Exercise 25:

**Case 1: Machines with Equal Capacities**

For machines with equal capacity and no prior work, the problem becomes the classic **"Load Balancing Problem"** or **"Multiprocessor Scheduling Problem"**. A common approach to solve this is by using the **"Greedy Algorithm"** known as **"Longest Processing Time First" (LPT)** or **"List Scheduling"**.

**Steps:**

1. **Sort jobs in descending order of execution time**: Sort the jobs based on the execution time T={t1,t2,...,tn}T = \{t\_1, t\_2, ..., t\_n\}T={t1​,t2​,...,tn​} such that t1≥t2≥...≥tnt\_1 \geq t\_2 \geq ... \geq t\_nt1​≥t2​≥...≥tn​.
2. **Assign jobs to the least loaded machine**: For each job in the sorted list, assign the job to the machine that currently has the least load (i.e., lowest cumulative job time so far).

**Greedy Algorithm (List Scheduling):**

1. Sort jobs JJJ in decreasing order of time ttt.
2. For each job JiJ\_iJi​, assign it to the machine with the least current total load.
3. Continue until all jobs are assigned.

This greedy approach is easy to implement and provides a solution close to the optimal one. It has a time complexity of O(nlog⁡n+nlog⁡m)O(n \log n + n \log m)O(nlogn+nlogm), where nnn is the number of jobs, and mmm is the number of machines.

**Case 2: Machines with Different Capacities**

When the machines have different capacities, this becomes a **"Heterogeneous Machine Scheduling"** problem. Each machine MjM\_jMj​ has a different capacity or speed sjs\_jsj​, which impacts how long it takes for each machine to complete a job.

Let sjs\_jsj​ represent the speed of machine MjM\_jMj​. If job JiJ\_iJi​ takes tit\_iti​ units of time to complete, the effective time to execute JiJ\_iJi​ on machine MjM\_jMj​ is ti/sjt\_i/s\_jti​/sj​.

**Steps:**

1. **Sort jobs in descending order** of execution time as in Case 1.
2. **Assign jobs based on effective load**: For each job JiJ\_iJi​, assign it to the machine where the ratio of job execution time to machine speed is lowest (i.e., current load of machine Mj+tisj\frac{\text{current load of machine } M\_j + t\_i}{s\_j}sj​current load of machine Mj​+ti​​ is minimized).

**Algorithm for Heterogeneous Machines:**

1. Sort the jobs in descending order of execution times t1,t2,...,tnt\_1, t\_2, ..., t\_nt1​,t2​,...,tn​.
2. For each job JiJ\_iJi​, calculate the estimated load for each machine MjM\_jMj​ (considering the speed of the machine) as: loadj=∑assigned job timessj\text{load}\_j = \frac{\sum \text{assigned job times}}{s\_j}loadj​=sj​∑assigned job times​
3. Assign job JiJ\_iJi​ to the machine that minimizes the load after assigning the job.
4. Continue until all jobs are assigned.

This approach, while more complex, aims to balance the load across heterogeneous machines, taking into account their varying speeds. The complexity is also O(nlog⁡n+nlog⁡m)O(n \log n + n \log m)O(nlogn+nlogm), but each load balancing step requires calculating effective load based on machine speeds.