Learning objective: study classical heuristics and anomalies for the scheduling problem with task precedence \( P|\text{pred}|C_{\text{max}} \).

All the exercises are essential.

Before starting, we introduce some concepts:

- The *bottom level* of a task is the longest length from this task to the sink, including its own cost.
- The *critical path scheduling* algorithm first sorts the tasks by their bottom-level in non-increasing order. Each task becomes available as soon as their predecessors are completed. The algorithm schedules each available task on any available processor (i.e., neither computing, nor transmitting data).
- Contention are ignored in the communication model: the cost to transfer data is independent of what occurs on the network.

## 1 Task graph scheduling heuristics

Consider the following DAG where a pair \( X/w \) means that task \( X \) has weight \( w \). For instance A has weight 3.

![DAG Diagram]

**Exercise 1: Scheduling without communications**

We first disregard the labelling of the edges and assume communications to come for free.

Compute the bottom level for each node.
Schedule the task graph on 3 processors using a list heuristic. What is the makespan of our schedule? Is it optimal?

This is optimal because the bottom level is a lower bound on the makespan. It is therefore not possible to obtain a better makespan.

Exercise 2: Critical path scheduling
From now on we will consider the communication costs which have to be accounted for when two adjacent tasks are scheduled on different processors.

How communications should be taken into consideration when computing the bottom level?

As we do not known the allocation yet, we must assume the worst-case: communication always occur between each dependent tasks. This amounts to assume one distinct processor per task.

Compute the bottom level for each node.

Schedule the task graph on 3 processors using the critical path heuristic. What is the makespan of our schedule?

18 (A and C on processor 1, E and G on 2).

Exercise 3: Modified critical path scheduling
Sometimes, it is worth waiting to schedule a task on a busy processor rather then using the first processor available.

Which wrong decision, made in the previous section, would be avoided by using this new heuristic?

We would not put task E on another processor, but wait for task C to finish.

Using this approach, propose a new schedule for our task graph with 3 processors. What is its makespan?

It reduces to 16.

2 List scheduling anomalies

Consider the following DAG with two components.

Exercise 4: No anomalies
What is the makespan achieved by critical path list scheduling with 2 processors? Is it optimal?


Exercise 5: Anomalies on weights
Assume that each task weight is decreased by one unit (now A has weight 7, B has weight 1, and so on). Show that the makespan achieved by critical path list scheduling increases. Show that, somewhat shockingly, it is impossible to get a lower makespan than before with a list scheduling algorithm.

35: A, F and I on processor 1.

Exercise 6: Anomalies on processors
Going back to original task weights, assume that we have 3 processors. Show that the makespan achieved by critical path list scheduling increases. Show that the makespan achieved by any list scheduling algorithm, shockingly again, increases.