MINISTÈRE
DE L'ÉDUCATION
NATIONALE
ET DE LA JEUNESSE

Liberté
Égalité
Fraternité

Rapport du jury

Concours : agrégation externe

Section : Informatique

Session 2023

Rapport de jury présenté par :
Sylvie Boldo, directrice de recherche, présidente du jury.

Table des matières

Chapitre 1

Déroulement et statistiques

Ce document est le rapport de la deuxième édition de l'agrégation externe d'informatique, qui s'est déroulée en 2023. Il a plusieurs objectifs : c'est d'une part une analyse et un bilan de cette édition (statistiques sur les candidates et candidats, réussite aux épreuves, erreurs fréquentes, conseils…) et d'autre part un document à l'attention des futurs candidates, candidats, préparatrices et préparateurs (attentes du jury, écueils à éviter…).

1.1 Déroulement du concours

L'agrégation externe section informatique est régie par l'arrêté MENH2112666A du 17 mai 2021 [1] qui en définit en particulier le programme et les épreuves.

Le **programme des épreuves d'admissibilité** se compose des programmes d'enseignement de la spécialité « numérique et sciences informatiques » (NSI) du cycle terminal de la voie générale du lycée (première et terminale), de ceux des classes préparatoires scientifiques aux grandes écoles « mathématiques, physique, ingénierie et informatique » (MP2I) et « mathématiques, physique, informatique » (MPI), auxquels s'ajoute un programme complémentaire. Ce programme complémentaire a été élaboré par le jury. Il a pour but de lui permettre d'évaluer le recul des candidats sur les notions enseignées et a été conçu par « adhérence » des programmes sus-cités. Rappelons que, pour l'ensemble du programme, il est attendu des candidates et candidats un recul correspondant au niveau master.

Pour la session 2023, les **épreuves écrites** se sont déroulées du 6 au 8 mars, organisées par les académies :

- Composition d'informatique : 6 mars 2023 de 9 heures à 14 heures,
- Étude d'un problème informatique : 7 mars 2023 de 9 heures à 15 heures,
- Épreuve spécifique (selon l'option choisie : étude de cas informatique ou fondements de l'informatique) : 8 mars 2023 de 9 heures à 15 heures.

Les **épreuves orales** se sont déroulées du 17 au 22 juin 2023 au lycée Paul Valéry à Paris, organisées par le jury. Chaque admissible a été convoqué pour les trois épreuves orales sur trois jours consécutifs :

- Leçon d'informatique (4 heures de préparation, 1h d'épreuve),
- Travaux pratiques de programmation (5 heures de préparation, 1h d'épreuve),
- Modélisation (4 heures de préparation, 1h d'épreuve).

Pour plus de précisions, nous renvoyons les lectrices et lecteurs aux descriptifs de ces épreuves se trouvant dans l'arrêté MENH2112666A[1]. Nous signalons aussi le site web du jury `https://agreg-info.org`, régulièrement mis à jour. Il contient en particulier des sujets des années précédentes pour les épreuves écrites et orales ainsi qu'une FAQ (foire aux questions).

1. `https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000043648279`

1.2 Statistiques

En résumé et comme en 2022, ce concours a été attractif, avec un très grand nombre de candidates et candidats. De plus, le niveau final des agrégées et agrégés a été excellent. Le jury a pourvu sans hésiter tous les postes, avec de plus une liste complémentaire de 4 noms. Les agrégées et agrégés ont montré de très belles qualités, autant disciplinaires que pédagogiques. Le jury les a volontairement testés sur des domaines différents de l'informatique et des langages différents avec succès et a pu sélectionner des informaticiennes et informaticiens complets, avec une vision large de la discipline.

Voici tout d'abord le nombre de candidatures lors des différentes phases du concours. Pour 2023, les épreuves écrites furent communes avec l'agrégation d'informatique du Maroc. Les candidates et candidats des deux concours ont passé les épreuves écrites en même temps. Leurs copies ont ensuite été corrigées par les mêmes correcteurs et correctrices, anonymement et sans connaissance de la nationalité.

Inscrits (dont Maroc)	Présents (dont Maroc)	Inscrits (France)	Présents (France)
511	215	440	174

Admissibles	Admis liste principale	Liste complémentaire
48	22	4

Le chiffre des inscrits et des présents est en légère baisse par rapport à 2023, entre 20 et 25%. Malgré cette baisse, l'agrégation d'informatique reste parmi les sections très sélectives du concours de l'agrégation externe, avec environ 8,4 candidats pour un poste.

Dans la suite, nous fusionnerons sous le vocable "admis" les candidats admis et les candidats classés en liste complémentaire, la petite taille de cette dernière donnant peu de sens à la publication d'éléments statistiques spécifiques.

Pour ce qui concerne les épreuves écrites, la moyenne des candidats présents aux trois épreuves est de 4,68/20 et celle des candidats admissibles de 12,35/20, avec une **barre d'admissibilité à 8,76/20**.

Pour ce qui concerne les épreuves orales uniquement, la moyenne des candidats présents aux trois épreuves est de 11,06/20, et la moyenne des candidats admis en liste principale de 13,32/20 ; cette année encore, la grande qualité des candidats a conduit le jury à proposer une liste complémentaire de quatre noms. Sur l'ensemble des épreuves, la **barre d'admission est de 12,17/20 pour la liste principale et de 11,44/20 pour la liste complémentaire**. L'ensemble de ces chiffres montre une agrégation restant toujours très sélective.

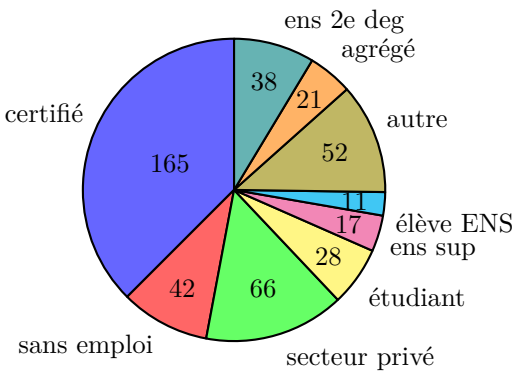
1.2.1 Statistiques par statut

Il est important de rappeler que ce tableau se base sur les statuts indiqués, de manière purement déclarative, par les candidates et candidats lors de l'inscription. Aucune vérification n'est opérée sur ce point, qui fait uniquement l'objet de la présente exploitation statistique.

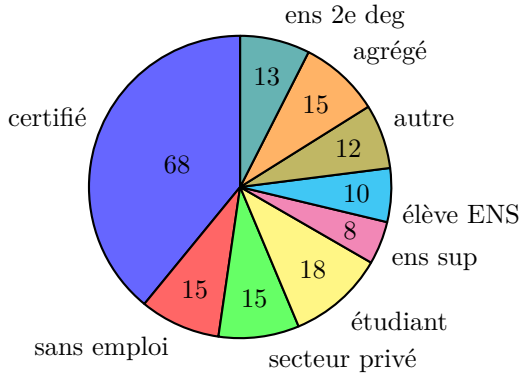
Les profils des candidats sont extrêmement divers, même si cette diversité tend à se réduire au cours du concours, pour favoriser progressivement les candidats bénéficiant de l'environnement d'une préparation. Le jury se réjouit d'observer un profil des admis un peu plus varié que l'an passé, même si les élèves des écoles normales supérieures constituent encore un bon tiers des lauréats, listes principale et complémentaire confondues.

Inscrits	Présents	Admissibles	Admis	Statut
165	68	12	1	Certifié
42	15	7	4	Sans emploi
38	9	1	1	Cadres secteur privé convention collective
21	15	4	0	Agrégé
14	10	0	0	Ens.stagiaire 2e deg. col/lyc
14	3	1	0	Salariés secteur tertiaire
12	3	1	0	Formateurs dans secteur privé
12	1	0	0	Professeur associé 2nd degré
11	10	10	10	Elève d'une ENS
11	5	2	1	Etud.hors inspe (sans prépa)
10	1	0	0	Contractuel 2nd degré
9	6	0	0	Etudiant en inspe en 2eme année
9	5	0	0	Maître contr.et agréé rem tit
9	2	0	0	Professions libérales
9	1	0	0	Personnel de la fonction publique
8	4	1	1	Enseignant du supérieur
7	7	5	5	Etud.hors inspe (prépa mo.univ)
7	3	2	2	Contractuel enseignant supérieur
6	0	0	0	Professeur des écoles
3	1	0	0	PLP
2	1	1	0	Militaire
2	1	0	0	Vacataire enseignant du sup.
2	0	0	0	Contractuel formation continue
2	0	0	0	Personnel enseignant non titulaire fonction publique
2	0	0	0	Personnel enseignant titulaire fonction publique
2	0	0	0	Salariés secteur industriel
1	1	1	1	Agent non titulaire fonction publique
1	1	0	0	Maître auxiliaire
1	1	0	0	Vacataire du 2nd degré
1	0	0	0	Assistant d'éducation
1	0	0	0	Emploi avenir prof.2nd d.publique
1	0	0	0	Enseignant non titulaire établissement scolaire étranger
1	0	0	0	Etudiant en inspe en 1ere année
1	0	0	0	Fonctionnaire stagiaire de la fonction publique
1	0	0	0	Maître contr.et agréé rem ma
1	0	0	0	PEPS
1	0	0	0	Personnel de direction

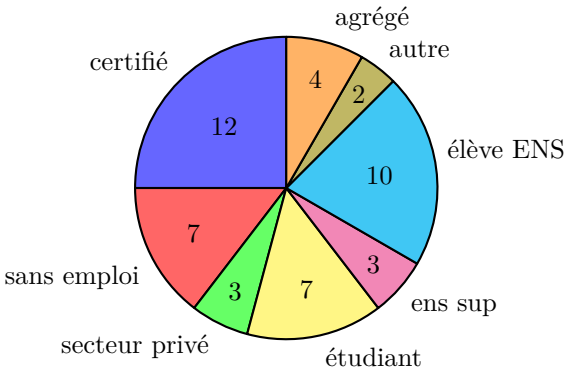
TABLE 1.1 – Répartition des candidats par statut.



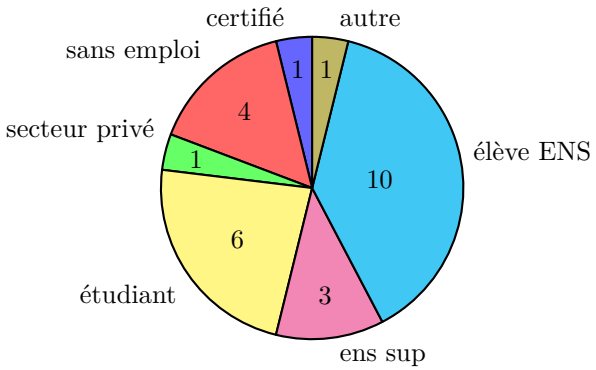
Inscrits par statut.



Présents par statut.



Admissibles par statut.

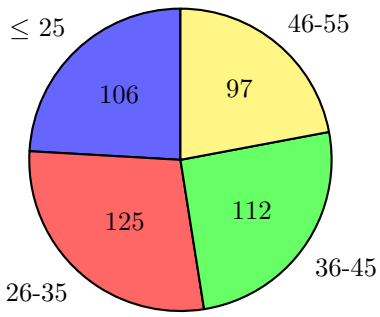


Admis par statut.

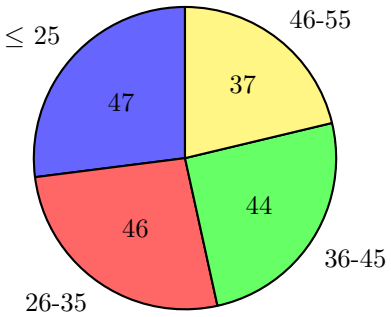
1.2.2 Statistiques par âge

Âge	Candidats	Présents	Admissibles	Admis
≤ 25	106	47	19	16
26-35	125	46	21	8
36-45	112	44	6	2
46-55	97	37	2	0
≥ 56	0	0	0	0

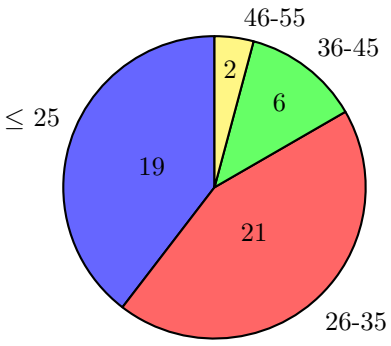
TABLE 1.2 – Répartition des candidats par âge.



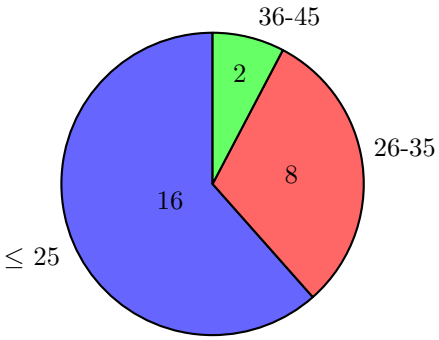
Inscrits par âge.



Présents par âge.



Admissibles par âge.



Admis par âge.

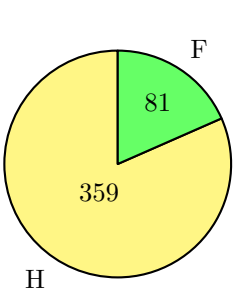
1.2.3 Statistiques par genre

Rappelons que le genre, également déclaratif, ne permet à l'inscription que les deux choix M. ou Mme.

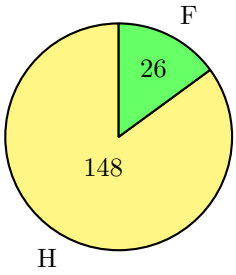
Genre	Candidats	Présents	Admissibles	Admis
M.	359	148	45	25
Mme	81	26	3	1

TABLE 1.3 – Répartition des candidats par genre.

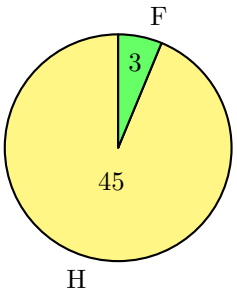
La représentation des femmes en sciences et en informatique en particulier reste très faible. Le taux de femmes inscrites est assez faible (moins de 19%), les femmes se présentent en moindre proportion, et elles ont plutôt moins bien réussi les épreuves écrites (sachant que les copies sont anonymes). Pour les épreuves orales, il est délicat de faire des analyses sur un échantillon aussi réduit, probablement corrélé avec d'autres facteurs statistiques mais ce point reste une préoccupation du jury, surtout au vu de la diminution du taux de femmes admises entre les sessions 2022 et 2023.



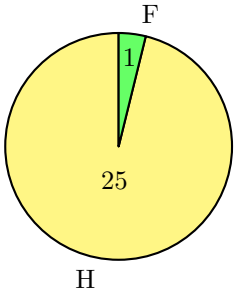
Inscrits par genre.



Présents par genre.



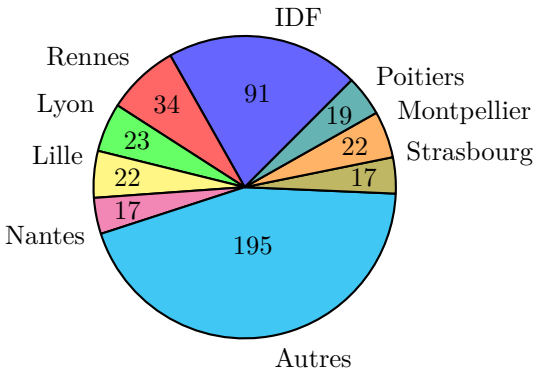
Admissibles par genre.



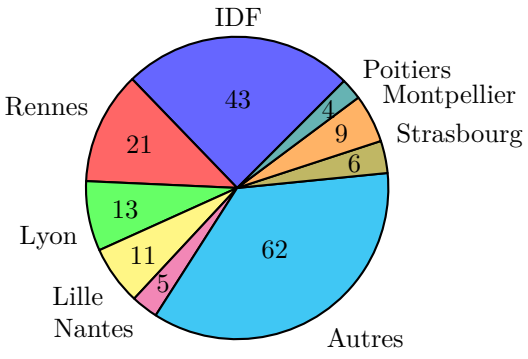
Admis par genre.

1.2.4 Statistiques par académie

Rappelons que les seules académies (à ce jour) à accueillir une préparation à l'agrégation externe d'informatique sont les académies de Créteil-Paris-Versailles, Lyon et Rennes, même si la première préparation peut se suivre partiellement à distance.



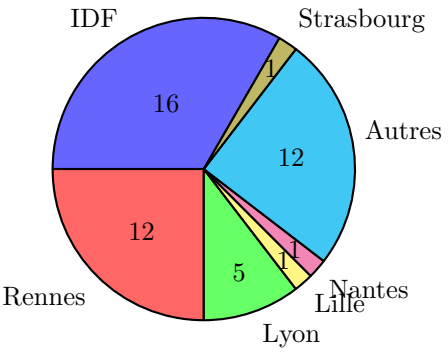
Inscrits par académie.



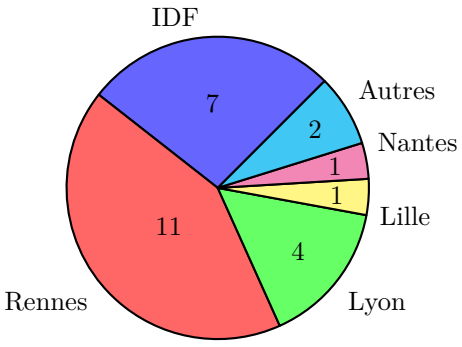
Présents par académie.

Académie	Inscrits	Présents	Admissibles	Admis
Créteil-Paris-Versailles	91	43	16	7
Rennes	34	21	12	11
Hors académie	31	0	0	0
Lyon	23	13	5	4
Lille	22	11	1	1
Montpellier	22	9	0	0
Poitiers	19	4	0	0
Strasbourg	17	6	1	0
Nantes	17	5	1	1
Toulouse	16	5	2	0
Nice	16	5	1	0
Grenoble	16	2	0	0
Normandie	15	6	0	0
La Réunion	14	5	0	0
Bordeaux	13	7	2	0
Aix-Marseille	13	6	2	0
Amiens	8	4	0	0
Orléans-Tours	7	4	2	1
Reims	7	2	0	0
Nancy-Metz	6	4	2	1
Nouvelle Calédonie	6	3	0	0
Guadeloupe	5	1	0	0
Limoges	4	1	1	0
Clermont-Ferrand	4	1	0	0
Mayotte	3	1	0	0
Besancon	3	0	0	0
Corse	2	2	0	0
Guyane	2	1	0	0
Dijon	2	0	0	0
Martinique	1	1	0	0
Polynésie francaise	1	1	0	0

TABLE 1.4 – Répartition des candidats par académie.



Admissibles par académie.



Admis par académie.

1.3 Remerciements

Le jury du concours 2023 souhaite remercier les personnels de la DGRH pour tout leur appui logistique. En particulier, un sincère merci à la gestionnaire de notre concours pour sa disponibilité sans faille et son aide précieuse.

Un très chaleureux merci à la direction et l'ensemble des personnels du lycée Paul Valéry qui nous ont accueillis avec une grande gentillesse, en étant constamment à l'écoute de nos demandes malgré les travaux se déroulant dans le lycée, et nous ont permis de tenir cette deuxième session du concours dans d'excellentes conditions.

La présidence du jury remercie également très chaleureusement les membres du jury pour leur travail dans la création de sujets pour ces très nombreuses épreuves, la qualité de la correction et de l'interrogation, leur ouverture d'esprit et leur implication.

Chapitre 2

Épreuves écrites

Pour alléger le texte, ce chapitre est uniquement écrit au féminin [1], mais cela doit être pris dans le sens du neutre : l'ensemble de ce texte, sans exception, concerne uniformément candidates et candidats.

L'architecture générale de l'écrit de l'agrégation externe d'informatique est définie par l'arrêté du 17 mai 2021. Dans les limites de la maquette définie par cet arrêté, le jury souhaite affirmer sa volonté d'utiliser l'ensemble des épreuves à sa disposition pour recruter des *informaticiennes complètes*, capables à la fois de réflexes sains dans un ensemble de domaines assez larges de l'informatique (épreuve 1), et dotées d'une solide maîtrise de la programmation et de l'algorithmique (épreuve 2). L'épreuve 3 est l'occasion de démontrer des connaissances approfondies dans un domaine au choix des candidates, soit plutôt l'ingénierie logicielle, soit plutôt les fondements de l'informatique.

Certaines remarques s'appliquent à toutes les épreuves. En particulier, on attend d'une candidate aux fonctions de professeure qu'elle sache rédiger une copie. En conséquence, un passage rayé ou hachuré ne sera jamais lu. De plus, quand plusieurs versions sont proposées par la candidate, la correctrice ne prend pas la meilleure. C'est à la candidate de choisir ce qu'elle présente à l'évaluation du jury, pour le meilleur et pour le pire.

Une préoccupation du jury est de valoriser les candidates précises sur les parties élémentaires du sujet plus que celles qui bâcleraient le début pour avancer plus vite vers les parties plus ambitieuses. Le jury recherche des candidates solides et précises sur les bases de la discipline, afin de pouvoir transmettre ces dernières.

Il est attendu, pour l'écrit comme pour l'oral, que les candidates maîtrisent le vocabulaire des mathématiques nécessaire à l'étude des éléments au programme de l'agrégation d'informatique, dont par exemple les algorithmes probabilistes et l'analyse de complexité.

Le jury tient compte dans la notation des épreuves de la maîtrise écrite et orale de la langue française (vocabulaire, grammaire, conjugaison, ponctuation, orthographe) [2]. Le jury attend donc une certaine tenue dans l'écriture, la présentation et l'orthographe. Lesdites attentes sont modérées, mais réelles. Le jury fait la part des choses entre ce qui est explicable par le contexte "concours" et ce qui risque d'handicaper une future agrégée dans sa pratique professionnelle et la transmission des savoirs. Les travers relevant de la seconde catégorie sont pris en compte, aux côtés de la maîtrise disciplinaire, dans l'évaluation d'une copie.

Le jury a apprécié de lire des réponses construites, c'est-à-dire des phrases. Rappelons qu'une phrase doit toujours commencer par une majuscule et se terminer par un point, même quand elle est très courte et constitue à elle seule la réponse à une question.

1. Le chapitre 3 est écrit au masculin.
2. `https://www.legifrance.gouv.fr/loda/id/LEGIARTI000046287651/2022-09-01/#LEGIARTI000046287651`

2.1 Épreuve 1

Cette épreuve était composée de trois parties, balayant largement plusieurs domaines de l'informatique : les bases de données, les graphes et l'architecture des ordinateurs.

Le jury a été déçu par les réponses à cette épreuve, car de nombreuses candidates ont peu abordé certaines parties. Hormis la sous-partie sur les requêtes SQL de la première partie, les correctrices ont estimé que trop de questions ont été mal ou pas traitées dans cette épreuve.

Pour cette épreuve en particulier, composée de plusieurs parties indépendantes, les correctrices souhaitent que les candidates notent bien le numéro complet de la question traitée (1.9 par exemple) et qu'elles ne mélangent pas les réponses aux différentes parties.

Données statistiques Pour chaque épreuve, et en plus des données de base, un tableau statistique est fourni. Il contient à la fois les quartiles et la proportion des candidates ayant une note supérieure à 5, 10 et 15.

— 185 présents
— Meilleure note : 18,09/20
— Moyenne : 6,51 ; écart-type : 3,89.

$\geq 3,99$	75%
$\geq 5,00$	57,3%
$\geq 5,47$	50%
$\geq 8,57$	25%
$\geq 10,00$	19,5%
$\geq 15,00$	3,2%

2.1.1 Partie I - Provenance en bases de données

Cette partie a pour but de tester les connaissances en bases de données, notamment des requêtes SQL. Le sujet se poursuit par un point de vue plus logique permettant de parler de la provenance des résultats d'une requête.

Remarques générales

Cette partie a été beaucoup traitée, les requêtes SQL étant souvent correctes. Le jury regrette la syntaxe SQL parfois approximative et le fait que vraiment peu de copies traitent la partie sur la provenance.

Plus précisément en ce qui concerne les requêtes SQL, les candidates doivent veiller à l'indentation pour aider à la lisibilité et à ne pas faire de jointure inutile. Il est aussi conseillé de bien lire les questions de façon à faire la requête demandée. En particulier, la question 1.4 a été mal comprise par beaucoup de candidates. Enfin, certaines candidates ont utilisé des doubles négations dans les requêtes, rendant le code peu compréhensible et souvent factuellement faux. L'algèbre relationnelle étant au programme complémentaire, le jury a accepté une certaine souplesse sur la syntaxe, mais n'a pas accepté `max` comme opérateur.

Remarques par question

Question 1.1 La clé primaire de Navigable est souvent oubliée ; le fait que les deux champs soient soulignés ne veut pas dire que chaque champ est une clé primaire. Beaucoup de candidates ont utilisé `CHAR` ou des `varchar(k)` avec k très faible pour stocker les chaînes de caractères alors que le jury attendait plutôt des `varchar`. L'ordre de création des tables n'est pas anodin : pour définir les contraintes d'intégrité de la table Navigation, il faut avoir défini auparavant les tables liées.

Question 1.2 On trouve souvent la jointure de la table Employe, qui n'apporte rien à la requête.

Question 1.4 De nombreuses copies écrivent une requête qui renvoie les identifiants des bateaux pouvant être pilotés par *au moins* un marin de plus de 40 ans, au lieu de *tous* les marins de plus de 40 ans.

Question 1.5 De nombreuses copies écrivent une requête qui renvoie les identifiants des marins qui peuvent piloter un bateau non catamaran de plus de 3000 nœuds d'autonomie.

Question 1.6 Des candidates ont listé les employés de plus d'un certain âge (arbitraire) ou un nombre (arbitraire) d'employés ayant les âges maximaux, alors qu'on demandait les employés ayant l'âge maximal parmi les âges de tous les employés.

Question 1.9 Cette question a été assez mal réussie. Le calcul de la moyenne en SQL se fait avec `AVG`, et non pas `MEAN` ou `MOY`. Comme dit dans l'énoncé, les marins doivent savoir piloter au moins un bateau. Il fallait également faire attention à la multiplicité introduite par le join avec la table Navigable pour calculer une moyenne correcte.

Question 1.10 Il fallait faire attention ici à ne renvoyer qu'un seul enregistrement : un `SELECT FROM Q` était insuffisant, car cette requête renvoie autant de 0-uplets que le nombre d'enregistrements de `Q`.

Question 1.11 Ici aussi, il fallait bien lire la question, car il était demandé la provenance des requêtes 1.2 et 1.9 seulement ; il est dommage de perdre du temps à donner l'ensemble des provenances des requêtes 1.3 à 1.8. Le jury a noté que certaines candidates ne savent pas calculer la moyenne de deux entiers. Enfin, une description simple en français suffisait ici, plutôt qu'une tentative d'énumération des 128 sous-ensembles.

Question 1.12 Pour la deuxième partie de la question, il fallait l'utilisation d'un ou exclusif (exprimable avec uniquement $\vee$, $\wedge$ et $\neg$), pas d'un $\vee$ uniquement. Le jury estime également que des écritures comme $\{\{e_1, e_2\}\} = x_1 \wedge x_2$ sont malvenues.

Question 1.13 Le jury attendait ici une justification rigoureuse de la taille de l'ensemble, avec une preuve par récurrence ou par induction par exemple ; se contenter de dire « il n'y a que deux possibilités par élément » est insuffisant.

2.1.2 Partie II - Ponts d'un graphe

L'épreuve portait sur l'implémentation, en langage OCaml, d'une structure de données permettant de déterminer efficacement les blocs d'un graphe non orienté (c'est-à-dire les composantes connexes restant après suppression des *ponts*, définis comme les arêtes dont la suppression fait croître le nombre de composantes connexes) dans le cas où le graphe est donné *en ligne*, autrement dit de façon incrémentale en ajoutant les arêtes une à une. Cette approche se base sur une variante de la structure classique de *unir et trouver*, présente au programme de MPI.

Remarques générales

Globalement, le jury regrette que cette partie de la première épreuve ait été si peu et si mal traitée, tant au niveau de la maîtrise du langage OCaml que du sujet lui-même, à savoir l'étude d'une structure de données.

Ainsi, il est manifeste que les bases du langage OCaml ne sont pas du tout acquises pour une proportion non négligeable de candidates, alors qu'il s'agit d'un des langages explicitement au programme. Dans certaines copies, on a pu lire une sorte de pseudo-code ressemblant vaguement à du Python, voire même du code Python de manière totalement assumée. Cela n'est pas acceptable. Au début de l'énoncé, il était rappelé un ensemble de notations concernant en particulier la manipulation

de tableaux. Il est donc regrettable de trouver dans des copies des notations incorrectes (par exemple, écrire `t[i]` au lieu de `t.(i)` comme rappelé au début du sujet) et une mauvaise utilisation des fonctions de base pour manipuler les tableaux.

Le jury a aussi eu l'impression que de trop nombreuses candidates pensent, de façon assez caricaturale, que programmer en OCaml signifie faire exclusivement de la programmation récursive, et que toutes les fonctions doivent commencer par un `match ... with`. Cette situation, très naturelle lorsque l'on manipule des paramètres d'un type somme (on effectue alors une disjonction de cas selon le constructeur de l'argument), ne correspondait pas à un grand nombre de questions de l'énoncé. D'ailleurs, un usage excessif et mal maîtrisé du filtrage peut être vu comme maladroit, voire créer des complications inutiles. Ainsi, écrire

```
match ... with
| true -> ...
| false -> ...
```

est correct bien que moins naturel que `if ... then ... else ...` mais la construction

```
match x with y -> ...
```

à la place de `if x = y then ...` n'est tout simplement pas correcte si `y` est une expression constituée d'un unique identifiant.

Notons enfin que certaines questions attendaient de façon naturelle une solution itérative et que si l'utilisation de fonctions auxiliaires est admise voire recommandée, il est **indispensable** d'en expliciter clairement le rôle.

Remarques par question

Question 2.4 (`find`) Quelques confusions ont été commises, du fait que contrairement à une implémentation classique de l'algorithme *unir et trouver*, ce n'est pas la racine d'un arbre mais le représentant d'une classe que l'on devait renvoyer ici.

Il était possible de répondre avec une fonction utilisant une boucle `while` pour rejoindre le représentant d'une classe, mais une fonction récursive donnait une solution bien plus courte et élégante.

Question 2.5 (`blocs`) Pour cette question, certaines candidates ont implémenté un parcours de graphe avec marquage de sommets. Bien que correct, c'était inutile ici puisque ce sont des arbres que l'on parcourait.

Question 2.6 (`chaine_racine`) Plusieurs copies n'utilisent pas le parent d'un sommet, rendant la fonction incorrecte.

Question 2.8 (`retourner_chaine`) Pour cette question relativement simple, il fallait prendre soin de ne pas oublier de traiter le cas de base, c'est-à-dire une liste contenant au plus un élément.

2.1.3 Partie III - Architecture des ordinateurs

Cette partie visait à évaluer les candidates sur l'architecture des ordinateurs, avec une première section sur l'algèbre de Boole, une deuxième sur les bascules et une troisième sur un automate de Moore.

Remarques générales

La première section a été plutôt bien réussie et abordée par une majorité de candidates.

La deuxième section concernait le fonctionnement des bascules RS et D et a été plus décevante. Les correctrices ont pu constater qu'une part importante des candidates n'a pas compris de manière précise le fonctionnement d'une bascule D. Beaucoup de candidates se sont échouées sur la question 5.

La troisième section, qui était probablement la plus facile de la partie, a été complètement ratée. Il s'agissait uniquement de donner les tables d'un automate de Moore donné, pour ensuite le synthétiser à l'aide de portes NOR. La notion d'automate de Moore, au programme de l'agrégation, ne semble pas assimilée.

Remarques par question

Question 3.1 Pour la question b, un certain nombre de candidates se sont perdues dans des expressions compliquées et illisibles. Certaines n'ont pas représenté le circuit comme demandé.

Pour la question c, il ne suffisait pas de développer et de constater que les expressions logiques sont différentes. Il fallait exhiber un contre-exemple.

Question 3.2 Pour la question a, une majorité de copies est passé par les tables de vérité pour démontrer l'égalité, ce qui est correct mais étonnant compte tenu de la simplicité de la preuve directe utilisant les propriétés de l'algèbre de Boole.

Pour la question b, le jury a constaté beaucoup d'erreurs ; notamment, beaucoup de candidates ont exprimé le résultat en utilisant l'opérateur de négation sur une partie de l'expression obtenue. Comme indiqué dans l'énoncé, la négation ne pouvait être considérée que pour les paramètres.

Question 3.3 Les questions a, b et c ont été plutôt bien comprises, mais les justifications sont souvent inexistantes. La question d consistait uniquement à reprendre dans une table les résultats des 3 sous-questions précédentes. Cependant, beaucoup de candidates ont construit de zéro une table de vérité complète en détaillant les valeurs de sortie des portes en fonction des valeurs d'entrées, et sans tenir compte de la stabilisation des valeurs. Par exemple, pour $(R, S, Q_1, Q_2) = (0, 1, 0, 1)$, la valeur de sortie des portes est $(Q_1, Q_2) = (0, 0)$. Mais, après stabilisation, on obtient $(Q_1, Q_2) = (1, 0)$ qui est ainsi la valeur attendue.

La question e a souvent été mal réussie, à cause d'une table de vérité fausse. La question f est une conséquence directe des questions a et b, et a été plutôt bien réussie.

Question 3.5 et 3.6 Beaucoup de candidates se sont arrêtées à ces questions. D'une manière générale, peu de candidates sont capables d'analyser les sorties du système en fonction des entrées alors que les systèmes obtenus dans la question 4 sont justes. Cela démontre que le fonctionnement des bascules RS n'est en fait pas compris.

Question 3.7 Une partie importante des candidates ne sait pas ce qu'est un automate de Moore (et la différence avec les automates d'états finis). La sortie demandée en question a et l'explication de la question b sont donc souvent incorrectes.

Question 3.8 Les peu nombreuses candidates qui ont traité cette question sont encore moins nombreuses à savoir ce qu'est un tableau de Karnaugh et son utilité pour simplifier les expressions booléennes.

2.2 Épreuve 2

— 179 présents
— Meilleure note : 20/20
— Moyenne : 5,80 ; écart-type : 4,76.

$\geq 2,00$	75%
$\geq 5,00$	45,3%
$\geq 4,35$	50%
$\geq 9,12$	25%
$\geq 10,00$	22,3%
$\geq 15,00$	4,5%

L'objet de cette épreuve est d'étudier la décidabilité du problème d'équivalence entre deux expressions rationnelles. Elle est composée de trois parties : la partie I contient des préliminaires algorithmiques, la partie II étudie le passage d'une expression rationnelle à un automate non déterministe équivalent, et enfin la partie III porte sur l'équivalence de deux automates déterministes. Le but principal de l'épreuve était, en plus de vérifier les connaissances des candidates sur ces parties du programme de l'agrégation, de tester leurs capacités à raisonner sur des programmes informatiques.

Cette épreuve part de concepts et de résultats explicitement au programme de l'agrégation, pour aller jusqu'à des résultats classiques et des extensions tirées d'articles de recherche. Le parti pris concernant les résultats non explicitement au programme a été de les introduire et les analyser à travers le prisme de la programmation. Le sujet comportait régulièrement des questions de déroulement d'algorithmes sur des exemples simples qui devaient permettre aux candidates de se familiariser progressivement avec les notions introduites.

La partie I permettait aux candidates de prouver leur maîtrise de base de la programmation Python et de l'analyse de la complexité temporelle de fonctions simples, à travers l'étude de la complexité amortie pour les listes Python et le tri lexicographique (notions utilisées pour la suite de l'énoncé).

La partie II commençait par une section visant à transformer une expression rationnelle en arbre. Il s'agissait ensuite d'appliquer l'algorithme de Berry-Sethi pour obtenir l'automate de Glushkov, et de prouver la correction totale du code fourni et sa complexité temporelle. Enfin l'énoncé s'inspirait de l'article *"Passage d'une expression rationnelle à un automate fini non-déterministe"* de D. Ziadi, J.-L. Ponty et J.-M. Champarnaud (*Bull. Belg. Math. Soc. 4 (1997), pp.177–203*) pour modifier la fonction de l'énoncé dans le but d'améliorer sa complexité. Cette partie permettait de tester les capacités des candidates à faire des raisonnements par récurrence et par induction.

La partie III portait sur la minimisation d'automates déterministes, dans le but de tester l'équivalence de deux automates déterministes, en étudiant l'équivalence de Nérode, l'algorithme de minimisation de Moore, et en terminant par un algorithme de type "unir & trouver". Cette partie reposait sur les capacités des candidates à analyser des fonctions pour prouver des résultats sur les objets manipulés par ces fonctions. L'algorithme de type "unir & trouver" est inspiré de l'article *"A linear algorithm for testing equivalence of finite automata."* de J. E. Hopcroft and R. M. Karp (*Technical Report 114, Cornell University, December 1971*). Cette partie permettait aux candidates de montrer leur capacité à faire le lien entre les algorithmes et structures de données au programme et la théorie des automates.

Remarques générales

Présentation Comme l'an passé, le jury invite les candidates à soigner la qualité scripturale, grammaticale et rédactionnelle de leur copie. Il y est doublement sensible : une copie convenablement écrite est plus facile à évaluer et, par ailleurs, il parait inconcevable de recruter des enseignantes ne montrant pas une maîtrise raisonnable de la langue française. Quel mauvais exemple pour ses élèves ferait une professeure commençant ses phrases par des minuscules, ne sachant pas conjuguer

un verbe au présent de l'indicatif ou négligeant d'accorder les adjectifs ! Il convient également d'utiliser un vocabulaire approprié : on ne dit pas d'une propriété qu'elle est « forcément » vraie, et il n'est pas davantage adéquat d'écrire « On est sur une complexité $O(n)$. ». La dislocation du sujet (« La seule opération qui ne s'effectue pas en temps constant, c'est l'ajout à la fin. » au lieu de « La seule opération qui ne s'effectue pas en temps constant est l'ajout à la fin. ») doit demeurer une figure de style marquant une emphase, qui est ici inutile et malvenue ; elle ne doit pas être banalisée et encore moins rendue systématique.

De la même façon, il est bon de respecter la syntaxe et la graphie lorsqu'on écrit du code informatique : la casse n'est pas anodine en Python. Le jury n'apprécie pas le code rempli de ratures ou le code sur deux colonnes (ce qui donne une indentation particulièrement peu lisible en Python).

Il faut éviter les renvois d'une page à une autre avec des flèches. D'une part cela pose des problèmes de propreté de la présentation, d'autre part les copies étant scannées et corrigées sur écran, il n'est pas toujours évident pour les correctrices de suivre les dites flèches au travers de plusieurs feuilles. Enfin, il n'est pas utile de recopier la question : en rappeler le numéro suffit.

Programmation Le jury est surpris que certaines candidates ne sachent pas programmer en Python, langage pourtant au programme de l'agrégation.

Dans les questions de programmation, le jury n'attend pas, sauf mention contraire, que le programme vérifie que les paramètres respectent les préconditions. Il n'est ainsi pas utile de mettre un `assert` en Q6 ou de lever des exceptions en Q15. Il n'est pas non plus utile de traduire du code en français.

Formalisme et raisonnement mathématiques Comme l'an passé, le jury constate une très grande variance dans la qualité des raisonnements, qu'il s'agisse de leur présentation ou du fond. Certaines copies pourraient servir de modèle de corrigé. D'autres semblent rédigées par des candidates qui n'ont pas compris qu'elles passent un concours scientifique. Rappelons notamment qu'un exemple ne peut pas servir à prouver la correction d'une fonction.

Les symboles mathématiques doivent être employés à bon escient et ne se substituent pas à un verbe : on ne peut pas utiliser $\forall$ dans une phrase comme si c'était une abréviation ; on n'écrit pas non plus « Si I_n, le nombre d'écritures, $= n + \sum_{i=0}^{k-1} 2^i$, alors… », mais plutôt « Si I_n, le nombre d'écritures, est égal à $n + \sum_{i=0}^{k-1} 2^i$, alors… » ou « Si I_n, le nombre d'écritures, est tel que $I_n = n + \sum_{i=0}^{k-1} 2^i$, alors… ». Enfin, il convient d'éviter de commencer une phrase par une formule, un mot-clé du langage de programmation ou un nombre.

Plus profondément, le jury déplore que dans certaines copies, le symbole d'implication $\implies$ semble être employé en lieu et place d'une déduction. On ne peut pas, en tant que future enseignante, se permettre cet abus d'écriture qui peut aussi constituer une erreur de logique grave : lorsqu'on écrit en Q3 « $\mathit{longueur} = 0$ et $\mathit{capacite} = 1 \implies \mathit{longueur} < \mathit{capacite} \implies I_1 = 1$ », on n'affirme pas que $\mathit{longueur} = 0$, ni que $\mathit{capacite} = 1$, ni que $\mathit{longueur} < \mathit{capacite}$, ni que $I_1 = 1$ (ce qu'on devait pourtant démontrer), mais seulement que les deux premières de ces propriétés entraînent la troisième (ce qui est vrai, mais peu intéressant en soi), et que la troisième entraîne la quatrième (ce qui est faux en général). Il convient d'employer ici le mot « donc » ou un synonyme pour affirmer successivement chacune de ces propriétés en les déduisant les unes des autres. Pour la même raison, la conclusion d'une réponse ne doit pas être introduite par une flèche et surtout pas par la flèche d'implication. L'absence de maîtrise de ce point, outre qu'elle fait désordre pour toute professeure scientifique, révèle un manque de recul sur les chapitres de logique du programme de l'agrégation.

La manipulation de la notation $O(\cdot)$ doit être raisonnablement rigoureuse. Même si, la plupart du temps, la notation dissimule le paramètre, écrire $O(\cdot)$ décrit un comportement d'une suite numérique lorsque le paramètre tend vers ∞ et non pas une propriété valable « pour de grandes valeurs du paramètre ». Il s'agit d'une propriété globale de la suite et non de ses termes. Ainsi, cela n'a pas de sens d'écrire, par exemple, que $I_n = O(1)$ pour $n = 1$ ou pour un n fixé. Une définition précise des notations O, Ω, Θ doit être maîtrisée par les candidates à l'agrégation. Il est par ailleurs important que ces trois symboles soient bien différenciés les uns des autres dans l'écriture.

Enfin, il convient d'être rigoureux lorsqu'on veut prouver une majoration de complexité par induction structurelle. On ne peut pas supposer que, pour une sous-expressions E, la complexité est en $O(|E|)$. Cela n'a pas de sens. Il faut passer par une majoration avec des constantes explicites.

Remarques par question

Question 4. La somme des puissances de 2 n'est pas négligeable devant n. On attend de la part d'une future enseignante qu'elle fasse le lien entre la quantité I_n introduite à la question précédente et la complexité demandée ici. Elle ne peut se contenter de donner un ordre de grandeur de I_n sans aucune explication.

Question 6. Une solution récursive avec un appel `plus_petit_k_uple(l1[1:], l2[1:])` n'est pas de complexité linéaire et ne répond donc pas à la question posée.

Dans l'usage scientifique français, au contraire de l'usage anglo-saxon, les relations d'ordre s'entendent au sens large. Il convient donc, en l'absence du mot « strictement », que la fonction `plus_petit_k_uple` renvoie `True` en cas d'égalité. Si on choisit d'écrire une boucle `while` qui parcourt les deux tableaux tant que les termes sont égaux, il faut exploiter convenablement l'évaluation paresseuse des booléens : une boucle commençant par `while t1[i]==t2[i] and i<len(t1)` provoque une erreur d'accès lorsque `t1` et `t2` sont égaux.

Il est bien entendu que la fonction `plus_petit_k_uple` ne doit pas modifier les tableaux passés en paramètre.

Question 9. Certaines copies procèdent à la concaténation finale des listes concernant chaque valeur de la clé en effectuant `for i in range(c): L = L + T[i]`. Outre que l'opération `+` ne fait pas partie des primitives proposées pour cette partie du problème, procéder ainsi est rédhibitoirement inefficace : chaque tour de cette boucle a un coût linéaire en la somme des longueurs de `L` et `T[i]`, et non linéaire en la longueur de `T[i]` comme attendu. Contrairement à ce que prétendent certaines candidates, il n'y a aucune contradiction dans la consigne « Écrire l'algorithme TriParCle(L, j, C) en Python » : en effet, rien n'interdit d'utiliser un fragment de Python comme langage de description d'algorithme ; son caractère concis et son modèle mémoire relativement abstrait le rendent d'ailleurs plutôt adapté à cette tâche dans bien des situations.

Question 13. Il s'agissait de répondre à cette question pour une expression E en général et non pas pour l'exemple de la fonction précédente.

Question 14. Plusieurs copies ne donnent que l'arbre, ou seulement la version Python : il convient de lire les questions avec attention. Le mot *syntaxique* s'écrit sans *h*.

Question 15. Plusieurs copies proposent un parenthésage non conforme aux spécifications de l'énoncé, lesquelles sont pourtant illustrées par un exemple. Il est recommandé de vérifier que le code que l'on propose est correct sur les exemples de l'énoncé.

Question 16. Un certain nombre de candidates recherche la présence de ε dans l'arbre syntaxique, soit directement par un traitement récursif, soit en cherchant `(_)` dans la version textuelle produite à la question précédente. Or c'est sur le langage dénoté par l'expression régulière que portait la question. Le jury considère cette erreur, qui relève de la confusion entre syntaxe et sémantique, comme extrêmement grave. Cette question a ainsi été utile pour classer les candidates.

Question 17. Les candidates doivent faire attention à ce que la fonction auxiliaire renvoie des valeurs de type homogène dans chaque cas traité.

Certaines copies présentent une linéarisation avec une numérotation par lettre plutôt que globale, produisant $a_1b_1a_2$ au lieu de $a_1b_2a_3$. On peut en effet procéder ainsi, mais ce n'est pas ce que l'énoncé décrivait et imposait : il convient de s'y conformer.

Cette question pouvait être traitée de plusieurs manières. Toutes impliquent l'écriture d'une fonction récursive auxiliaire. Il était possible de travailler de façon très fonctionnelle, en donnant à cette fonction un paramètre supplémentaire repérant le prochain numéro à utiliser et en lui faisant renvoyer le prochain numéro disponible. L'écriture de la fonction auxiliaire au sein de la fonction principale et l'utilisation d'une variable introduite avec `nonlocal` est également tout à fait valable. L'utilisation d'une variable globale a aussi été tolérée, à condition qu'elle ait été convenablement introduite par le mot-clé `global` ici nécessaire en raison des affectations que cette variable subit, et surtout que la remise à zéro de cette variable ait été convenablement gérée (faute de quoi des linéarisations successives d'expressions régulières sans rapport recevraient des numéros sans cesse croissants). Notons que dans ces deux dernières façons de faire, il n'y a pas de problème en évaluant `[expr[0], linearise(expr[1]), linearise(expr[2])]` car *en Python*, il est garanti que l'évaluation se fait de gauche à droite — le signaler était bienvenu. Enfin, procéder en définissant une classe avec une méthode `__call__` était également possible même si ce n'est pas vraiment dans l'esprit du programme ou tout au moins celui du sujet. Pour la même raison, il convient d'éviter, même quand elles sont techniquement correctes, des constructions bancales telles que `e[0] in ".+"` ou `if e[0]`, pour leur préférer, respectivement, `e[0] in [".", "+"]` (sémantiquement plus adéquat) et `if e[0] != ""` (conforme à l'esprit du programme).

Question 19. Il s'agissait de répondre à cette question pour une expression E en général, pas pour l'exemple de la fonction précédente. Peu de réponses satisfaisantes à cette question il est vrai pénible. Plusieurs candidates montrent que, ε mis à part, l'automate $\mathcal{A}_\ell$ reconnaît le langage des mots dont les premières lettres sont dans $\mathrm{First}(E_\ell)$, les dernières lettres dans $\mathrm{Last}(E_\ell)$ et les facteurs de taille 2 dans $\mathrm{Follow}(E_\ell)$ — ce qui est assez immédiat, puisque la définition de l'automate traduit ces règles. Cela ne suffit pas : il faut encore prouver que ce langage est $\mathcal{L}(E_\ell)$, ce qui est le cas car E_ℓ est une expression régulière linéaire mais n'a rien d'évident (c'est faux en général pour les expressions régulières non linéaires) et nécessite une démonstration par induction sur les expressions régulières linéaires.

Question 28. De très nombreuses candidates montrent que la complexité est $O(n)$ mais négligent de montrer qu'elle est aussi un $\Omega(n)$. Le jury attendait pour ce point l'exhibition d'une famille d'expressions régulières permettant de conclure.

Question 34. Certaines candidates signalent comme erreur dans le sujet le fait que sur l'état 2 arrivent des transitions étiquetées par des lettres différentes. De fait l'exemple n'est pas un automate local, mais le sujet ne l'affirmait pas.

Question 37. Très peu de copies ont traité correctement le cas où il y a une unique classe d'équivalence dans la partition 0.

2.3 Épreuve 3

Cette épreuve permet à la candidate de choisir (à l'inscription) entre étude de cas informatique ou fondements de l'informatique.

Étude de cas informatique

— 107 présents
— Meilleure note : 17/20
— Moyenne : 7,30 ; écart-type : 3,60.

$\geq 4,97$	75%
$\geq 5,00$	73,8%
$\geq 7,11$	50%
$\geq 9,75$	25%
$\geq 10,00$	23,4%
$\geq 15,00$	3,7%

Fondements de l'informatique

— 68 présents
— Meilleure note : 20/20
— Moyenne : 8,64 ; écart-type : 5,38.

$\geq 3,73$	75%
$\geq 5,00$	66,2%
$\geq 8,89$	50%
$\geq 10,00$	42,6%
$\geq 13,23$	25%
$\geq 15,00$	14,7%

Les deux épreuves ont été choisies de façon équilibrée avec 61% pour étude de cas informatique et 39% pour fondements de l'informatique. Il y a eu d'excellentes copies dans les deux épreuves. Par contre, il apparaît que le vivier des candidates se présentant à chacune des épreuves ne soit pas semblable. Les deux épreuves ont donné lieu à des moyennes assez différentes. Cet écart se reflète dans les autres épreuves : les candidates ayant choisi étude de cas informatique ont eu une moyenne inférieure de plusieurs points à celle des candidates ayant choisi fondements de l'informatique, et ceci sur les deux autres épreuves.

2.3.1 Étude de cas informatique

Remarques générales

Le sujet s'inspire d'un projet concernant des livraisons par des véhicules. La partie 1 présente la problématique et s'attache à des considérations algorithmiques et de programmation objet en Python. L'idée de la partie 2 était d'explorer un protocole applicatif de communication multi-sauts entre véhicules. Un grand nombre de réponses n'ont pas su faire la différence entre un protocole applicatif et le protocole TCP. La partie 3 explore la concurrence et la partie 4 concerne la gestion de bases de données. La partie 5 s'intéresse à la programmation web.

Il est attendu que tout code intègre des commentaires faisant le lien avec le sujet, par exemple pour `fusionnable` et ses 3 critères. On s'étonne que certaines candidates ne sachent pas écrire de code Python, JavaScript, HTML ou SQL, alors que tous ces langages sont au programme. Au delà de la maîtrise syntaxique d'un langage de programmation, il est aussi attendu que tout code soit lisible et intelligible par une tierce personne. À cette fin, le recours à des fonctions/attributs intermédiaires/privés avec un libellé explicite peut être une aide utile à la compréhension d'un code. De même, il est plus pertinent de placer un code source sur une nouvelle page que de vouloir à tout prix le faire rentrer dans un petit espace en bas d'une feuille.

Remarques par question

Question 1. L'utilisation de langage haut niveau, tel que Python, ne doit pas faire oublier de penser aux éléments de base, tels que la gestion mémoire. Ainsi, beaucoup de copies semblent penser que `dist = A` (un simple alias) permet de modifier librement `dist` sans impacter le contenu de `A`.

Question 2. Il s'agit ici de dérouler l'algorithme sur un exemple facile pour bien le comprendre. Quant à la solution optimale sur cet exemple, une explication était attendue.

Question 3. Une explication de la réponse était attendue.

Question 5. Cette question permet de constater si l'algorithme naïf a bien été compris : pour aller d'un sommet A vers son voisin B, l'algorithme prend la distance la plus courte selon la matrice des distances (en passant éventuellement par plusieurs sommets). Il était attendu que les contre-exemples soient expliqués.

Question 6. La fonction `distance` est absente d'un grand nombre de copies. La fonction `fusionnable` devait contenir au moins trois tests au vu du sujet. Seules certaines copies ont fait l'effort d'expliciter le lien avec le sujet avec des commentaires.

Question 7. Il n'était pas nécessaire de réimplémenter une fonction de tri ; expliciter la clé du tri était suffisant. Le sujet explicite que la fonction doit renvoyer aussi la longueur totale des tournées, ce qu'un grand nombre de copies n'a pas fait.

Question 12. Le but de cette question était d'évaluer la compréhension de la mise en place d'une communication TCP et plus particulièrement la notion de *Timeout*. Des réponses du type « Ça ne marche pas » ne sont pas satisfaisantes.

Question 13. L'utilisation du terme croissance exponentielle à la place du nombre infini (ou non borné) de messages échangés a été souvent retrouvée mais n'est pas adaptée.

Question 14. Cette question a mené à une grande confusion entre le protocole applicatif et le protocole TCP. Ici, plusieurs messages concernant la même information peuvent être retransmis par plusieurs ordinateurs de bordure. Il faut donc les traiter au niveau du serveur au niveau applicatif.

Question 15. Le cadre de cet exercice étant les communications sans fil, tout ordinateur à portée reçoit le message. La question était donc liée à la réaction à la réception du message et non pas comme dans plusieurs réponses sur l'envoi ciblé à un ordinateur particulier.

Question 16. Plusieurs propositions supposent la possibilité pour chaque véhicule d'avoir accès à la liste de l'ensemble des autres véhicules ou de pouvoir les contacter individuellement. Le cadre de la question était lié aux communications sans fil, donc avec une couverture de zone et non point à point, et aussi une connaissance uniquement du voisinage proche.

Question 17. Peu de réponses ont proposé une évaluation de performance, même simpliste.

Question 18. Plusieurs copies ont mélangé le niveau où se situe le problème (ici des interférences physiques) avec le niveau auquel il est possible de le résoudre (qui est donc un niveau supérieur dans la classification).

Question 19. Un grand nombre de réponses ont supposé que tout ordinateur pouvait accéder directement à un ordinateur de bordure. De plus, les détails fournis des propositions ne permettaient pas toujours de les évaluer. Par exemple, regrouper les messages est une bonne piste de réponse, mais il est alors nécessaire d'expliciter une notion de *Timeout*, de nombre maximal de messages, d'identifiant unique pour ne pas regrouper plusieurs fois le même message…

Question 20. Cette question explore les notions de virtualisation, d'émulation ainsi que leur impact sur les performances.

Question 21. Le code de la classe `Acces` implémente un mutex. Les explications fournies ont été claires dans la majorité des copies.

Question 22. Il s'agit ici d'un exemple de schéma de synchronisation au programme : lecteurs-rédacteurs (ici avec priorité aux rédacteurs).

Question 23. Un faible nombre de copies a pu expliquer le cas de famine suivant une arrivée régulière de lecteurs. Plusieurs copies ont proposé des configurations liées à des erreurs de programme, ce qui n'était pas dans le cadre demandé.

Question 24. Il s'agit ici de l'écriture d'une requête simple avec fonction d'agrégation. Le bon ordre des mots-clés SQL est requis.

Question 28. Il s'agit ici de reconnaître l'impact du choix de modélisation sur l'utilisation de la base de données, ainsi que les problèmes de cohérence de données qui peuvent apparaître.

Question 30. La lisibilité grâce au retour à la ligne est utile et attendue, même en HTML. Pour cette question, il était attendu que la réponse mette bien en évidence la structure d'un tableau ainsi que le choix approprié d'un bouton d'interaction.

Question 31. Si le langage JavaScript offre beaucoup de liberté quant aux constructions applicables pour mettre en oeuvre un algorithme, il était attendu que les candidates rédigent un code concis permettant d'itérer sur toutes les lignes du tableau qui font référence à une heure en manipulant les ancres de style appropriées pour que l'affichage se mette à jour selon les 3 situations identifiées par le sujet.

Question 32. Cette question couvrait spécifiquement la problématique de la gestion des événements dans JavaScript.

Question 33. Le jury a observé peu de réponses à cette question pour laquelle il était attendu de démontrer la maîtrise de l'API HTTP de JavaScript.

2.3.2 Fondements de l'informatique

Remarques générales

Le sujet de cette épreuve consistait à étudier différents modèles de réseaux de neurones formels, et à déterminer leur puissance en tant que modèles de calculs. Il visait à relier ce modèle à différents modèles plus classiques de l'informatique fondamentale : circuits booléens, automates finis, machines à compteurs, machines de Turing, et à voir que chacun de ces derniers modèles correspond à une certaine classe de réseaux de neurones formels.

Dans la partie I, on se focalisait sur le cas d'un unique neurone formel dont la fonction d'activation est la fonction de Heaviside (neurone à seuil). On devait dans un premier temps (sous-partie 1) étudier comment calculer certaines fonctions booléennes, et montrer que des fonctions comme la fonction parité ne peut pas se représenter de cette façon. On discutait alors la représentation de ces fonctions, en utilisant des coefficients entiers (sous-partie 2). On étudiait alors la question de l'apprentissage des coefficients d'un neurone à seuil, en étudiant l'algorithme classique du Perceptron proposé en 1957 par Frank Rosenblatt, en démontrant sa convergence selon les arguments proposés par Novikoff en 1962. Dans la partie II, on commencait à étudier des réseaux de neurones, avec des questions de programmation en OCaml, et l'observation que ces réseaux permettent de coder les fonctions booléennes, et que l'on peut toujours supposer que les réseaux sont organisés en couches. Puis, on cherchait à établir qu'il y a a des fonctions comme la fonction parité qui se calculent de façon beaucoup plus succinte avec un réseau de neurones par rapport aux circuits booléens. Dans partie III, on montrait la NP-complétude du problème de l'apprentissage des coefficients d'un réseau de neurones à partir d'un ensemble d'exemples. Dans la partie IV, on reliait les réseaux de neurones récurrents aux automates finis, en jouant avec différentes façons de donner les entrées à un calcul. Dans la partie V, on reliait les réseaux de neurones dont la fonction d'activation est la fonction ReLU (unité linéaire rectifié) aux machines à compteurs, et les réseaux de neurones dont la fonction d'activation est la fonction linéaire saturée aux machines de Turing.

La plupart des copies ont très bien compris le sujet, et traité de façon convenable les parties I et II. La partie III a inspiré beaucoup moins de copies, en dehors des questions les plus élémentaires. La partie IV a été traitée par un nombre plus grand de candidates, cependant trop souvent sans rentrer suffisamment dans les détails. En particulier, les candidates ne distinguent pas avec suffisamment de clarté le raisonnement pour les différentes variantes possibles d'automates finis correspondants,

alors que c'était parfois précisément l'objet de plusieurs des questions. Les parties IV et V ont été traitées globalement par très peu de candidates, en dehors des questions de programmation par les machines à compteur.

Globalement, trop de copies évitent les questions nécessitant du code OCaml ou les traitent dans une syntaxe plus qu'approximative. La nature inclusive des bornes des boucles OCaml ne semble pas connue de nombreuses candidates. Trop d'inégalités strictes se transforment en inégalités larges au gré de la rédaction, et trop de copies ne font pas attention au signe des quantités par lesquelles sont multipliées les quantités dans les inégalités.

Remarques par question

Question 2. Certaines candidates ne tiennent pas compte du signe du seuil, en supposant qu'on a toujours $2h \geq h$, ce qui rend leur raisonnement faux.

Question 3. De nombreuses candidates affirment que la composition de fonctions booléennes linéaires à seuil en est encore une. Les candidates font régulièrement une confusion entre XOR_2 et sa négation PARITY_2.

Question 4. Cette question, pourtant relativement simple, a conduit à de nombreux développements extravagants. Certaines candidates considèrent le minimum d'un ensemble de vecteurs en dimension n. Certaines candidates ne traitent que l'une des deux implications dans la définition de la séparabilité. Et certaines candidates pensent que la fonction linéaire à seuil prend la valeur 0 pour tous les points de l'ensemble fini. Pour les réponses correctes, la question de distinguer l'ensemble vide pour en prendre un maximum ou un minimum n'a presque jamais été faite.

Question 5. Il est impératif d'utiliser le fait que les coefficients par lesquels on multiplie une inégalité sont positifs pour conserver la correction de cette inégalité, ici la stricte positivité de $\frac{\delta}{\lambda}$.

Question 6. Beaucoup de candidates ont compris l'esprit de la question.

Question 7. Très peu de copies ont pensé à utiliser la partie entière. Pour les rares candidates qui y ont pensé, la formalisation s'est révélée en général fort maladroite.

Question 8. Trop de candidates appliquent le résultat de la question 7 sans en vérifier les hypothèses.

Question 10. Le cas de l'égalité au seuil a été ignoré par trop de candidates, qui n'ont pas vu le lien avec le concept de réprésentation δ-séparable des questions précédentes.

Question 11. Il fallait faire attention au fait que X était à deux dimensions et donc ne pas oublier le 0 supplémentaire.

Question 12. Cette question a souvent été bien traitée. Attention en général (pas seulement dans cette question) aux $<$ qui se transforment en $\leq$ sans raison dans les copies de plusieurs candidates.

Question 13 et 14. Ces questions ont été très peu traitées.

Question 15. Cette question était facile, mais il était demandé de bien préciser les couches, ce que toutes les candidates ne font pas.

Question 16. Cette question de programmation a été assez bien traitée par les candidates qui l'ont faite. Attention toutefois à la manipulation des couples en OCaml.

Question 17. Il ne fallait pas oublier les portes constantes et la manière de les réaliser.

Question 18. Le jury a constaté beaucoup de réponses informelles, alors que l'on attendait une discussion propre de la conservation de la profondeur. La nature (poids, seuil) des portes intermédiaires est souvent peu expliquée.

Question 19. La formule du produit d'une matrice par un vecteur semble mal connue des candidates et conduit souvent à des programmes faux.

Question 21. Cette question a été traitée de façon correcte dans la plupart des copies qui l'ont abordée.

Question 22. Le calcul du nombre de portes AND est souvent trop informel. Le cas NOT - OR - AND est très rarement traité. Plusieurs copies affirment qu'il est impossible d'obtenir une forme normale conjonctive pour PARITY_n.

Question 24. Si la plupart des candidates ont vu qu'il fallait utiliser les portes de la question précédente pour compter le nombre d'entrées à 1, très peu ont proposé une couche de profondeur 2 qui permettait de conclure.

Question 26. Il faut penser à justifier que le certificat est de taille polynomiale (nombre fini de cas notamment pour les valeurs de seuil) en la taille du circuit. Une discussion sur le sujet, qui était attendue, n'a presque jamais été faite. Quelques copies pensent que NP signifie exponentiel.

Question 28. Cette question a souvent été bien traitée, sauf par quelques copies qui n'ont pas compris qu'on attendait une condition nécessaire et qui se sont appuyées sur leur solution de la question 27.

Question 29. Une étude exhaustive des cas était attendue, mais les correctrices n'en ont presque pas vu.

Question 30. Cette question n'a presque jamais été traitée.

Question 31 et 32. Les quelques réponses proposées sont très difficiles à suivre. Certaines copies ne semblent pas comprendre l'énoncé même de la question.

Question 33. Il ne fallait pas oublier le neurone de décision.

Question 34. Cette question n'a presque jamais été traitée. Il était important de justifier le caractère fini du nombre d'états et pas seulement de décrire ce que ces états représentent.

Question 35. La question est souvent bien traitée par les candidates qui ont tavaillé sur cette partie du sujet.

Peu de copies traitent les questions ultérieures, à l'exception des questions 38, 39, 40 et 41 et 45 traitées par certaines des candidates de globalement de façon satisfaisante.

Chapitre 3

Épreuves orales

Pour alléger le texte, ce chapitre est uniquement écrit au masculin [1], mais cela doit être pris dans le sens du neutre : l'ensemble de ce texte, sans exception, concerne uniformément candidates et candidats.

Les épreuves orales se sont déroulées du 17 au 22 juin 2023 au lycée Paul Valéry à Paris. Les candidats admissibles ont été convoqués pour les trois épreuves orales sur trois jours consécutifs. Le jury est bien conscient du stress et de la fatigue des candidats avec des temps de préparation longs et des horaires parfois décalés. Son objectif n'est pas de piéger les candidats mais de les mettre dans un contexte leur permettant de montrer le meilleur d'eux-mêmes.

Le jury a jugé l'oral de l'agrégation d'un très solide niveau global, à la fois en matière de connaissances disciplinaires et de compétences pédagogiques.

Le jury rappelle que les livres sont disponibles pour les préparations de toutes les épreuves. La liste des livres fournis par le jury est disponible sur le site `https://agreg-info.org` et sera mise à jour pour la session 2024. Les préparations à l'agrégation ont elles aussi mis à disposition des livres, également accessibles à toutes les candidats. L'environnement informatique à disposition des candidats (pour les trois épreuves) est celui téléchargeable sur le site du jury et les fichiers des candidats sont sauvegardés régulièrement sur un serveur. Il est à noter que les responsables informatiques ne sont pas là pour aider les candidats à débugger leur code, mais peuvent intervenir en cas de souci avec l'environnement à disposition. Consulter la documentation disponible dans cet environnement doit être le premier réflexe.

De nombreux conseils et remarques s'appliquent à l'ensemble des épreuves orales et sont regroupés ici.

Les trois épreuves sont distinctes et permettent au jury d'évaluer des points différents. Il est donc requis des candidats qu'ils connaissent la description et les attendus des trois épreuves.

Des temps maximaux sont donnés pour les trois épreuves et rappelés au début de chaque oral. Il est important d'essayer de se rapprocher le plus possible de ces temps sans toutefois les dépasser : même s'ils sont indiqués comme des temps à ne pas dépasser (le jury interrompra le candidat le cas échéant), ils sont en fait des temps-cibles.

Le jury d'une épreuve d'oral n'a pas connaissance des résultats de l'écrit du candidat et s'impose une discipline stricte de ne pas discuter des prestations d'un candidat devant les autres membres du jury avant que celui-ci ait passé l'ensemble de ses épreuves. Lorsqu'un candidat se présente à une épreuve, il est ainsi assuré d'être examiné sans aucun préjugé.

Il est souhaitable d'éviter l'auto-dénigrement. Même si le candidat a l'impression que l'interrogation se passe mal, il est difficile pour lui de se comparer aux autre candidats. Il faut de plus rappeler que le jury évalue plusieurs dimensions (disciplinaire, pédagogique, réactivité).

Il est attendu que le candidat adopte une posture d'enseignant lors des interrogations orales, sans toutefois considérer que le jury doit nécessairement adopter une posture d'apprenant. Des

1. Le chapitre 2 est écrit au féminin.

présentations didactiques et illustrées sont notamment attendues. Le jury apprécie notamment les remarques pédagogiques.

Les épreuves orales ne sont pas un jeu de rôle dans lequel le candidat joue au professeur et le jury joue à la classe indisciplinée. Au cours de son exposé ou de ses réponses aux questions du jury, le candidat ne doit en aucun cas faire référence à un élément de compréhension qu'il aurait expliqué dans une séance antérieure comme il pourrait le faire avec des élèves qu'il suivrait sur une année complète. Lorsque le jury pose une question, la réponse « vous n'avez pas été attentif lors du cours de la semaine dernière » est dilatoire et à éviter.

Le candidat est responsable de ce qu'il présente au jury, quelle que soit sa provenance.

Le jury peut poser des questions simples au candidat, qui ne doit pas chercher systématiquement des réponses compliquées. Le jury peut aussi interroger sur un spectre thématique plus large mais en lien avec le thème de l'épreuve en cours, par exemple en posant des questions liées à l'impact de la hiérarchie mémoire sur le comportement d'un programme, même si le thème de l'épreuve est plus algorithmique. Le jury vise à recruter des informaticiens *complets*. Les questions peuvent permettre d'évaluer la capacité des candidats à faire des liens avec d'autres sous-domaines que celui étudié spécifiquement par le sujet de l'oral.

Le jury est susceptible d'interrompre un candidat pendant sa réponse à une question, sans que cela ne doive être perçu négativement, pour différentes raisons : il a obtenu la réponse attendue, il souhaite enchaîner sur une autre question sur le même thème, il estime que le temps pour répondre risque d'être trop long, …

Les candidats ne doivent pas chercher un sens caché ou figuratif aux questions du jury. En cas de doute, il peut être constructif pour un candidat de reformuler la question avant d'y répondre. Le jury ne manquera pas de reformuler la question à son tour si les deux diffèrent.

Un futur professeur doit pouvoir répondre à brûle-pourpoint à des questions de dimensionnement. Or, de nombreux candidats trébuchent sur des questions simples d'ordre de grandeur : par exemple, combien d'opérations un ordinateur peut-il effectuer en une seconde, quelle est la taille caractéristique d'une mémoire, quel est le temps caractéristique dans un protocole de communication réseau, combien y a-t-il de pixels dans une photo, etc. Dans ce genre de question, le jury se satisfait parfaitement d'une réponse sous la forme d'une puissance de 10 (ou de 2) et d'une unité.

Le jury rappelle que la gestion du tableau est également évaluée. Lors de la session 2023, les candidats disposaient d'un tableau blanc et de quatre couleurs de feutre effaçable (noir, bleu, vert et rouge). Un contenu propre, clair et organisé, éventuellement utilisant des couleurs, sera valorisé. Les candidats peuvent effacer le tableau ou une partie de tableau après avoir demandé l'autorisation du jury. Ils peuvent toutefois s'autoriser à effacer ce qui n'est pas du contenu (une faute d'orthographe, un trait mal tracé, …). Le jury s'attend à ce que le candidat économise l'espace du tableau et structure son occupation, pour éviter tant que possible de réclamer à effacer.

Dans les épreuves au cours desquelles le candidat est amené à projeter du code au tableau, le jury abaisse et relève instantanément l'écran blanc à la demande du candidat. L'écran de projection pouvant cacher une partie du tableau, le candidat est invité à anticiper l'usage des différentes zones du tableau et sa chronologie, dès son entrée dans la salle d'examen. Il est, par exemple, admissible que le candidat commence son exposé sur la partie du tableau jamais cachée par l'écran et exploite initialement l'autre zone pour projeter son code sur l'écran déroulé, puis, en fin d'exposé, fasse relever l'écran définitivement et écrive sur la partie de tableau restante.

Il est à noter qu'en 2023, l'ensemble des interrogations étaient ouvertes aux visiteurs.

3.1 Leçon

— 45 présents
— Meilleure note : 19,25/20
— Moyenne : 10,44 ; écart-type : 4,62.

$\geq 5,00$	84,4%
$\geq 6,50$	75%
$\geq 10,00$	50%
$\geq 14,75$	25%
$\geq 15,00$	22,2%

Le candidat se voit proposer (aléatoirement) deux sujets au choix ; il doit choisir de traiter l'un des deux, mais n'a pas à justifier son choix sur ce point. Il acte son choix au début de l'interrogation proprement dite et peut donc changer d'avis en cours de préparation (il va sans dire que ce n'est toutefois guère conseillé). Le jury n'interroge pas, bien entendu, sur le sujet qui n'a pas été choisi.

Si la leçon d'agrégation permet d'évaluer les compétences didactiques des candidats, elle est cependant un oral scientifique avant tout. Cela a des conséquences sur les trois moments de l'oral :
— le plan doit comporter du contenu scientifique, et non seulement l'annoncer ;
— le développement doit bien sûr montrer des qualités d'exposition, mais appliquées à un contenu scientifique clair et précis ;
— les questions posées par le jury peuvent porter sur la façon dont on expliquerait telle chose à tel type d'élève, mais elles portent également fréquemment sur le fond des notions abordées, au niveau de recul M2 attendu des candidats.

Sur chaque point, la maîtrise et le recul didactique sur le contenu scientifique est apprécié et valorisé, mais il s'agit de la cerise sur le gâteau : il faut d'abord qu'il y ait un gâteau.

Par ailleurs, l'agrégation est un concours de recrutement d'enseignants : une expertise sur des domaines hors programme ne se substitue pas à un recul pédagogique.

Plan

Le jury souhaite voir proposé par le candidat un plan de la leçon d'au plus trois feuillets A4 manuscrits. Il est demandé de rédiger le plan dans un format spécifique permettant la photocopie pour le jury dans de bonnes conditions. Le fichier pdf correspondant se trouve sur le site du jury. Des feuilles à ce format sont bien entendues mises à disposition lors de la préparation.

La dénomination traditionnelle de *plan* peut être trompeuse : **le plan d'une leçon d'agrégation n'est pas une table des matières ni une simple annonce de la structure de la leçon. Il doit comporter, outre cette structure, le contenu scientifique correspondant.** Ainsi, dans une leçon sur l'algorithmique du texte, un candidat ne peut pas simplement, par exemple, écrire dans son plan « motif », il doit choisir une définition formelle de cette notion et l'écrire ; de même, s'il choisit de mentionner l'algorithme de Rabin–Karp, il ne peut se contenter d'en donner le nom, il doit en écrire une description formalisée. Il en va de même des propriétés, théorèmes, résultats de complexité, etc. En effet, le jury souhaite pouvoir, dans la phase de questions, interroger, par exemple, la cohérence ou l'intérêt du choix de formalisation de la notion de motif vis-à-vis de la formulation retenue pour l'algorithme de Rabin–Karp. L'inobservance de ces prescriptions emporte une triple peine pour le candidat, car
— le format de l'épreuve n'est pas respecté,
— le plan est souvent superficiel et sa présentation tourne court et
— au lieu d'interroger sur les liens entre les éléments du plan, le jury en est réduit à demander d'expliciter les éléments qui auraient dû l'être dans le plan, ce qui est peu intéressant et surtout bien plus périlleux.

On attend des candidats qu'ils indiquent explicitement dans le plan les développements proposés. Lors de la présentation orale du plan, le jury ne souhaite pas que le candidat recopie les grandes lignes du plan en restant dos au jury.

Développement

Le développement peut traiter de sujets très variés : démonstration d'un théorème, présentation d'un protocole et formalisation de ses propriétés, exposition rigoureuse d'un algorithme, etc.

À titre d'exemple, dans ce dernier cas, le déroulé de l'algorithme sur un exemple peut apporter une plus-value pédagogique au développement, mais il ne saurait en aucun cas en constituer le cœur : le jury attend des résultats précis sur la correction, la complexité, les performances, etc.

De même, un développement centré sur un théorème, comme le fait que tout algorithme de tri par comparaison a une complexité au pire $\Omega(n \log(n))$, bénéficie d'une présentation et d'une explication du raisonnement, mais celui-ci doit aussi être formalisé : certes, on ne démontrera probablement pas *chacun* des lemmes (relation entre nombre de feuilles et taille dans un arbre binaire strict, etc.), mais, le résultat n'étant pas d'un niveau très élevé par rapport à ces lemmes, il est malvenu de ne pas prouver rigoureusement *une partie* d'entre eux.

Le jury préfère un développement soigné et précis plutôt qu'un développement plus large mais montré seulement partiellement ou mal maîtrisé.

Le jury apprécie des suggestions de travaux pratiques, démonstrations ou codes qui seraient une partie du cours correspondant.

Questions

De nombreuses questions simples visent simplement à vérifier que le candidat comprend et maîtrise ce qu'il écrit ; elles sont posées assez systématiquement et le candidat ne doit pas en être déstabilisé. Certaines questions peuvent porter sur l'enchaînement des éléments du plan ou sur sa cohérence interne. Le jury s'attend à des notations homogènes de la part du candidat, même s'il a utilisé plusieurs sources adoptant des conventions de notation ou des définitions légèrement différentes. Si le travail d'homogénéisation n'est pas fait dans le plan, le candidat sera interrogé et invité à le faire pendant l'échange avec le jury. Le candidat doit également faire attention au développement logique de son cours : il peut être maladroit d'utiliser des notions essentiellement introduites postérieurement dans le plan. Le jury ne manquera pas de questionner ces choix.

Les candidats doivent bien écouter les questions qui leur sont posées, et notamment bien différencier, par exemple :

— *Comment expliqueriez-vous l'algorithme de Rabin–Karp à un bon élève de terminale NSI ?* : de nombreuses réponses sont possibles ; on peut notamment envisager de dérouler l'algorithme à la main sur un exemple bien choisi ;

— *Pouvez-vous expliquer l'algorithme de Rabin–Karp ?* : on attend une réponse scientifique de même nature que pour un oral de master.

S'ils présentent une « méthode générale » dans leur plan, les candidats peuvent s'attendre à ce que le jury leur fournisse un exemple ou un cas d'usage à dérouler. Par exemple, dans une leçon sur la *programmation dynamique* où le candidat expose une manière générale d'obtenir une équation de récurrence entre sous-problèmes, le jury peut proposer sous forme d'un petit exercice un exemple de problème où il faut d'abord trouver cette équation et les bons sous-problèmes.

3.2 Travaux pratiques

— 44 présents
— Meilleure note : 20/20
— Moyenne : 12,38 ; écart-type : 4,33.

$\geq 5,00$	93,2%
$\geq 9,67$	75%
$\geq 10,00$	70,5%
$\geq 12,67$	50%
$\geq 15,00$	31,8%
$\geq 16,33$	25%

Cette épreuve orale a pour objectif principal l'évaluation de compétences pédagogiques des candidats. Elle se décompose en deux parties : les *Travaux pratiques* proprement dits et la *Revue de code*. Dans la première partie, le candidat doit produire et présenter du code, et justifier sa correction (à l'aide de tests, de preuves, etc.). Dans la seconde, le candidat est confronté à du code de mauvaise qualité et défectueux, comme pourrait en écrire un élève durant une séance de travaux pratiques, et doit proposer des corrections ou améliorations, comme le ferait un enseignant durant une telle séance.

Dans les deux cas, l'accent est mis sur la pédagogie et non sur la compétence disciplinaire ou la virtuosité, les codes à écrire ou à commenter étant conçus pour ne pas poser de difficultés techniques insurmontables aux candidats.

Comme pour les deux autres épreuves orales, on attend une conduite claire et organisée de l'exposé, avec mise en contexte (qui doit être présente sans être excessive) et présentation structurée des résultats. L'usage du tableau doit être au service de la compréhension du discours du candidat, et celui d'une vidéo-projection doit se concentrer autour de la projection de code, de l'exécution de tests et éventuellement de la présentation (éventuellement sous forme graphique) de résultats d'exécution. Une présentation basée sur des diapositives n'est pas souhaitée.

Le candidat doit veiller à respecter l'équilibre entre les deux parties de l'épreuve et, en particulier, ne pas passer trop de temps sur la revue de code.

Partie Travaux pratiques

Dans cette partie, lors de la production du code, l'accent doit être mis sur la clarté et la facilité de compréhension, ce n'est pas une épreuve de virtuosité ou de vitesse. Durant la préparation de cette épreuve, le candidat devrait avoir en tête des questions comme "Mon code peut-il être présenté à des élèves ?", voire "Est-il digne de figurer dans un manuel ?" Pour cela, des éléments auxquels il faut porter une attention particulière sont le choix des noms de variables (conventions de casse par exemple), l'usage de commentaires, la factorisation des expressions ou encore l'organisation spatiale du code (indentations, sauts de lignes).

Le candidat doit particulièrement travailler la réflexion concernant les choix de structures de données lorsqu'on lui demande de les proposer ; en effet, un mauvais choix peut avoir de lourdes conséquences.

Durant la présentation, les candidats doivent éviter une présentation linéaire du code produit. À la place, ils sont encouragés à décrire la structure de leur code et les principaux points clés, le jury pouvant ensuite, durant les questions, demander au candidat de revenir sur des portions spécifiques du code ou de développer plus en détail un point technique. Il est possible, voire recommandé, de structurer le code produit en plusieurs fichiers. Une utilisation d'un langage en respectant ses usages et spécificités est appréciée. Par contre, une utilisation excessive de bibliothèque peut nuire à la compréhension du code.

Il est mal vu pour un candidat d'essayer d'impressionner le jury en présentant des optimisations excessives de code si elles ne sont pas utiles. Par exemple, réécrire une fonction récursive afin d'avoir de la récursivité terminale n'est pas pertinent si le nombre d'appels récursifs de cette fonction reste peu élevé. L'accent doit être mis sur la production d'un code simple, lisible, maîtrisé, dont le fonctionnement est parfaitement compris et expliqué.

Une fois le code présenté, il importe de transmettre l'idée que celui-ci est correct. Cela repose tout d'abord par une exécution du code produit qui doit être conforme aux spécifications. Ainsi, le jury s'attend à des exécutions, sur des exemples bien choisis, du code produit. Durant la phase de questions, il peut demander de nouvelles exécutions, avec par exemple du code compilé en direct, ou sur des entrées de son choix.

Mais la justification de la correction d'un code ne s'arrête pas à des exécutions correctes sur des exemples spécifiques, et d'autres techniques doivent être mises en œuvre, par exemple une preuve formelle ou l'utilisation de tests.

Concernant les preuves formelles, sans forcément utiliser une formalisation excessive, l'utilisation et la justification de variants et d'invariants bien choisis peuvent contribuer efficacement à convaincre du bon comportement d'un algorithme.

Une autre méthode pouvant simplement être mise en œuvre et exposée lors de l'oral repose sur l'usage de jeux de tests. Une présentation exhaustive de ceux-ci n'est *a priori* pas pertinente, et le candidat doit au contraire transmettre l'idée que le choix des tests utilisés se place dans une démarche méthodologique rigoureuse et construite, et éviter d'en faire une simple énumération. Enfin, l'utilisation d'outils d'analyse simples est appréciée.

Les candidats peuvent être questionnés sur tous les aspects de leur code, ainsi que sur toutes les phases de développement. Concernant le code, ils peuvent être interrogés sur les structures de données choisies, les algorithmes mis en place, le typage, la complexité, les performances, etc. Sur le processus de développement, le jury s'attend à ce que les candidats connaissent les processus de compilation, d'exécution, de test et de déboguage.

Partie Revue de code

Durant cette partie, le candidat est confronté à des éléments de code défectueux et maladroits, dans un ou plusieurs langages parmi ceux du programme, et doit les étudier afin d'en proposer des améliorations (correction d'erreurs, restructuration, etc.), comme le ferait un enseignant avec un élève durant une séance de travaux pratiques.

De même que lors des autres épreuves orales, le jury attend un exposé structuré de la revue de code, avec une mise en contexte et une présentation organisée et motivée des corrections à apporter plutôt qu'une énumération séquentielle. Le jury apprécie que le candidat en profite pour rappeler et appliquer de bonnes pratiques de programmation.

Bien sûr, il est attendu que le code corrigé soit correct, celui-ci doit donc avoir été exécuté et testé et le jury peut, durant les questions, demander explicitement une exécution du code dans les conditions de son choix.

Rappelons enfin qu'il est demandé au candidat de corriger le code proposé, en essayant de rester au plus près du code initial (et donc de l'intention de l'élève) et non d'en proposer une version sans rapport avec le code initial, même si celui-ci est correct. L'objectif n'est pas de voir si le candidat est capable de produire le code demandé mais, dans un objectif pédagogique, de voir comment il peut aider un élève pour que celui-ci soit capable d'écrire correctement un tel code.

3.3 Modélisation

— 45 présents
— Meilleure note : 17,50/20
— Moyenne : 10,66 ; écart-type : 4,03.

$\geq 5,00$	93,3%
$\geq 6,75$	75%
$\geq 10,00$	53,3%
$\geq 11,00$	50%
$\geq 14,75$	25%
$\geq 15,00$	20,0%

Forme et structure de l'oral

L'heure d'oral se décompose en deux parties. Premièrement, un exposé dont la durée cible est de 35 minutes. Celui-ci prend la forme d'un cours d'ouverture à destination d'élèves de niveau L1/L2 ou CPGE. Le cadre d'énonciation repose sur une triple fiction :

— le jury n'a pas connaissance du texte proposé au candidat ;
— le candidat a trouvé ce texte tout seul lors de recherches pour préparer son exposé ;
— le candidat a formulé lui-même les pistes de réflexion qui sont proposées à la fin du texte (ou bien : ces pistes sont présentes dans les brouillons d'un collègue, et le candidat les reprend à son compte pour les développer).

On ne demande donc pas un exposé sur le texte, mais sur le sujet traité par le texte. Il n'y a donc pas lieu de citer le texte et encore moins de s'y référer. L'exposé doit être structuré et il est fortement conseillé de présenter son plan en début d'oral après une courte introduction. L'exposé doit également inclure la présentation argumentée d'une ou plusieurs illustrations sur ordinateur, ainsi que la discussion d'une dimension éthique, sociétale, environnementale, économique ou juridique telle que mentionnée par l'arrêté.

La deuxième partie de l'oral consiste en un échange avec le jury.

Contenu de l'exposé

Il est attendu des candidats d'une part une présentation et une discussion du problème et de sa formalisation, d'autre part une capacité à développer, compléter, voire améliorer (ou critiquer) les ébauches de solution esquissées. Par exemple, lorsque l'on fait des études de complexité asymptotique d'algorithmes présentés pour être appliqués à des cas concrets, le jury attend que le candidat soit capable de raisonner en ordres de grandeur. Est-ce que la complexité de cet algorithme est un problème étant donné la taille attendue de l'entrée ? Jusqu'à quelle taille d'entrée cet algorithme peut-il raisonnablement être appliqué ?

Outre la capacité à mobiliser ses connaissances dans un contexte concret, le jury cherche également à évaluer la communication scientifique et pédagogique, c'est-à-dire la capacité de tenir un discours structuré, cohérent et pédagogique sur un sujet non trivial. Pour cela, le candidat doit construire un exposé fondé sur le texte fourni. Il ne s'agit pas de faire une lecture commentée du texte, mais bien de construire un exposé autonome. En particulier, les candidats ne doivent supposer du jury aucune connaissance préalable du texte. Le jury n'attend pas forcément un traitement exhaustif de tout ce qui est abordé dans le texte et dans les pistes, mais l'esprit du document doit être respecté.

L'exposé du candidat doit prendre la forme d'un cours d'ouverture à destination d'élèves niveau L1/L2 ou CPGE. L'évaluation de l'utilisation du tableau est partie intégrante de l'évaluation des compétences didactiques et pédagogiques des candidats. Le jury attend que le candidat montre sa capacité à utiliser le tableau pour transmettre des notions et expliquer des exemples bien choisis. Le vidéo-projecteur n'est utilisé que ponctuellement durant la présentation. Il sert à projeter l'illustration informatique et éventuellement une figure ou un algorithme. En revanche, il ne doit pas être utilisé comme support principal de l'exposé, pour projeter un plan, un diaporama, des définitions ou des énoncés de résultats.

Les illustrations doivent s'intégrer à l'exposé de manière cohérente et la plus fluide possible. L'exercice porte en particulier sur la communication, et on attend donc une présentation efficace d'une ou plusieurs illustrations qui éclairent réellement un aspect du texte. Il ne s'agit pas d'une seconde épreuve de TP : en cas de programmation, les attentes sont modestes. Le jury sera plus sensible à la pertinence de l'illustration qu'à la complexité de l'implémentation, même s'il restera toujours attentif à la qualité du code présenté. Le candidat peut choisir d'insister plus ou moins sur la programmation (ou d'ailleurs mettre en place une illustration qui n'implique pas de programmation proprement dite), mais le reste de l'exposé doit rester équilibré et contenir une présentation détaillée et précise d'une solution esquissée par le texte, qu'il s'agisse d'une preuve, d'un algorithme, de l'architecture d'une solution matérielle ou logicielle, etc. En cas de programmation, le jury attend que le candidat exécute ses programmes durant la présentation afin de vérifier leur fonctionnalité.

De même qu'à l'épreuve de leçon, le jury est attaché à la capacité des candidats à prendre du recul et envisager l'informatique comme une discipline et non comme un patchwork de domaines ; la capacité à s'appuyer sur le texte pour faire apparaître des liens dans d'autres sous-domaines que celui étudié par le texte constitue une vraie preuve de maturité scientifique. Insistons toutefois sur le fait que le discours doit rester dans le cadre du texte, et que ce dernier ne doit pas servir de simple prétexte à une longue digression dans un domaine où le candidat se sent plus à l'aise.

Les candidats doivent également intégrer une discussion courte et synthétique d'une dimension éthique, sociétale, environnementale, économique ou juridique. Cette ouverture ne suit pas nécessairement une piste proposée par le jury, toute initiative personnelle pertinente ayant d'ailleurs

vocation à être valorisée.

L'épreuve de modélisation nécessite une réflexion préalable des candidats sur les différents aspects de l'épreuve : construction d'un véritable exposé, utilisation du tableau, place de l'illustration informatique, conclusion du propos... On pourra, à cette fin, s'appuyer sur les exemples de sujets donnés aux sessions 2022 et 2023 publiés sur le site agreg-info.org.

Questions

L'interrogation se poursuit ensuite par les questions du jury, qui peuvent porter sur l'ensemble de la présentation du candidat, sur l'ensemble des sujets du programme se rattachant à celle-ci, ou encore sur les dimensions éthiques, sociétales, environnementales, économiques ou juridiques en lien avec le texte. Ce temps est l'occasion pour le jury d'affiner sa perception de la compréhension du texte par les candidats, de faire préciser ou corriger des points, de fournir des indications pour permettre au candidat de développer certains aspects qui lui ont résisté, ou encore de proposer des pistes de réflexions, des prolongements, ou des variantes du problème étudié par le texte. Comme pour les séances de questions des autres épreuves, il est primordial que le candidat adopte une posture facilitant les échanges. Il doit écouter les questions jusqu'au bout et répondre de façon concise et la plus précise possible. Bien que cela soit parfois nécessaire, il est difficile et désagréable pour le jury de devoir couper la parole d'un candidat qui s'est lancée dans une réponse très longue.

Remarques générales sur la session 2023

L'épreuve de modélisation de cette session reposait sur deux sujets portant sur des thèmes différents : l'analyse de complexité en prenant en compte la prédiction de branchement et l'impact de la réduction du nombre de couleurs sur la taille des images stockées au format png. Les candidats ont plutôt bien compris le format de l'épreuve. Il reste tout de même quelques exceptions qui poussent le jury à rappeler des conseils d'ordre général.

- Il est important de présenter un plan de son exposé dans son introduction.
- L'épreuve de modélisation n'est pas une seconde épreuve de TP, les illustrations informatiques ne doivent pas représenter l'essentiel de la présentation. En revanche, elle doivent servir le propos, mettre en lumière des phénomènes et ne pas être considérées comme de simples exercices de programmation.
- L'épreuve de modélisation est une épreuve d'informatique, même si les interactions avec d'autres disciplines scientifiques sont très intéressantes, ce sont bien les dimensions informatiques qui doivent être au cœur de l'exposé. En particulier, il n'est pas judicieux de passer trop de temps sur les aspects très mathématiques de certains sujets.
- La gestion du tableau est un enjeu important de l'épreuve. Il ne faut pas utiliser le tableau comme une feuille de brouillon, mais bien comme un support pédagogique.
- La gestion du temps est également un enjeu important de l'épreuve. Il est par conséquent déconseillé de passer beaucoup de temps à recopier du code fourni dans le texte.

Le jury a été satisfait du niveau global des candidats. Malgré tout, certaines maladresses et lacunes sont apparues à plusieurs reprises.

- Le jury attend une posture d'enseignant de la part des candidats. Ils doivent en particulier parler et écrire dans un français correct, en utilisant un vocabulaire précis et rigoureux. Par exemple, les termes "aléatoire" et "optimal" ont été utilisés de façon très maladroites. En particulier, l'optimalité n'est pas une échelle de valeur. On n'est pas plus ou moins optimal.
- Certains candidats ont montré des fragilités concernant l'analyse de complexité et les ordres de grandeur. Le logarithme en base 2 et la racine carrée ne doivent pas être confondus.

Remarques sur le sujet *Algorithmique et prédiction de branchement*

Le premier sujet portait sur l'analyse de l'impact de la prédiction de branchement sur la complexité et les performances de certains algorithmes. Le sujet a été bien compris et bien traité dans l'ensemble, mais beaucoup de candidats ont traité l'épreuve un peu comme une suite d'exercices.

Certains ont ainsi fait les implémentations attendues, mais sans s'intéresser aux performances obtenues. Le jury attendait plutôt que les candidats considèrent vraiment la problématique et construisent donc leur exposé en fonction avec un réel souci d'aller observer empiriquement les impacts sur les temps de calcul des programmes.

Remarques sur le sujet *Réduction du poids d'une image*

Le second sujet portait sur différentes façons de réduire le nombre de couleurs dans une image et de l'impact que cela a sur le poids des images au format png. De la même façon que pour le premier sujet, certains candidats ont traité le sujet comme une suite de petits exercices à assembler autour du thème des images. Le jury attendait plutôt que la question de l'impact sur le poids des images soit centrale. Plusieurs illustrations informatiques n'ont pas du tout traité cette dimension alors même qu'il aurait simplement suffi d'aller voir le poids des images générées.

MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

Concours externe de l'agrégation du second degré

Section informatique

Liste des leçons pour la session 2024

La liste des sujets de leçons pour la session 2024 est la suivante :

1. Exemples de méthodes et outils pour la correction des programmes.
2. Paradigmes de programmation : impératif, fonctionnel, objet. Exemples et applications.
3. Tests de programme et inspection de code.
4. Principe d'induction.
5. Implémentations et applications des piles, files et files de priorité.
6. Implémentations et applications des ensembles et des dictionnaires.
7. Accessiblité et chemins dans un graphe. Applications.
8. Algorithmes de tri. Exemples, complexité et applications.
9. Algorithmique du texte. Exemples et applications.
10. Arbres : représentations et applications.
11. Exemples d'algorithmes d'approximation et d'algorithmes probabilistes.
12. Exemples d'algorithmes glouton et de retour sur trace.
13. Exemples d'algorithmes utilisant la méthode « diviser pour régner ».
14. Programmation dynamique.
15. Exemples d'algorithmes d'apprentissage supervisé et non supervisé.
16. Exemples d'algorithmes pour l'étude des jeux.
17. Algorithmes d'ordonnancement de tâches et de gestion de ressources.
18. Gestion et coordination de multiples fils d'exécution.
19. Hiérarchie mémoire. Structure et performances.
20. Mémoire : du bit à l'abstraction vue par les processus.
21. Problèmes et stratégies de cohérence et de synchronisation.
22. Stockage et manipulation de données, des fichiers aux bases de données.
23. Fonctions et circuits booléens en architecture des ordinateurs.
24. Principes de fonctionnement des ordinateurs : architecture, notions d'assembleur.
25. Échanges de données et routage. Exemples.
26. Client-serveur : des sockets TCP aux requêtes HTTP.
27. Architecture d'Internet.
28. Modèle relationnel et conception de bases de données.
29. Requêtes en langage SQL.
30. Analyses lexicale et syntaxique. Applications.
31. Classes P et NP. Problèmes NP-complets. Exemples.
32. Décidabilité et indécidabilité. Exemples.
33. Formules du calcul propositionnel : représentation, formes normales, satisfiabilité. Applications.
34. Langages rationnels et automates finis. Exemples et applications.