BICRITERION SCHEDULING PROBLEM

The main idea of the algorithm is that at each decision point the set of schedulable jobs is inspected (i.e. these jobs that can be scheduled without violating their due dates). Then, the largest job is selected to be scheduled next. Note that the algorithm uses backward loading in other words it always decides upon a job to be placed in the last position.

The steps of the algorithm are as follows:

Step 0: Set $\Delta = \sum_{i=1}^{n} p_i$

Step 1: Let $D_i = d_i + \Delta$ for all i

Step 2: Set $R = \sum_{i=1}^{n} p_i$, $k = n$

Step 3: Find a job i^* such that $p_{i^*} \geq p_i$ from all i satisfying $D_i \geq R$, and if there are multiple i^* satisfying $p_{i^*} \geq p_i$, then select the one that satisfying $D_{i^*} \geq D_i$ for all i *(Break ties arbitrarily).*

Assign job i^* to position k.

If there is no job satisfies these conditions, go to Step 8.

Step 4: Set $R = R - p_{i^*}$

$I = I - \{i^*\}$

$k = k - 1$

If $k = 0$, go to Step 5. Else go to Step 3.

Step 5: Compute completion time of jobs (C_i).

$C_i = C_{i-1} + p_i$

Step 6: Compute $T(i) = \max(0, C_i - d_i)$ for all i.

Then, compute $T(\pi^*) = \max\limits_{i=1,\ldots,n} \{T(i)\}$ and $H(\pi^*) = \sum_{i=1}^{n} C_i$

(An iteration is completed.)

Step 7: Set $\Delta = T(\pi^*) - 1$. Go to Step 1.

Step 8: Stop.

Consider there are ***n* jobs** to be sequenced on a single processor. All jobs are simultaneously available and they have **processing time (p_i)** and **due date (d_i)**.

The objective functions are:

- Minimizing the flow time (H)
- Minimizing the maximum tardiness (T)

This problem (P) is formulated as follows (Wassenhove and Gelders, 1978):

$\prod$: the set of schedules

I : the set of jobs, $i = \{1, 2, \ldots, n\}$

π : a schedule

C_i : the completion time of job i (given a schedule)

$T(\pi)$: the maximum tardiness of schedule π

$H(\pi)$: the total flow time of schedule π

$$\min_{\pi \in \prod} \sum_{i=1}^{n} C_i = \min_{\pi \in \prod} H(\pi) \qquad (1)$$

$$\min_{\pi \in \prod} \max_{i=1,\ldots,n} \{\max(0, C_i - d_i)\} = \min_{\pi \in \prod} T(\pi) \qquad (2)$$

A schedule $\pi^* \in \prod$ is <u>efficient</u> in problem (P) if there exists no $\pi \in \prod$ such that $H(\pi) \leq H(\pi^*)$ and $T(\pi) \leq T(\pi^*)$ where at least one relation holds with strict inequality.

A schedule π_i is said to <u>dominate</u> a schedule π_m when $H(\pi_i) \leq H(\pi_m)$ and $T(\pi_i) \leq T(\pi_m)$ where at least one relation holds with strict inequality.

There are 10 jobs to be scheduled. The processing times and due dates are given in the following table.

Table 1: Processing times and due dates of jobs

Job ID	p_i	d_i
1	9	32
2	9	49
3	6	7
4	7	25
5	2	55
6	4	9
7	7	54
8	2	40
9	7	52
10	8	51

You will be using **SchedulingProblem.xlsx** including the following sheets.

a. **Jobs:** This sheet includes the processing time (p_i) and due date (d_i) information for each job.

b. **Iterations:** This sheet includes the iteration steps of the algorithm. There are blocks for iterations, i.e. range A1:M16 consists the output of the first iteration. In each iteration block you should fill the following rows by coding the algorithm:

   1. Δ should be calculated.
   2. p_i and d_i values should be taken from "Jobs" sheet.
   3. D_i values should be calculated.
   4. R should be calculated, and corresponding job ID and processing time should be written in *Job ID* and p_i rows, respectively.
   5. *Order* row stands for the order of jobs in the schedule. For example, if the first job is in third order, then for the first job, 3 should be written in the *Order* row.
   6. C_i values should be calculated.
   7. d_i values should be taken from "Jobs" sheet.
   8. $T(i)$ values should be calculated.
   9. Finally, $H(\pi^*)$ and $T(\pi^*)$ should be calculated.