I Algorithme de tri fusion

On rappelle le principe du tri fusion sur une liste OCaml `l` de taille n :

- On divise la liste en deux sous-listes de taille $n/2$.
- On trie chaque sous-liste de manière récursive.
- On fusionne les deux sous-listes triées pour obtenir la liste triée.

1. Écrire l'algorithme de tri fusion en OCaml. On pourra décomposer en plusieurs fonctions.
2. Déterminer la complexité dans le pire cas de votre algorithme.

Soit A un algorithme de tri sur un tableau `T` de taille n. On considère l'arbre de décision a de A, où chaque noeud interne correspond à une comparaison entre deux éléments de `T` (c'est-à-dire un test de la forme `T[i] < T[j]`). L'exécution de A sur un tableau de taille n correspond à un chemin dans cet arbre, et la feuille atteinte correspond à l'ordre final des éléments (valeur de retour de A).
On note $f(a)$ et $h(a)$ respectivement le nombre de feuilles et la hauteur de l'arbre a.

3. Montrer que $f(a) \geq n!$.
4. Montrer que $f(a) \leq 2^{h(a)}$.
5. En déduire qu'il n'existe pas d'algorithme de tri en complexité $o(n \log n)$ dans le pire cas.
6. Montrer qu'il est possible de trier un tableau de taille n dont les éléments sont des entiers compris entre 0 et $n-1$ en complexité linéaire. Est-ce en contradiction avec le résultat précédent ?
7. Soit a un arbre binaire non vide et $S(a)$ la somme des profondeurs des feuilles de a. Montrer que $S(a) \geq f(a) \log_2(f(a))$, où $f(a)$ est le nombre de feuilles de a.
8. Montrer qu'il n'existe pas d'algorithme de tri en complexité $o(n \log n)$ en moyenne.

Exercice 4 *Activation de processus (type A)*

Soit un système temps réel à n processus asynchrones $i \in [\![1, n]\!]$ et m ressources r_j. Quand un processus i est actif, il bloque un certain nombre de ressources listées dans un ensemble P_i et une ressource ne peut être utilisée que par un seul processus. On cherche à activer simultanément le plus de processus possible.

Le problème de décision `ACTIVATION` correspondant ajoute un entier k aux entrées et cherche à répondre à la question : "Est-il possible d'activer au moins k processus en même temps ?"

1. Soit $n = 4$ et $m = 5$. On suppose que $P_1 = \{r_1, r_2\}$, $P_2 = \{r_1, r_3\}$, $P_3 = \{r_2, r_4, r_5\}$ et $P_4 = \{r_1, r_2, r_4\}$. Est-il possible d'activer 2 processus en même temps ? Même question avec 3 processus.
2. Dans le cas où chaque processus n'utilise qu'une seule ressource, proposer un algorithme résolvant le problème `ACTIVATION`. Évaluer la complexité de votre algorithme.

On souhaite montrer que `ACTIVATION` est NP-complet.

3. Donner un certificat pour ce problème.
4. Écrire en pseudo code un algorithme de vérification polynomial. On supposera disposer de trois primitives, toutes trois de complexité polynomiale :
   (a) `appartient(c,i)` qui renvoie `Vrai` si le processus `i` est dans l'ensemble d'entiers `c`.
   (b) `intersecte(`$\mathtt{P}_i$`,R)` qui renvoie `Vrai` si le processus `i` utilise une ressource incluse dans un ensemble de ressources `R`.
   (c) `ajoute(`$\mathtt{P}_i$`,R)` qui ajoute les ressources $\mathtt{P}_i$ dans l'ensemble `R` et renvoie ce nouvel ensemble.

En théorie des graphes, le problème `STABLE` se pose la question de l'existence dans un graphe non orienté $G = (S, A)$ d'un ensemble d'au moins k sommets ne contenant aucune paire de sommets voisins. En d'autres termes, existe-t-il $S' \subset S$, $|S'| \geq k$ tel que $s, p \in S' \Rightarrow (s, p) \notin A$?

5. En utilisant le fait que `STABLE` est NP-complet, montrer par réduction que le problème `ACTIVATION` est également NP-complet.

Commentaires du jury

1. On attend du candidat une réponse en oui / non avec une précision de quels processus activer dans le cas où la réponse est oui et une très rapide justification dans le cas où la réponse est non. Une réponse orale suffit.
2. De multiples réponses sont possibles sur cette question et sont acceptées du moment qu'elles sont clairement décrites et correctement analysées. On n'attend pas une complexité optimale.
3. Une courte phrase décrivant la nature d'un tel certificat et indiquant sa polynomialité en la taille de l'entrée suffit à obtenir tous les points.
4. On attend un algorithme utilisant à bon escient les primitives proposées par le sujet.
5. Le jury est attentif au fait que tous les arguments soient bien présents : description de la réduction, preuve que c'en est une et justification de son caractère polynomial pour le caractère NP-difficile ; caractère NP pour pouvoir conclure quant à la NP-complétude.