Composition d'Informatique (2 heures), Filière PC
(XEC)

Rapport de MM. Étienne LOZES et Dominique ROSSIN, correcteurs.

1. Bilan général

A titre de rappel, cette épreuve n'est corrigée que pour les candidats admissibles. Cette année le nombre total de candidats admissibles dans cette filière était de 529. La note moyenne est 9,2/20 avec un écart-type de 3,83. Les tableaux ci-dessous donnent la répartition détaillée des notes par série, ainsi que la synthèse calculée sur l'ensemble des copies corrigées. La note minimale est 0,5/20 et la note maximale 19,3/20. Deux copies n'ont pas été notées, les candidat(e)s ne s'étant pas présenté à l'épreuve.

	Série 1		Série 2		Série 3		Série 4		**Synthèse**	
$0 \leq N < 4$	9	6,6%	8	5,9%	9	6,7%	7	5,8%	**33**	**6,3%**
$4 \leq N < 8$	57	41,9%	51	37,5%	53	39,6%	44	36,4%	**205**	**38,9%**
$8 \leq N < 12$	45	33,1%	36	26,5%	40	29,9%	43	35,5%	**164**	**31,1%**
$12 \leq N < 16$	18	13,2%	32	23,5%	22	16,4%	18	14,9%	**90**	**17,1%**
$20 \leq N \leq 20$	7	5,1%	9	6,6%	10	7,5%	9	7,4%	**35**	**6,6%**
Total	136	100,0%	136	100,0%	134	100,0%	121	100,0%	**527**	**100,0%**

	Série 1	Série 2	Série 3	Série 4	**Synthèse**
Nombre de copies	136	136	134	121	**527**
Note minimale	0,7	1,5	2,1	0,5	**0,5**
Note maximale	17,7	17,9	19,3	19,3	**19,3**
Note moyenne	8,8	9,4	9,2	9,2	**9,2**
Ecart-type	3,6	4,0	3,9	3,9	**3,8**

Le langage de programmation choisi par les candidats est très largement dominé par Maple (92%). Les autres langages utilisés sont Python (4%), Mathematica (2%). Quelques copies sont en C++, Pascal, Caml, ou encore en français (2%). Nous invitons les enseignants de classes préparatoires et les candidat(e)s à suivre dès l'année prochaine les évolutions du programme d'informatique qui pourraient avoir pour conséquence de restreindre le choix du langage de programmation à cette épreuve.

2. Commentaires

Le sujet présentait une complexité "en cloche" : après deux questions faciles, la question 3 était d'un niveau de complexité relativement avancé, complexité qui culminait à la

question 7. Un bon nombre de candidats n'ont pas abordé les questions au-delà de la question 7, qui étaient pourtant significativement plus faciles à résoudre et demandaient essentiellement un esprit de synthèse.

Question 1 :

Cette question consiste à déterminer le nombre de 0 contigus à partir d'un indice donné du tableau. Elle a été bien traitée dans l'ensemble, malgré quelques débordements de tableaux à noter. Il est regrettable qu'un bon quart des candidats n'ait pas utilisé un déroutement de type break ou return à cette question (non pénalisé si la solution restait sous-linéaire).

Question 2 :

Cette question n'est finalement qu'une application de la question précédente, dans la mesure où il suffit d'appeler cette fonction précédente autant de fois que nécessaire pour parcourir l'ensemble du tableau, et calculer la plus grande des valeurs ainsi collectées.

A nouveau cette question a été traitée plutôt correctement, avec parmi les erreurs les plus courantes :

- oubli ou erreur d'écriture de l'incrémentation du compteur de boucle (boucle while),
- pas de protection contre le débordement de tableau,
- cas assez fréquent de boucle infinie lorsque la fonction précédente retourne 0.

Question 3 :

Cette question était sans doute la deuxième question la plus difficile du sujet, après la question 7. Il était demandé de calculer l'aire du plus grand rectangle composé de 0 ayant pour coin inférieur gauche la case de coordonnée (i, j), pour i et j donnés. Cette question a été correctement traitée par environ 10% des candidats.

Certains candidats ont simplement compté le nombre de 0 au-dessus et à droite de la case considérée, d'autres ont renvoyé l'aire du rectangle le plus étiré (horizontalement ou verticalement). Une solution efficace consistait à parcourir les lignes en remontant, et en retenant le nombre minimal de zéros contigus à droite rencontré sur les lignes précédentes, afin de pouvoir, à chaque étape, calculer l'aire d'un rectangle potentiellement maximale. Certains candidats ont agrémenté leur algorithme d'une figure ou de commentaires, très appréciables à cette question.

Question 4 :

Cette question amène le candidat à réfléchir à une approche naïve du problème de calcul d'un rectangle de 0 et d'aire maximale et d'en donner la complexité. L'idée consiste simplement à itérer la fonction précédente pour toutes les cases du tableau, et à évaluer la plus grande des aires ainsi collectées, ce qui donne un algorithme en temps $\mathcal{O}(n^4)$.

Question 5 :

Cette question consiste à calculer un tableau intermédiaire permettant de stocker le nombre de 0 contigus au-dessus de chaque case du tableau. Le point essentiel est ici d'écrire un code efficace, à savoir en temps $\mathcal{O}(n^2)$, il faut donc remarquer que chaque ligne de ce

tableau intermédiaire peut très facilement se déduire de la ligne précédente, sans calculs particuliers. Les solutions plus compliquées n'ont pas été pénalisées tant qu'elles étaient correctes et en temps $\mathcal{O}(n^2)$.

Question 6 :

A cette question, le candidat est invité à prendre un certain recul afin d'évaluer la complexité de son code à la question précédente. Les candidats ayant mené une bonne analyse de complexité vis-à-vis de leur solution ont obtenu tous les points même si cette solution n'était pas optimale.

Question 7 :

Cette question était sans doute la plus difficile du sujet. Elle avait pour prérequis de bien comprendre la définition de la grandeur $L[i]$ définie dans le sujet comme l'indice $j \leq i$ minimal tel que tous les histogrammes entre i et j soient au moins aussi grands que celui à l'indice i. Un algorithme était proposé pour calculer le tableau de tous les $L[i]$ par récurrence sur i. Il était demandé au candidat d'une part de prouver que l'algorithme était correct, et d'autre part qu'il s'exécutait en temps linéaire sur un histogramme de forme pyramidale.

La calcul de $L[i]$ se faisait en décrémentant j en partant de i, et en exploitant les $L[j]$ déjà calculés pour "sauter" certains indices au lieu de décrémenter j de 1 à chaque étape. Ce dernier point, le plus important à justifier, a été dans l'ensemble très mal compris, voire interprété comme une erreur d'énoncé.

L'histogramme pyramidal permettait de vérifier que le candidat avait compris d'une part la définition de $L[i]$, d'autre part le principe de l'algorithme. Beaucoup de candidats ont décompté, sur la seconde moitié de l'histogramme, un nombre $\mathcal{O}(i)$ d'étapes pour le calcul de $L[n+i]$, et ont malgré tout conclu à une complexité en temps linéaire pour le calcul de l'ensemble du tableau L. Un certain nombre de candidats ont par ailleurs prétendu que le calcul de $L[n+i]$ terminait sur l'indice n. Il était attendu que le candidat remarque que le calcul de $L[n+i]$ se faisait en deux étapes et terminait à l'indice $n-i$. Moins de 10% des candidats ont répondu aux deux parties de cette question de manière convaincante.

Question 8 :

Cette question consiste à justifier l'inclusion d'un rectangle de 0 caractérisé par ses $L[i]$, $R[i]$ et hauteur, dans l'histogramme introduit en début de partie III. Cette question ne présente vraiment aucune difficulté, et a été très majoritairement correctement traitée.

Question 9 :

La réponse à cette question repose sur l'hypothèse de maximalité de l'aire du rectangle considéré, hypothèse qui intervenait à plusieurs endroits dans la preuve attendue.

Question 10 :

Cette question demandait au candidat de faire une synthèse des questions traitant des histogrammes. Il s'agissait de calculer l'aire maximale d'un rectangle contenu dans

un histogramme donné. Les candidats ayant abordé la question l'ont bien traitée dans l'ensemble. Afin d'obtenir un algorithme en temps linéaire, il était crucial de comprendre que les tableaux L et R devaient être calculés une seule fois en appelant calculeL et calculeR en début de fonction. Petit détail : la longueur d'un intervalle situé entre les indices l et r est $r-l+1$ et non $r-l$. A noter également que certains candidats ne savent pas calculer l'aire d'un rectangle connaissant sa longueur et sa hauteur.

Question 11 :

Nouvelle question de synthèse relativement triviale. Il s'agissait de réutiliser les fonctions écrites aux questions 5 et 10. A noter ici que la fonction de la question 10 doit prendre en paramètre l'histogramme calculé à la question 5, et non le tableau tab2 d'origine.

Question 12 :

Cette dernière question consiste à prendre un peu de recul afin d'évaluer la complexité du code de la question 11, et de fait celle du code des questions 5, 7 et 10. Il s'agissait de conclure que l'algorithme donné s'exécutait en $\mathcal{O}(n^2)$ (en admettant certes que l'exemple donné à la question 7 jouait le rôle d'exemple dans le pire cas), et donc que cette solution était significativement plus efficace que la solution naïve étudiée à la question 4.