements of S are hashed into the same location in the hash table.

(b) Describe a set $S \subset U$ of size n for which the function h is perfect: all elements of S are hashed into the different locations in the hash table.

Credits: Thanks to Prof. Surender Baswana for sharing many of the above questions from earlier offerings of this course.