Corrigé du dm 1

1. Ordonnancement pour minimiser le nombre de tâches en retard

i. Les tâches en retard peuvent être exécutées dans n'importe quel ordre après que toutes les tâches non en retard ont été exécutées. De plus, lorsque l'ordre des tâches a été fixé, un ordonnancement au plus tôt, c'est-à-dire dans lequel la première tâche commence à l'instant 0 et les autres tâches commencent à la fin de leur prédécesseur, est clairement optimal. Il faut donc trouver l'ordre des opérations non en retard. Dans un ordonnancement optimal, s'il y a deux tâches non en retard consécutives T_i et T_j telles que T_i précède T_j et $e_i \geq e_j$, on peut inverser ces deux tâches sans modifier la position des autres tâches et, dans ce nouvel ordonnancement T_i et T_j ne sont pas en retard. Il existe donc un ordonnancement optimal dans lequel les tâches non en retard sont ordonnées selon leur date d'échéance e_i.

ii. Soit t_{pj} la date de fin au plus tôt des sous-ensembles de $\{T_1, \cdots, T_k\}$ de poids supérieur ou égal à p. On conviendra que $t_{pj} = \infty$ s'il n'y a pas au moins un tel sous-ensemble. On a ainsi $t_{0j} = 0$ et $t_{p0} = \infty$ si $p > 0$. On s'intéresse maintenant à la relation de récurrence. On considère le sous-ensemble S de $\{T_1, \cdots, T_{j+1}\}$ et de poids supérieure ou égale à p et qui peut être exécuté le plus rapidement (S se termine par définition à $t_{p,j+1}$). Soit T_{j+1} est dans S, soit il ne n'est pas. Si $T_{j+1} \notin S$, alors $S \subseteq \{T_1, \cdots, T_k\}$ et S est donc le sous-ensemble de $\{T_1, \cdots, T_j\}$ et de poids supérieure ou égale à p qui peut être exécuté le plus rapidement, S se termine donc à t_{pj}. Si $T_{j+1} \in S$,c ette tâche a la plus grande date d'échéance et on peut l'ordonnancer en dernier. L'ensemble des tâches qui la précèdent a une poids d'au moins $p - p_{j+1}$ et cet ensemble doit pouvoir être exécuter le plus rapidement possible. On a donc :

$$t_{p,j+1} = \begin{cases} \min(t_{pj}, t_{p-p_{j+1},j} + d_{j+1}) & \text{si } t_{p-p_{j+1},j} + d_{j+1} \leq e_{j+1} \\ t_{pj} & \text{sinon} \end{cases}$$

Puisqu'il n'y a pas de sous-ensemble de poids strictement supérieur à $\sum p_j$, on peut arrêter la récurrence quand p atteint cette valeur, ce qui donne un algorithme en $O(n \sum p_i)$. La plus grande valeur p pour laquelle $t_{pn} < \infty$ donne le poids maximal des tâches sans retard. L'ensemble correspondant peut être déduit des valeurs t_{pj} en prenant les j tels que$t_{pj} \neq t_{p,j-1}$.