Learning objective: find extreme instances showing the tightness of approximation ratios for classical scheduling strategies and their behaviors for the problem $P|C_{\text{max}}$.

**Exercise 1: $(2 - 1/m)$ ratio for LST**
Recall that LST has two properties:
1. task order is arbitrary;
2. each task is put on the first available processor.

To identify a worst-case instance, we need to define:
- the number of processors $m$;
- the number of tasks $n$, their costs and their order.

Recall the worst-case instance with $m = 2$. What is the worst-case instance with $m = 3$?

Hint: start by considering an arbitrary instance with a few tasks, draw the Gantt diagram obtained with LST and compare it with the optimal one.

**Exercise 2: $(4/3 - 1/3m)$ ratio for LPT**
Recall that LPT has two properties:
1. tasks are ordered by decreasing cost;
2. each task is put on the first available processor.

To identify a worst-case instance, we need to define:
- the number of processors $m$;
- the number of tasks $n$ and their costs.

What is the worst-case instance with $m = 2$?

Hint: the objective is to build an instance for which the optimal schedule is different than the one given by LPT. While the largest task is necessarily on any processor in both cases, the second largest task can be on the same processor in the optimal solution while it is necessarily on the other processor for LPT.

**Exercise 3: $13/11$ ratio for MULTIFIT**
MULTIFIT relies on a binary search to find the best possible objective value (makespan). For each considered makespan, it uses FFD to try producing a valid schedule. FFD considered tasks by decreasing cost and assign each of them to the first processor (starting with only one processor and adding a new one anytime a task does not fit). If it ends up with more than $m$ processors, the schedule is invalid.

To identify a worst-case instance, we need to define:
- the number of processors $m$;
- the number of tasks $n$ and their costs.
Find an instance for which MULTIFIT is not optimal?

Hint: with $m = 2$ processors, the worst-case ratio is $\frac{8}{7}$.

**Exercise 4: SLACK**

SLACK is an algorithm more recent (Della Croce et Scatamacchia, 2020) based on the following greedy strategy:

- Sort the tasks in decreasing order of their size;
- Cut the sorted set into sets of $m$ tasks;
- So, “slack” the difference between the size of the first job and the size of the last job of each set;
- Sort the sets of jobs in order of decreasing size of the “slack”.

SLACK has a complexity in time of $O(n \log(n) + n \log(m))$.

**Algorithm 1 : SLACK**

**Data:** instance of $P||C_{\text{max}}$, with $m$ machines, $n$ jobs and their execution times

1. Re-index the jobs, in order to obtain $p_1 \geq p_2 \geq \ldots \geq p_n$
2. Cut the obtained set into $\lceil \frac{n}{m} \rceil$ sets of $m$ jobs (add “dummy” jobs of size $n$ to each set, if $n$ is not a multiple of $m$)
3. Consider each set and compute the difference of the time between the first job and the last job.
4. Sort the sets in decreasing order of “slack” and form a new set // e.g: $\{ p_1 - p_m, p_m + 1 - p_{2m}, \ldots \}$
5. Apply the scheduling (allocation to the least loaded machine at this moment) to the resulting set.

Run SLACK on a simple example with $m = 3$ and $\{ p_i \} = \{ 11, 10, 7, 6, 5, 5, 5, 3, 2 \}$.

Find an instance with a different makespan with SLACK and LPT.

**Exercise 5: Proof of 2-approximation ratio for LST**

Provide the basic bounds to achieve the 2-approximation ratio for list heuristics. Which properties of those heuristics is noteworthy for this proof? Provide the reasoning of the rest of the proof.