**Assignment 2**

**Job Partitioning on two machines using Greedy Algorithm**

**Team members: NAME - EMAIL - UID**

* Chris Davis Jaldi – jaldi.2@wright.edu – U01099335
* Aditya Mallakula – mallakula.2@wright.edu – U01093160
* Vanaja Uppala – uppala.19@wright.edu – U01080568

**Greedy Heuristic Used:**

This scheduling algorithm's greedy heuristic prioritizes the **earliest job finish time**:

* Jobs are scheduled by selecting the one that completes the quickest and can be placed on an available machine (prioritizing machine M1 over M2).
* By sorting the jobs by end time, the algorithm ensures that each job selected leaves the most amount of time for subsequent jobs to be scheduled.

**Greedy Algorithm Pseudocode:**

1. Initialization:
   * We start by creating two lists, *“M1\_jobs”* and *“M2\_jobs”*, which contain jobs assigned to machines M1 and M2, respectively.
   * To keep track of the end times of the last job scheduled on each machine, we define two variables, *“M1\_end”* and *“M2\_end”*, which are both initialized to zero.
   * We also define a variable called *“total\_jobs”* to keep track of how many jobs are scheduled.
2. Core loop:
   * The algorithm starts by sorting the list of jobs by end time in ascending order.
   * For each job (*job\_id, start\_time, end\_time*) on the sorted job list:
     + If the job's *“start\_time”* is greater than or equal to *“M1\_end”*, assign it to machine M1 and add the job ID to *“M1\_jobs”*.
     + Update *“M1\_end”* to *“end\_time”*.
     + Increase the *“total\_jobs”* count.
   * Otherwise, if the job's *“start\_time”* is greater than or equal to *“M2\_end”*, assign it to machine M2.
     + add the *“job\_id”* to *“M2\_jobs”*.
     + Set *“M2\_end”* to *“end\_time”*.
     + Increase the *“total\_jobs”* count.
3. Output:
   * The loop repeats until all jobs have been processed.
   * After all jobs are scheduled, the algorithm writes the following information into an output file.
     + Total number of jobs scheduled (total\_jobs).
     + The list of jobs assigned to M1 (M1\_jobs).
     + The list of jobs assigned to M2 (M2\_jobs).

**Complexity Analysis:**

* **Time Complexity:**
  + Job Loading Complexity: Reading n jobs from the input file takes O(n) time.
  + Sorting Jobs: Sorting the jobs by their end times takes O(nlogn).
  + Scheduling Jobs: Iterating through the sorted jobs and scheduling them takes O(n) time.
  + Overall Complexity: O(nlogn)
    - The dominant step is the sorting of the jobs, so the overall time complexity of the algorithm is O(nlogn).