# Composition d'Informatique - A - (XULCR)

Filière MP spécialité Info

## 1 L'épreuve

Il s'agissait dans ce sujet de mettre en œuvre des algorithmes efficaces pour colorier des graphes non orientés. Le coloriage de graphes étant connu pour être NP-complet, l'objectif dans ce sujet était de calculer des solutions optimales pour un sous-problème plus simple, le coloriage à deux couleurs, et ensuite de calculer efficacement des solutions non optimales pour le problème général.

La partie 1 avait pour but de familiariser les candidat(e)s avec la notion de coloriage de graphe. Deux questions simples sur des exemples étaient proposées, et il fallait écrire un algorithme simple vérifiant la propriété de coloriage pour un graphe étiqueté. Il était ensuite demandé de démontrer une borne supérieure exponentielle en temps pour le problème de coloriage. Dans la partie 2, le sous-problème du 2-coloriage était introduit. Il était demandé d'écrire un algorithme de parcours en profondeur du graphe pour 2-colorier un graphe supposé 2-coloriable.

Dans la partie 3, le problème plus général, et plus difficile, du coloriage à plus de 3 couleurs était abordé. Un algorithme glouton, produisant une solution non optimale, était proposé. Il était demandé de donner le résultat de cet algorithme sur deux exemples. Il était ensuite demandé d'écrire un programme le réalisant en temps quadratique, de prouver sa correction et de prouver une borne supérieure sur le nombre de couleurs utilisées par cet algorithme. Une version optimisée de cet algorithme était ensuite demandée, utilisant un tri préalable des sommets du graphe par degrés décroissants.

Dans la partie 4, enfin, un algorithme de coloriage un peu meilleur était étudié : l'algorithme de Wigderson, pour des graphes 3-coloriables. Là encore, il s'agit d'un algorithme polynomial en temps, produisant une solution non optimale. Il était demandé de prouver sa correction et de prouver une borne supérieure sur le nombre de couleurs utilisées par l'algorithme, à l'aide des résultats obtenus dans la partie 3. Enfin, il était demandé d'écrire un programme réalisant cet algorithme. Une dernière question, plus ouverte, demandait aux candidat(e)s de généraliser cet algorithme au cas des graphes $k$-coloriables, pour une valeur de $k$ supérieure à 3.

Les notes des 1072 copies se répartissent selon le tableau suivant, avec une moyenne de 9,80 et un écart type de 4,13.

| | | |
|---|---|---|
| $0 \leq N < 4$ | 77 | 7,2 % |
| $4 \leq N < 8$ | 295 | 27,5 % |
| $8 \leq N < 12$ | 374 | 34,9 % |
| $12 \leq N < 16$ | 234 | 21,8 % |
| $16 \leq N \leq 20$ | 92 | 8,6 % |
| Nombre de copies | 1072 | 100 % |
| Note moyenne | 9,80 | |
| Écart-type | 4,13 | |

## 2 Remarques générales

**Preuves de correction.** Trop de copies se contentent de justifier la correction d'un algorithme en le paraphrasant en français ou en indiquant qu'« on voit bien que ». Les correcteurs attendent des preuves rigoureuses, avec une hypothèse de récurrence (un invariant de boucle, dit autrement) correctement identifiée et formulée. Bien entendu, un raisonnement approximatif à base de « de proche en proche » ou de points de suspension n'est pas considéré comme une preuve par récurrence rigoureuse.

**Programmation.** Concernant les questions de programmation, les candidat(e)s doivent être conscients du fait qu'une réponse longue doit être expliquée en détail et que très souvent un programme très long contient un grand nombre d'erreurs. Découper un programme un peu long en plusieurs fonctions est une bonne initiative, mais appeler ces différentes fonctions `aux`, `aux2`, `aux3`, etc., ne l'est pas. Certaines copies utilisent une couleur différente pour écrire leurs programmes et cette initiative est très appréciée des correcteurs. Il en va de même pour le soin apporté à l'indentation du code, qui en facilite grandement la compréhension.

Beaucoup de candidat(e)s utilisent une boucle « `for i = 0 to n` » pour parcourir un tableau de taille `n`, ce qui est incorrect. Quelques copies font un mélange farfelu de syntaxe Caml et Python, par exemple en recourant à une fonction `range` pour parcourir un tableau. Certains candidat(e)s perdent du temps à ré-écrire des fonctions données dans l'énoncé (comme `list_length`), pour un résultat parfois faux.

L'énoncé demandait de justifier les complexités en temps des algorithmes, mais beaucoup de candidat(e)s oublient de le faire. Affirmer que la complexité est en $O(n)$ sans en expliquer les raisons n'est pas une justification.

Dans plusieurs questions de programmation, l'énoncé indique que « le comportement de la fonction est laissée au choix du candidat » dans certains cas pathologiques. De nombreux candidat(e)s écrivent de longs programmes pour gérer ces cas pathologiques, alors qu'il faut au contraire interpréter cette formulation comme une invitation à les ignorer.

## 3 Commentaire détaillé

Pour chaque question, sont indiqués entre crochets le pourcentage de candidat(e)s ayant traité la question et le pourcentage de candidat(e)s ayant obtenu la totalité des points. Beaucoup de candidat(e)s sont parvenus jusqu'à la dernière question mais aucun n'a su traiter toutes les questions correctement. En particulier, les questions 4, 6, 11 et 14 ont été très mal traitées, quand elles n'ont pas été tout simplement ignorées par les candidat(e)s.

Beaucoup de candidat(e)s supposent implicitement que le graphe n'a pas de boucle. Même si cette supposition est raisonnable dans un contexte de coloriage de graphes, elle ne paraît pas explicitement dans le sujet. Ce point n'a pas été sanctionné.

### Partie I

**Question 1 [100% - 99%].** Il s'agit là d'une question très simple, de nature à se familiariser avec la notion de coloriage. À noter que la définition de « colorié » donnée dans l'énoncé est pour le moins troublante, mais seuls quelques candidat(e)s le remarquent.

**Question 2 [100% - 53%].** Beaucoup de candidat(e)s oublient de justifier qu'il n'est pas possible de 2-colorier le graphe. Attention à proposer un coloriage lisible : utiliser cinquante

nuances de gris n'est pas une bonne idée. Utiliser des lettres ou des noms de couleurs en toutes lettres est une bonne initiative.

**Question 3 [100% - 37%].** Pour éviter des erreurs d'indice dans les tableaux, il ne faut tester la correction de l'étiquetage qu'après avoir testé que sa taille est au moins égale au nombre de sommets.

**Question 4 [74% - 18%].** La justification des bornes établies est plus importante que la description de l'algorithme utilisé pour respecter ces bornes. On s'attend à ce que le candidat mentionne trois points importants : le nombre chromatique d'un graphe à $n$ sommets est au plus $n$ ; le nombre d'étiquetages à $k$ couleurs est en $k^n$ ; et chaque étiquetage peut être testé en $O(n^2)$ par la question précédente.

## Partie II

**Question 5 [99% - 90%].** Un graphe sans arêtes est biparti, 1-coloriable et donc également 2-coloriable. Il n'y a pas lieu d'en faire un cas particulier dans la preuve. Il s'agit cependant d'une question très bien traitée, en général rédigée avec beaucoup de soin.

**Question 6 [86% - 12%].** Le parcours en profondeur est un algorithme au programme. Il est particulièrement simple à écrire ici, car l'étiquetage tient lieu en même temps de marquage des sommets déjà visités. Pourtant, très peu de copies proposent un code correct. Par ailleurs, la bonne alternance des couleurs a souvent posé un problème supplémentaire.

## Partie III

**Question 7 [99% - 53%].** Une réponse lisible est appréciée. Un dessin, avec une couleur associée à chaque sommet, est facilement lisible. Une table de correspondance (sommet/couleur), dans le désordre, l'est moins. Lorsque le choix des couleurs utilisées ne respecte pas l'énoncé, c'est encore pire.

**Question 8 [96% - 23%].** Des candidat(e)s cherchent le minimum ou le maximum des valeurs des étiquettes, pour lui retirer ou lui ajouter un au final. La valeur qui en résulte n'a rien à voir avec la question. Certains candidat(e)s s'en remettent à un test répétitif d'appartenance à une liste de couleurs, mais cela ne permet pas d'obtenir la bonne complexité. Dans une solution utilisant un tableau auxiliaire pour marquer les couleurs utilisées, il doit être justifié que la dernière boucle n'excédera pas la taille de ce tableau ou bien il faut écrire un code défensif.

**Question 9 [97% - 47%].** Cette question est relativement facile, puisqu'il s'agit d'une simple boucle appelant la fonction précédente sur tous les sommets. Beaucoup de copies, cependant, échouent à correctement appliquer la numérotation des sommets.

**Question 10 [94% - 24%].** De nombreux candidat(e)s répètent la question ou tournent autour du pot. Il y a une confusion fréquente entre preuve par invariant de boucle et preuve par récurrence sur le nombre de sommets. Les preuves par récurrence sur le nombre de sommets qui omettent de considérer que le $(k+1)$-ième sommet considéré dans l'étape d'hérédité est celui qui est colorié en dernier sont en général fausses.

**Question 11 [71% - 19%].** Beaucoup de candidat(e)s perçoivent l'ordre qu'il faut considérer, à savoir par couleurs croissantes pour $L$, mais peu le définissent rigoureusement et encore moins le justifient correctement. Une définition de l'ordre par récurrence est acceptable lorsqu'elle est bien posée, mais ce n'est pas le cas dans de nombreuses copies.

**Question 12 [84% - 14%].** Les correcteurs apprécient que les candidats indiquent explicitement l'algorithme de tri qu'ils ont choisi de programmer, plutôt que de les laisser deviner à la lecture de leur code. Certains candidat(e)s indiquent un choix d'algorithme de tri, pour finalement en écrire un autre ! Des copies récitent un algorithme de tri en $O(n \log n)$ vu en cours, souvent correctement, mais c'est inutilement compliqué ici, la complexité étant déjà en $O(n^2)$ par ailleurs. À propos de complexité, calculer les degrés à la volée lors de chaque comparaison n'est pas une bonne façon de respecter la complexité demandée. Beaucoup de copies oublient de prendre en compte leur calcul des degrés dans la complexité totale.

## Partie IV

**Question 13 [80% - 65%].** Les correcteurs ont vu à plusieurs reprises des arguments totalement faux basés sur le degré des sommets dans un graphe $k$-colorié et souhaitent donc faire remarquer qu'une étoile à $n$ branches est 2-coloriable.

**Question 14 [54% - 17%].** Il y avait deux parties dans cette question. Le bon coloriage n'est pas difficile à justifier mais requiert néanmoins un peu de soin. Cette partie-là est souvent bâclée dans les copies. Le nombre de couleurs en $O(\sqrt{n})$, quant à lui, nécessite plusieurs arguments. En particulier, beaucoup de candidat(e)s se doutent que l'étape 2 de l'algorithme n'est effectuée qu'au plus $\sqrt{n}$ fois, mais peu le justifient correctement.

**Question 15 [82% - 58%].** Beaucoup de copies créent incorrectement une matrice avec `make_vect n (make_vect n false)` mais cela n'a pas été sanctionné. Cette question constitue avec les deux suivantes un groupe de questions de programmation particulièrement faciles et plutôt bien traitées.

**Question 16 [83% - 61%].** Il est amusant de noter que beaucoup de copies adoptent ici un style récursif pour la première fois. Certains candidat(e)s programment ici la fonction demandée à la question 17, en lisant probablement trop vite l'énoncé.

**Question 17 [77% - 58%].** Quelques copies tentent de réutiliser la fonction de la question 16, en faisant l'union des voisins non coloriés de tous les sommets du graphe. C'est évidemment incorrect.

**Question 18 [54% - 7%].** À la différence des trois précédentes, cette question de programmation est plutôt difficile. Par son ampleur, elle offre de multiples possibilités d'erreurs : oublier d'incrémenter les couleurs, mal reporter les couleurs du sous-graphe vers le graphe complet, considérer les sommets de fort degré en oubliant de vérifier qu'ils sont non coloriés, etc. Il faut de plus bien justifier de sa complexité polynomiale.

**Question 19 [36% - 8%].** Il y a souvent ici une confusion entre « appeler récursivement l'algorithme de Wigderson sur un graphe $k-1$-coloriable » et « colorier récursivement le sous-graphe avec $k-1$ couleurs ». L'algorithme de Wigderson sur un graphe 3-coloriable ne produit pas un 3-coloriage.