Design and Analysis of Algorithms Practice
LAB 7
Date: 8^{th} October 2024

General instructions:

1. Students have to write the pseudo code first in their notebooks and implement it after that. Students can use either C / C++.

2. The point of contact (Member 1 as submitted in Gform) from the group has to submit all the programs. You may ask the TA, if you forgot the point of contact (Member 1).

3. Submit all the programs as a single Zip file in Google Class Room (GCR).

4. Pseudo code, Demonstration and Viva will be evaluated by the TA for 10 marks each and a total of 30. Pseudo code and Viva will be evaluated in the lab itself.

5. If the students wish to submit the programs later, then they can do it with in 2 days (i.e., if the lab is on Tuesday, then programs need to be submitted by Thursday 11:59 PM by point of contact (Member 1).). This evaluation will be considered for Demonstration 10 marks.

Greedy Paradigm

Q1) **Interval Scheduling Problem:** Given a set of n jobs. Job i starts at S_i and and finishes at F_i. Two jobs are called compatible if they don't overlap. The objective is find maximum subset of mutually compatible jobs. Means maximizing the number of jobs which are mutually compatible.

Generate the instances randomly. Design and implement **the greedy strategy (i.e., sort jobs by finish time)** and evaluate their associated time complexities in terms of Asymptotic Notations (Big O / Theta). Attached the Greedy_2.pdf for your reference.

Q2) **Scheduling to Minimizing Lateness Problem:** Please refer to the Greedy_2.pdf for understanding the problem.

Generate the instances randomly. Design and implement **the greedy strategy (i.e., sort jobs by deadline)** and evaluate their associated time complexities in terms of Asymptotic Notations (Big O / Theta). Attached the Greedy_2.pdf for your reference.

Q3) **Load Balancing Problem:** Suppose you need to complete n jobs, and the time it takes to complete job i is t_i. You are given m identical machines $M_1, M_2, ..., M_m$ to run the jobs on. Each machine can run only one job at a time, and each job must be completely run on a single machine. If you assign a set $J_j \subseteq \{1, 2, ..., n\}$ of jobs to machine M_j, then it will need $T_j = \sum_{i \in J_j} t_i$ time. Your goal is to partition the n jobs among the m machines to minimize $Max_i\ T_i$.

Generate the instances randomly. Design and implement **any three greedy strategies** and evaluate their associated time complexities in terms of Asymptotic Notations (Big O / Theta).