CS 403 Algorithm Design & Analysis
Lab Assignment 2

- Submit a report (with full explanation of your algorithm's running time and complexity) along with the codes and readme file in a zipped folder. The report should be in PDF format as a single document. If you want to assume something during coding, then mention in your report.
- The last date of submission is 6th March, 2017. You are advised to understand and read assignment questions and if you have any query then post it on moodle forum so that we all can have a good discussion.
- Late submissions will have penalty of 15% per day (that is 15% per day will be reduced on the score you achieve as the late submission penalty).
- You have to do code for all questions and give a good explanation in your report. Your reports would be evaluated thoroughly. Please provide pseudo-codes in report.
- We will provide test datasets at the time of evaluation. In that case, you code should be well generalized. Analyze your codes with different test sets during implementations of algorithms.
- Submit your assignment to:

qishtiyaq2402@gmail.com / tapesh_joham@students.iitmandi.ac.in

Q-1 : Implement an algorithm for Huffman codes using priority queues and without using priority queues and give comparison of running time with some fixed input dataset.

Q-2 : The incidence matrix for a directed graph $G = (V,E)$ is a set $|V| \times |E|$ matrix M such that $M_{ve} = -1$ if edge e leaves vertex v, $M_{ve} = 1$ if edge e enters vertex v, and $M_{ve} = 0$, otherwise. If a set of columns of M is linearly independent, then the corresponding set of edges does not contain a directed cycle. Verify the statement by implementation.

Q-3 : Suppose that a nonnegative weight w(e) is associated with each edge in an undirected graph $G = (V,E)$. Develop an efficient algorithm and implement it to find an acyclic subset of E of maximum total weight.

Q-4 : Suppose you are given a set $S = \{a_1,a_2,...,a_n\}$ of tasks, where task a_i requires p_i units of processing time to complete, once it has started. You have one computer on which to run these tasks, and the computer can run only one task at a time. Let c_i be the completion time of task a_i, that is, the time at which task a_i completes processing. Your goal is to minimize the average completion time, that is, to minimize $(1/n) \sum^{n}_{i=1} c_i$. For example, suppose there are two tasks, a_1 and a_2, with p1 = 3 and p2 = 5, and consider the schedule in which a_2 runs first, followed by a_1. Then $c_2 = 5$, $c_1 = 8$, and the average completion time is $(5+8)/2 = 6.5$.

Implement an algorithm that schedules the tasks so as to minimize the average completion time. Each task must run non-preemptively, that is, once task a_i is started, it must run continuously for p_i units of time. Prove that your algorithm minimizes the average completion time, and state the running time of your algorithm.

Q-5 : Suppose we have a set of activities to schedule among a large number of classrooms. We wish to schedule all the activities using as few classrooms as possible. Implement an efficient greedy algorithm to determine which activity should use which classroom. Also mention its computational complexity.

Hint : This is also known as the **interval-graph coloring problem**. We can create an interval graph whose vertices are the given activities and whose edges connect incompatible activities. The smallest number of colors required to color every vertex so that no two adjacent vertices are given the same color corresponds to finding the fewest classrooms needed to schedule all of the given activities.

Q-6 : Suppose we are given an instance of the Minimum Spanning Tree Problem on a graph G, with edge costs that are all positive and distinct. Let T be a minimum spanning tree for this instance. Now suppose we replace each edge cost c_e by its square, $(c_e)^2$, thereby creating a new instance of the problem with the same graph but different costs. Will T still be a minimum spanning tree for this new instance. Come up with an algorithm and implement it. Mention its computational complexity also.