The learning outcome of this practical session is to confirm skills related to streams by implementing common scheduling strategies for the problem $P||C_{\text{max}}$.

1 Scheduling Algorithms Implementation

Implement a function that uses a stream to generate an array of $n$ task execution times. The generation is assumed to be random uniformly between 0 and 1.

The maximum completion time of any schedule is bounded by 2 values: the sum of the execution times divided by the number of processors, and the maximum execution time of any task. Implement a function that computes these two lower bounds from the execution times and the number of processors (using a stream).

Implement LST using streams: one stream can be used to initialize an array of availability times, one for each processor, and another can be used to iterate over all tasks. The function takes as input execution times and a number of processors and returns the maximum completion time of the schedule.

Adapt the code for LPT and SPT.

2 Performance Evaluation

In the following, we will establish for which instance sizes (i.e., number of tasks) the previously implemented strategies reach a given level of performance. It is required to produce a structured code with simple and generic functions that rely on streams as much as possible. We can observe that strategies are first optimal for small instances, then their performance differ from the lower bound for medium sizes. For large sizes, the difference decreases. The objective consists in identifying for which minimum number of tasks (excluding small instances) the maximum completion time is at most at 1% of the lower bound for each strategy. For each instance size, it is necessary to compute the average ratio between the maximum completion time and its lower bound on 30 distinct instances. We assume the number of processors to be $m = 10$.

3 Parallelization

How to best parallelize the previous performance evaluation?