# Option Informatique

## Présentation du sujet

Le sujet 2016 de l'option informatique traite de l'étude d'un algorithme de tri. L'intérêt de cet algorithme est d'avoir une bonne complexité dans le cas où la liste est presque triée. La mise en œuvre de cet algorithme permet la manipulation d'arbres, de listes, de tableaux. Quelques études de complexité sont également demandées. Le problème est raisonnablement progressif, nécessite une bonne lecture des hypothèses et des représentations de données. La longueur du texte est volontairement raisonnable, ce qui permet aux candidats d'aborder l'ensemble du problème.

## Analyse globale des résultats

Le sujet a été correctement compris. Comme prévu, un nombre significatif de candidats a pu traiter le problème dans son ensemble. Le nombre de copies très faibles est réduit. Le caractère progressif de la mise en œuvre de l'algorithme a permis d'éviter les blocages. Les signatures des fonctions `Caml` étaient imposées ; elles ont été bien respectées.

Malheureusement, sur la fin du problème, elles n'étaient pas logiques par rapport à l'esprit des questions, mais nous avons avec plaisir constaté que cela n'a pas eu l'air de déstabiliser les candidats, qui ont corrigé d'eux-mêmes. Nous avons encore observé des confusions entre `Caml` et `Python`, mais moins que l'année dernière. Par contre, peu d'évolution concernant les autres difficultés signalées. Beaucoup de candidats compliquent le sujet en n'utilisant pas les éléments précédemment définis, omettent de numéroter les questions, écrivent des codes complexes sans aucun commentaire, ou sur plusieurs pages, ou encore en multipliant inutilement le nombre de fonctions auxiliaires rendant la lecture très difficile. La syntaxe `Caml` est parfois approximative. Les calculs de complexité pas ou mal justifiés. Cela dit, le niveau moyen est satisfaisant.

## Commentaires sur les réponses apportées et conseils aux futurs candidats

Après l'introduction sur les motivations du problème, le sujet commence par des algorithmes sur les listes puis les arbres. Les principales erreurs sont liées à un non respect des indications. Pour que les complexités soient correctes, il ne faut pas introduire de parcours de liste inutile, comme compter le nombre d'éléments dans la liste simplement pour tester l'égalité éventuelle de ses deux premiers éléments, ou ajouter des éléments en fin de liste quand il suffit de les mettre au début. On observe cette année encore de grandes difficultés dans l'usage des types construits : affectations incorrectes, confusion entre la déclaration de type `Vide` de l'arbre et la liste vide `[]`, ... De même utilisation parfois fausse (exemple `|q::q::a` pour tester l'égalité des deux premiers termes de la liste), ou peu efficace des `match`. Il faut également veiller à la cohérence des codes. Par exemple, dans la fonction `min_tas`, il n'était pas demandé de traiter le cas du tas vide, mais il ne faut pas alors, dans la fonction `min_quasi` qui suit, appeler `min_tas` sur un tas vide. Bien sûr, le jury accepte la modification de l'une ou l'autre des fonctions, tant que la cohérence globale reste assurée. Ainsi, il faut que les candidats prennent le temps d'analyser l'ensemble des relations entre les différentes fonctions programmées.

La suite du problème étudie la décomposition parfaite d'un entier. L'objectif était de permettre aux candidats de comprendre l'algorithme suivant sur la liste de tas, en particulier de constater que la décomposition se prête à un calcul récursif. La méthode est ensuite utilisée pour construire efficacement la liste de tas. Pour bien suivre l'ensemble, il faut que le candidat prenne le recul suffisant pour comprendre où on cherche à le conduire. Comme le but est d'avoir une bonne complexité globale, il ne faut pas chaque fois tout reconstruire, ou trier totalement les tas. Ce défaut de compréhension est encore plus visible dans le tri des racines, puisque beaucoup de candidats cherchent à `percoler` tous les tas. On peut également conseiller aux candidats d'éviter l'usage systématique et mal géré de la récursivité terminale, qui a conduit certains à avoir des listes à l'envers ne permettant plus à l'algorithme décrit de fonctionner. De plus, il convient de vérifier la cohérence entre la description d'un algorithme, sa transcription en `Caml` et son résultat sur des exemples. De nombreux candidats semblent traiter ces trois éléments de façon totalement indépendante avec des solutions différentes pour chacun.

L'algorithme se termine avec l'extraction, qui produit la liste triée. Globalement bien comprise par les candidats qui l'ont abordée, cette partie reprend les outils précédents. Les calculs de complexité sont parfois effectués de façon très approximative.

La dernière partie du problème reprend le même algorithme, mais en réduisant la complexité spatiale à l'aide d'un tri en place dans un tableau. Moins abordée que le reste, on a pu constater les erreurs usuelles sur les décalages d'indices, et des difficultés pour certains avec les enregistrements.

Dernier conseil aux candidats, toujours le même, ne pas oublier que ce sont des humains qui corrigent. Comme nous l'avons déjà signalé, cela oblige à écrire des codes clairs, mais également utiliser des notations courantes ou explicitées, et bien sûr conserver celles du texte quand elles sont données.

## Conclusion

La pratique sur machine est indispensable pour s'approprier les réflexes de programmation. Bien que le temps de formation soit limité, on ne peut qu'encourager vivement les candidats à faire l'effort nécessaire. Dans le cadre de l'épreuve, il est également souhaitable de rechercher la lisibilité et la simplicité des programmes. Pour cela, il faut avoir des idées claires et une bonne capacité d'adaptation. Une lecture précise des hypothèses et des indications est absolument nécessaire.

Le niveau global des candidats est satisfaisant. Une fraction notable des candidats nous a rendu des copies tout à fait excellentes, avec des codes élégants et clairs, ce qui est remarquable sans l'aide d'un compilateur. Le jury félicite les candidats qui s'investissent ainsi dans la discipline.