# Problem 1 | rj, dj | -

- This is another optimal job scheduling problem.
- The notation means:
    - 1 — single machine
    - $r_j$ — for each job a release time is given before which it cannot be scheduled, default is 0.
    - $d_j$ — for each job a due date is given.
    - '-' — means there is no objective function, the task is to simply produce a feasible scheduling
- The problem description is: We have a single processor and a set of tasks, where the j-th task takes $p_j$ time to complete and it needs to be completed in the interval $[r_j, d_j]$. Is is possible to perform all tasks within their intervals?
- This problem is NPH, because you can reduce set partition problem to it.

## Reducing set partition

- Given a set of integers $A = \{a_1, a_2, \ldots, a_n\}$ decide whether it can be partitioned into two subsets $A_1$, $A_2$ of equal sums.
- Let's create a job for each integer in $A$ setting the $r_i = 0$, $d_i = 3T$, $p_i = 2a_i$, where $T = \sum_{i=1}^{n} a_i$ (total sum of $A$) and creating an extra job with $r_{n+1} = T$, $d_i = 2T$, $p_i = T$.
- If we can schedule all the jobs that means we partitioned the set $A$ into two subsets of equal sums, since the extra tasks needs to be performed exactly at $T$ until $2T$, and other tasks need to be performed in two intervals of equal length.
- That means we can reduce the set partition problem intro our job scheduling problem.