**Numéro d'anonymat :**

**UE COMPLEX.**
**M1 Informatique.**

**Examen du 9 novembre 2023.**

*Seule une feuille A4 portant sur les cours et les TD est autorisée, tout autre document est interdit. Téléphones portables éteints et rangés dans vos sacs. Le barème est indicatif et est susceptible d'être modifié. Toutes les réponses doivent être justifiées.*

## **Exercice** 1 (3 points)

Répondre aux questions à choix multiples suivantes. On supposera dans cet exercice que $P \neq NP$. Chaque bonne réponse apportera 0.5 point et chaque mauvaise réponse retranchera 0.5 point au total de l'exercice (qui restera tout de même au minimum 0). Une absence de réponse ne retranchera pas de point au total.

| **Questions** | **Réponses** |
|---|---|
| **1.** Soient $A$ et $B$ deux problèmes de décision tels que $A \leq_P B$. Si $A \in P$, a-t-on nécessairement $B \in P$ ? | ☐ Oui. |
| | ☐ Non. |
| **2.** Soient $A$ et $B$ deux problèmes de décision tels que $A \leq_P B$. Si $B \in P$, a-t-on nécessairement $A \in P$ ? | ☐ Oui. |
| | ☐ Non. |
| **3.** Soient $A$ et $B$ deux problèmes de décision tels que $A \leq_P B$. Si $A$ est NP-complet, $B$ est-il nécessairement NP-complet ? | ☐ Oui. |
| | ☐ Non. |
| **4.** Si $A$ et $B$ sont deux problèmes NP-complets, a-t-on nécessairement $B \leq_P A$ ? | ☐ Oui. |
| | ☐ Non. |
| **5.** Un algorithme de branch and bound permet d'obtenir une solution optimale à un problème NP-difficile en un temps polynomial en la taille de l'entrée. | ☐ Oui. |
| | ☐ Non. |
| **6.** Un algorithme de branch and bound pour un problème de minimisation nécessite de connaître une borne inférieure de la valeur d'une solution d'un sous-arbre enraciné en un nœud donné. Sans quoi, l'algorithme explorera tout l'arbre de recherche. | ☐ Oui. |
| | ☐ Non. |

**Exercice** 2 (4 points)

Décrire formellement une machine de Turing reconnaissant les mots formés sur le langage $\{\#, a, b\}$, commençant par un $'\#'$ et suivis de $'a'$ et de $'b'$, et qui contiennent un nombre pair de $'b'$ ainsi que au moins trois $'a'$ à la suite. Par exemple, le mot $\#aabaaa$ ne sera pas accepté car il ne contient pas un nombre pair de $b$, et le mot $\#abaabbb$ ne sera pas accepté car il ne contient pas le motif $aaa$. Au contraire, les mots $\#abaaabbaba$ et $\#aaaa$ seront acceptés.

Vous indiquerez l'alphabet du ruban et dessinerez le diagramme d'états.

**Exercice 3** (13 points)

Le problème d'ordonnancement d'atelier est un problème classique en théorie de l'ordonnancement. Ce problème formalise des problèmes rencontrés dans des chaînes de fabrication de pièces, quand les pièces doivent passer sur différentes machines, et quand le but est de terminer la fabrication des pièces au plus tôt.

Une instance du problème d'ordonnancement d'atelier consiste en $m$ machines $M_1, M_2, \ldots, M_m$ et $n$ tâches $T_1, T_2, \ldots, T_n$. Chaque tâche $T_j$ consiste en $m$ opérations $O_{i,j}$, avec $i \in \{1, \ldots, m\}$. L'opération $O_{i,j}$ doit être effectuée par la machine $M_i$ et demande un temps $p_{i,j} \in \mathbb{N}$. Une fois commencée, une opération ne peut pas être interrompue (elle s'exécute donc pendant $p_{i,j}$ unités de temps). Pour chaque tâche, l'ordre dans lequel sont effectuées ses $m$ opérations n'est pas fixé à l'avance : tous les ordres sont acceptables. Ainsi, les opérations des différentes tâches ne sont pas nécessairement faites dans le même ordre (par exemple, la première opération de la tâche $T_x$ peut être $O_{1,x}$ – sur la machine $M_1$ –, tandis que la première opération de la tâche $T_y$ peut être $O_{2,y}$ – sur la machine $M_2$ –).

Un ordonnancement consiste à affecter à chaque opération $O_{i,j}$ un intervalle de temps de durée $p_{i,j}$ de manière à ce que :

— aucune tâche ne peut être exécutée simultanément sur plusieurs machines différentes. Autrement dit, les intervalles d'exécutions des opérations d'une tâche sont tous disjoints.

— Une machine ne peut exécuter qu'une seule tâche à la fois. Autrement dit, à un moment donnée, soit une machine est inactive (elle n'exécute aucune opération), soit elle est en train d'exécuter une et une seule opération.

La date de fin d'un ordonnancement est la date à laquelle toutes les tâches ont été exécutées (i.e. toutes les opérations de toutes les tâches sont terminées).

Le problème de l'ordonnancement d'atelier, noté OrdoAtelier, consiste à retourner un ordonnancement réalisable *de date de fin minimale*. On notera $OPT$ la date de fin d'un ordonnancement de date de fin minimale.

Les figures 1 et 2 représentent deux ordonnancements réalisables pour l'instance décrite dans la table 1. L'ordonnancement dessiné dans la figure 2 est optimal : pour cette instance, $OPT = 12$.

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| $M_1$ | 4 | 1 | 2 | 1 |
| $M_2$ | 4 | 3 | 1 | 4 |
| $M_3$ | 4 | 2 | 1 | 2 |

Table 1 – Une instance du problème d'OrdoAtelier à 3 machines et 4 tâches. La case à l'intersection de ligne notée $M_i$ et colonne notée $T_j$ représente la durée $p_{i,j}$ de l'opération $O_{i,j}$ effectuée par la tâche $T_j$ sur la machine $M_i$.

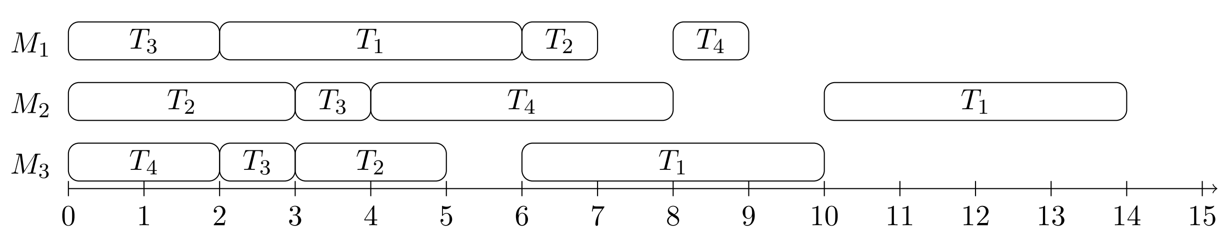


Figure 1 – Un ordonnancement réalisable pour l'instance décrite dans la table 1.

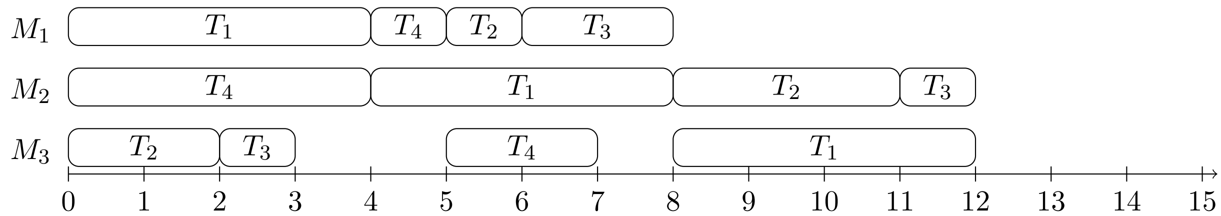


FIGURE 2 – Un ordonnancement optimal pour l'instance décrite dans la table 1.

**Question** 1 (2/13) — Dessiner un ordonnancement optimal pour l'instance composée de 3 machines et des 5 tâches $T_1, \dots, T_5$, telles que pour chaque tâche $j \in \{1, \dots, 5\}$ et pour chaque machine $i \in \{1, \dots, 3\}$, on a $p_{i,j} = j$ (i.e. chacune des opérations de la tâche $T_j$ s'exécute en $j$ unités de temps). Quelle est la date de fin de l'ordonnancement obtenu ?

On considère dans la question suivante la version décision de notre problème ORDOATELIER. Ce problème, noté ORDOATELIERDEC est le suivant : étant donnée une instance de ORDOATELIER ($m$ machines et $n$ tâches ayant chacune $m$ opérations – la durée de l'opération $O_{i,j}$ étant $p_{i,j}$), et un entier $B$, la question est : existe-t-il un ordonnancement réalisable ayant une date de fin inférieure ou égale à $B$ ?

**Question** 2 (0.5/13) — Montrer que ORDOATELIERDEC appartient à NP. Justifiez en une phrase votre réponse.

On propose de réduire le problème de PARTITION, qui est NP-complet, au problème ORDOATELIERDEC. On rappelle que le problème de PARTITION est le suivant : étant donné un ensemble $S = \{a_1, \ldots, a_x\}$ de $x$ nombres tels que $\sum_{a_i \in S} a_i = 2A$, la question est : est-il possible de partitionner $S$ en deux sous-ensembles $S_1$ et $S_2$ tels que $\sum_{a_i \in S_1} a_i = \sum_{a_j \in S_2} a_j = A$ ?

La réduction est la suivante. L'instance de OrdoAtelierDec correspondant à l'instance de Partition est constituée de $m = 3$ machines et de $n = x + 1$ tâches. Pour toute tâche $i \in \{1, \ldots, x\}, p_{1,i} = p_{2,i} = p_{3,i} = a_i$. En ce qui concerne la tâche $T_{x+1}$, on a $p_{1,x+1} = p_{2,x+1} = p_{3,x+1} = A$. On fixe $B = 3A$.

**Question** 3 (4/13) —

a) Montrer que si la réponse au problème Partition est "Oui" alors la réponse au problème OrdoAtelierDec sur l'instance correspondante est "Oui".

b) Montrer que si la réponse au problème OrdoAtelierDec est "Oui" alors la réponse au problème Partition est "Oui".

*Indice :* vous observerez le placement de la tâche $T_{x+1}$ dans un ordonnancement de date de fin au plus $3A$.

**Question** 4 (0.5/13) — Déduire des questions précédentes que OrdoAtelierDec est NP-complet.

On s'intéresse à l'algorithme glouton suivant, que l'on appellera AlgoGlouton : pour chaque machine $M_i$, pour $i$ allant de 1 à $m$, si $M_i$ est disponible, lui affecter une tâche qu'elle n'a pas déjà ordonnancée et qui n'est pas ordonnancée (ou commencée) au même moment sur une autre machine (si plusieurs tâches sont possibles, en prendre une parmi celles-là, peu importe laquelle). Ainsi, à chaque instant, soit une machine ordonnance une tâche, soit toutes les tâches qu'elle n'a pas encore ordonnancées sont en cours d'exécution sur une autre machine, soit la machine a terminé d'exécuter toutes ses tâches.

**Question** 5 (1/13) — En remarquant que l'ordonnancement dessiné dans la figure 1 peut être obtenu par l'algorithme AlgoGlouton, que peut-on dire du rapport d'approximation de cet algorithme ?

**Question** 6 (2/13) — Montrer que

a) Pour tout $j \in \{1, \ldots, n\}$, $OPT \geq \sum_{i\in\{1,\ldots m\}} p_{i,j}$

b) Pour tout $i \in \{1, \ldots, m\}$, $OPT \geq \sum_{j \in \{1, \ldots n\}} p_{i,j}$

**Question** 7 (2/13) — Déduire de la question précédente un rapport d'approximation de l'algorithme AlgoGlouton. Justifiez votre réponse.

*Indication :* On pourra considérer la machine $M_i$ sur laquelle se termine la dernière tâche, que l'on notera $T_j$. On pourra noter $C_{\max}$ la date de fin de l'ordonnancement obtenu.

**Question** 8 (1/13) — Il a été montré en 1997 par des chercheurs américains et néerlandais [1] que :

— Le problème qui consiste à savoir s'il existe un ordonnancement de date de fin au plus 3 appartient à la classe $P$.

— Le problème qui consiste à savoir s'il existe un ordonnancement de date de fin au plus 4 est NP-complet.

Que peut-on en déduire d'un point de vue de l'approximation du problème OrdoAtelier ?

1. Ce résultat a été publié dans l'article : *Short Shop Schedules*, D. P. Williamson, L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevast'janov and D. B. Shmoys, Operations Research, volume 45, numéro 2 (mars 1997), pages 288-294.