Problème 2 : Permutation Flowshop avec contraintes.

On vous demande d'effectuer cet exercice de modélisation à l'aide de Gurobipy.

Dans le problème classique du *permutation flowshop*, n tâches doivent être traitées sur un ensemble de m machines dans le même ordre de passage. Chaque tâche peut être traitée sur au plus une machine au même moment et chaque machine peut traiter au plus une tâche à la fois à un instant donné. De plus, les machines sont traversées dans l'ordre : chaque tâche débute sur la machine 1, puis sur la machine 2, et ainsi de suite jusqu'à la machine m.

Le temps de traitement de la tâche i sur la machine k est noté d_{ki}, avec $i = 1, ..., n$ et $k = 1, ..., m$. La date d'échéance de la tâche i est quant à elle notée e_i.

Dans diverses situations industrielles, des contraintes supplémentaires permettent à la modélisation d'être plus réaliste. Dans ce projet, nous allons considérer qu'entre l'exécution du job i sur la machine $k-1$ et son exécution sur la machine k, il faut un décalage d'au moins $\theta_{ki}^{\min}$ et d'au plus $\theta_{ki}^{\max}$, avec $k = 2, ..., m$.

On vous demande :

◇ Implémenter une fonction `def flowshopwithtimelags(d,e,tlagsmin,tlagsmax):` où les paramètres d'entrée sont

- `d`, une liste de listes contenant les durées d'exécution des n tâches sur les m machines,
- `e`, une liste de longueur n avec les dates d'échéances des tâches,
- `tlagsmin`, une liste de listes contenant les time lags minimaux pour les n tâches sur les m machines
- `tlagsmax`, une liste de listes contenant les time lags maximaux pour les n tâches sur les m machines

et dont le paramètre de sortie est

- `ordonnancement`, une permutation optimale des nombres $\{1, ..., n\}$ représentant l'ordre dans lequel il faut traiter les tâches afin de minimiser N_t, le nombre de tâches en retard.
- `Ntmin`, le nombre minimal de tâches en retard.

◇ Implémenter une fonction `def flowshopwithtimelagslexico(d,e,tlagsmin,tlagsmax)` où les paramètres d'entrée sont sont les mêmes que précédemment, mais où l'objectif consiste à déterminer la permutation minimisant $C_{\max}$ **parmi** l'ensemble des permutations possédant un nombre de tâches en retard minimal.

Testez votre code sur les instances fournies, et calculez le temps de résolution.