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

Liberté
Égalité
Fraternité

Rapport du jury

Concours : agrégation externe

Section : Informatique

Session 2022

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 première édition de l'agrégation externe d'informatique, qui s'est déroulée en 2022. 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 2022, les **épreuves écrites** se sont déroulées du 7 au 9 mars 2022, organisées par les académies :

- Composition d'informatique : 7 mars 2022 de 9 heures à 14 heures,
- Étude d'un problème informatique : 8 mars 2022 de 9 heures à 15 heures,
- Épreuve spécifique (selon l'option choisie : étude de cas informatique ou fondements de l'informatique) : 9 mars 2022 de 9 heures à 15 heures.

Les **épreuves orales** se sont déroulées du 16 au 24 juin 2022 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].

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

Pour cette session, la question de la **communication** avec les candidates, candidats, préparatrices et préparateurs a été au centre de nos préoccupations. Les médias utilisés ont été :

— le site web du jury `https://agreg-info.org`, régulièrement mis à jour. Il contient en particulier une FAQ (foire aux questions) et un formulaire de soumission de demandes à la FAQ qui a permis de l'étoffer significativement.

— une liste de diffusion par email, qui rappelle les dates et informe des ajouts importants au site web.

— un webinaire ouvert à tous, qui a eu lieu le jeudi 25 novembre 2021 de 14h à 16h, avec publication ultérieure des transparents et du compte-rendu.

Nous avons également ouvert les interrogations aux visiteurs sur la dernière session (du 22 au 24 juin) avec 95 visiteurs uniques sur les trois jours.

La liste de diffusion et le site web sont conservés pour la session 2023. Un webinaire est également prévu début décembre 2022.

1.2 Statistiques

En résumé, et en avant-propos des statistiques, 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 3 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.

Le jury tient à féliciter toutes les admises et tous les admis, car ils ont toutes et tous démontré à la fois maîtrise disciplinaire, qualités didactiques et pédagogiques et une grande aptitude à communiquer à l'écrit et à l'oral. Le jury a classé 2 personnes ex-æquo en première place avec un niveau excellent sur tous les points.

Les premiers chiffres sont bien sûr ceux des candidatures résumés ici :

Inscrits	Présents	Admissibles	Admis liste principale	Liste complémentaire
549	251	55	20	3

En ce qui concerne les admises et les admis, nous avons volontairement fusionné pour la suite, la liste principale (20 places) avec la liste complémentaire (3 places).

Mis en regard du nombre de postes, le nombre de présentes et présents aux trois épreuves fait de l'agrégation externe d'informatique une agrégation fortement sélective, avec environ 12 candidats présents par poste.

En ce qui concerne les épreuves écrites, la moyenne des candidates et candidats présents aux 3 épreuves est de 6,83/20 et celles des candidates et candidats admissibles de 12,97/20 avec une **barre d'admissibilité à 10,47/20**. Les épreuves écrites ont donc été très sélectives pour aboutir à seulement 55 admissibles.

En ce qui concerne les épreuves orales uniquement, la moyenne des candidates et candidats présents aux 3 épreuves est de 11,02/20 et celles des candidates et candidats admis sur liste principale de 14,62/20. Au vu de la très grande qualité des candidates et candidats, le jury a souhaité proposer une liste complémentaire de 3 noms. Pour l'ensemble des épreuves, cela donne des **barres d'admission à 12,86/20 pour la liste principale et 12,58/20 sur la liste complémentaire**.

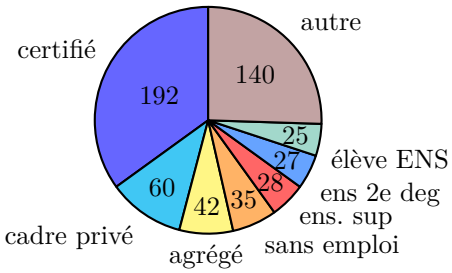
L'ensemble de ces indicateurs numériques montre que la première session de l'agrégation d'informatique a été extrêmement sélective.

1.2.1 Statistiques par statut

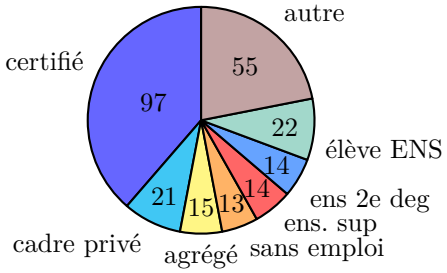
Il est important de rappeler que ce tableau se base sur les statuts indiqués, de manière purement déclarative, par les candidates et candidats lors de l'inscription. Aucune vérification n'est opérée sur ce point, qui fait uniquement l'objet de la présente exploitation statistique.

Statut	Inscrits	Présents	Admissibles	Admis & LC
certifié	192	97	9	2
cadres sect privé conv collect	60	21	1	0
agrégé	42	15	3	0
sans emploi	35	13	3	0
enseignant du supérieur	28	14	3	2
ens. stagiaire 2d deg. col/lyc	27	14	2	0
élève d'une ENS	25	22	21	16
salariés secteur tertiaire	18	5	0	0
professions libérales	14	7	1	0
étud. hors espe (sans prépa)	13	7	2	1
contractuel 2d degré	13	2	0	0
pers fonction publique	11	2	1	0
contract. enseignant supérieur	10	5	2	0
salariés secteur industriel	8	3	0	0
étud. hors espe (prepa mo.univ)	7	6	4	2
formateurs dans secteur privé	5	1	0	0
ag non titulaire fonct publiq	5	2	0	0
PLP	4	3	0	0
pers enseig tit fonct publique	4	2	0	0
maitre contr. et agree rem tit	4	2	0	0
assistant d'éducation	3	0	0	0
vacataire enseignant du sup.	2	1	0	0
maitre auxiliaire	2	1	0	0
fonct stagiaire fonct publique	2	1	0	0
étudiant en espe en 2e année	2	1	0	0
étudiant en espe en 1re année	2	2	1	0
étud. hors espe (prépa cned)	2	0	0	0
vacataire apprentissage (cfa)	1	0	0	0
professeur écoles	1	0	0	0
professeur associe 2d degré	1	0	0	0
peps	1	0	0	0
militaire	1	1	1	0
maître délégué	1	0	0	0
étud. hors espe (prépa privée)	1	0	0	0
chaire supérieure	1	0	0	0
agent admi. membre ue (hors fra)	1	1	1	0

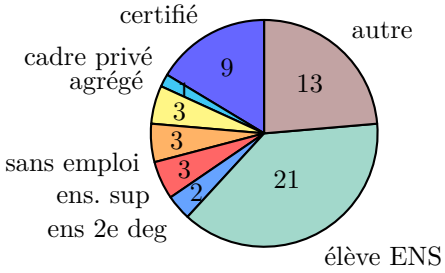
On peut remarquer la diversité des profils. Les certifiés sont les plus nombreux, mais le secteur privé ainsi que les sans emploi sont également nombreux. Nous n'avons malheureusement pas l'information de la matière du CAPES des candidates et candidats certifiés, ni l'information de s'ils et elles enseignent déjà NSI. L'essentiel des admis se déclarent élèves d'une ENS.



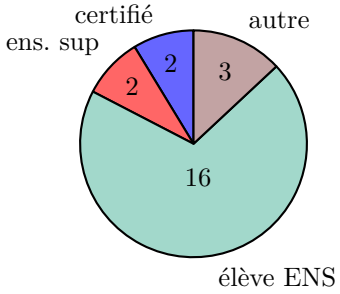
Inscrits par statut.



Présents par statut.



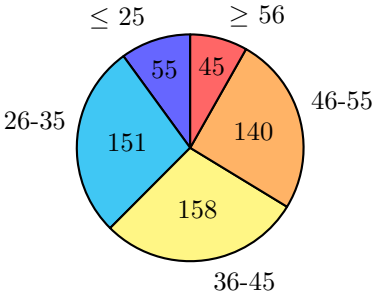
Admissibles par statut.



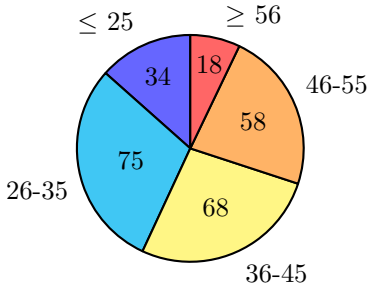
Admis par statut.

1.2.2 Statistiques par âge

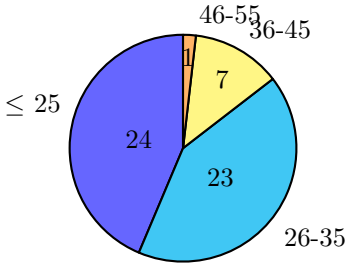
Âge	Inscrits	Présents	Admissibles	Admis & LC
≤ 25	55	34	24	15
26-35	151	75	23	7
36-45	158	68	7	0
46-55	140	58	1	1
≥ 56	45	18	0	0



Inscrits par âge.



Présents par âge.



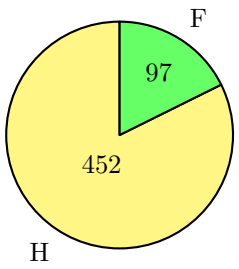
Admissibles par âge.

1.2.3 Statistiques par genre

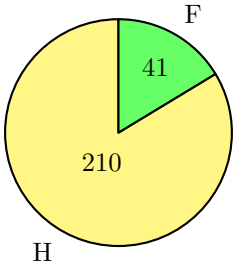
Le genre est lui aussi déclaratif et ne permet à l'inscription que les deux choix « M. » et « Mme ».

Genre	Inscrits	Présents	Admissibles	Admis & LC
M	452	210	51	20
F	97	41	4	3

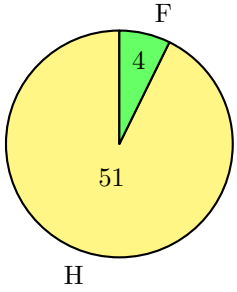
La représentation des femmes en sciences et en informatique en particulier reste très faible. Le taux de femmes inscrites est assez faible et les femmes ont plutôt moins bien réussi les épreuves écrites (sachant que les copies sont anonymes). Par contre, les femmes admissibles ont plutôt mieux réussi les épreuves orales. Il est délicat de faire des analyses sur un échantillon aussi réduit, probablement corrélé avec d'autres facteurs statistiques mais le genre reste une préoccupation du jury et ce point sera regardé dans les prochaines années.



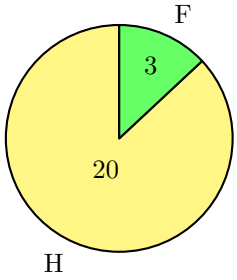
Inscrits par genre.



Présents par genre.



Admissibles par genre.

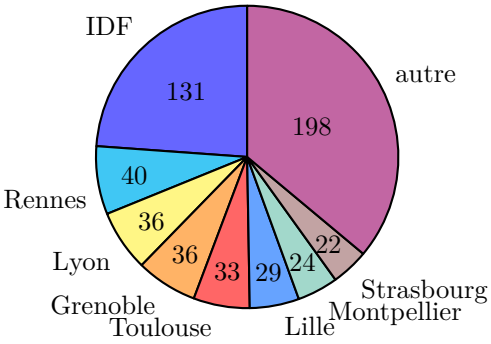


Admis par genre.

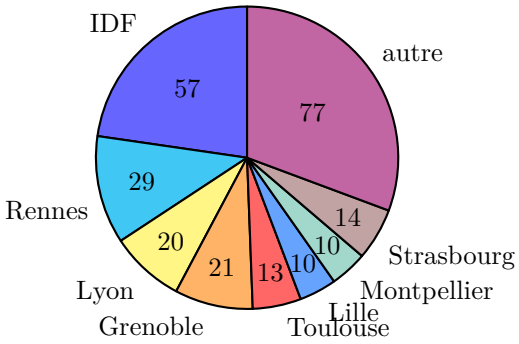
1.2.4 Statistiques par académie

Académie	Inscrits	Présents	Admissibles	Admis & LC
Créteil-Paris-Versailles	131	57	18	7
Rennes	40	29	15	10
Lyon	36	20	6	4
Grenoble	36	21	4	1
Toulouse	33	13	2	0
Lille	29	10	1	0
Montpellier	24	10	2	1
Strasbourg	22	14	1	0
Aix-Marseille	19	9	0	0
Nice	18	5	0	0
Rouen	15	6	0	0
Nantes	15	7	0	0
Nancy-Metz	14	6	3	0
Poitiers	13	5	1	0
Bordeaux	13	8	0	0
La Réunion	12	4	0	0
Amiens	10	4	0	0
Dijon	8	4	0	0
Reims	7	3	0	0
Orlénas-Tours	7	4	1	0
Caen	7	4	1	0
Nouvelle-Calédonie	6	3	0	0
Mayotte	6	3	0	0
Limoges	5	1	0	0
Martinique	4	1	0	0
Guyane	4	1	0	0
Guadeloupe	4	1	0	0
Corse	4	2	0	0
Clermont-Ferrand	3	1	0	0
Besançon	3	2	0	0
Polynésie Française	1	1	0	0

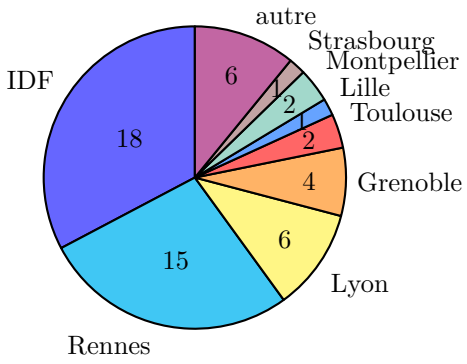
Sur ce point, le jury tient simplement à souligner l'existence d'exactement 3 préparations à l'agrégation en France en 2022, situées à Lyon, Paris et Rennes.



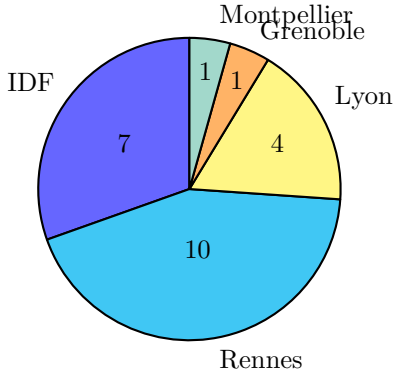
Inscrits par académie.



Présents par académie.



Admissibles par académie.



Admis par académie.

1.3 Un peu d'histoire

La création d'une section informatique au concours de l'agrégation externe a été annoncée le 9 mars 2021 par M. Jean-Michel Blanquer, ministre de l'Éducation nationale, de la jeunesse et des sports, lors d'une audition à l'Assemblée nationale. Cette création s'inscrivait dans le prolongement de la création du CAPES « Numérique et sciences informatiques » ouvert à la session 2020 et dans un mouvement plus général de développement de l'enseignement de l'informatique dans l'enseignement scolaire (création de l'enseignement SNT et de la spécialité NSI) et en classes préparatoires (ouverture de la filière MP2I/MPI à la rentrée 2021).

Cette création a été programmée pour la session 2022, dans un calendrier très resserré. Pour répondre au défi posé par ce calendrier, un groupe de travail a été constitué conjointement par la DGRH et l'inspection générale, sous la coordination de M. Olivier Sidokpohou, inspecteur général de l'éducation, du sport et de la recherche. Ce groupe de travail, chargé de proposer au ministre une maquette pour le concours, était composé, outre le coordinateur, de M. Yannick Alméras, inspecteur général de l'éducation, du sport et de la recherche ; M. Jean-Marie Chesneaux, inspecteur général de l'éducation du sport et de la recherche ; Mme Nadine Collineau, sous-directrice du recrutement à la direction générale des ressources humaines ; Mme Pascale Costa, inspectrice générale de l'éducation, du sport et de la recherche ; Mme Isabelle Debled-Rennesson, professeure des universités ; Mme Christine Gaubert-Macon, inspectrice générale de l'éducation, du sport et de la recherche ; Mme Christine Froidevaux, professeure des universités émérite ; M. Guillaume Hanrot, professeur des universités ; M. Benoît Martin, chef du bureau des affaires générales, réglementaires et des systèmes d'information à la direction générale des ressources humaines.

Ce groupe de travail a débuté ses travaux le 18 mars 2021 et a rendu ses conclusions le 23 avril 2021, conclusions qui ont conduit à la création formelle de la section informatique du concours de l'agrégation externe par l'arrêté du 17 mai 2021. Cet arrêté acte en particulier la structure du concours, mais aussi le

fait que le programme du concours se construit autour des contenus des programmes des enseignements d'informatique de l'enseignement secondaire et des classes post-baccalauréat des lycées, auquel s'adjoint un programme complémentaire défini par le jury.

Suite à ce premier groupe de travail, un second groupe a été sollicité pour concevoir ledit programme complémentaire, dont les travaux se sont échelonnés du 12 mai 2021 au 21 juillet 2021. Ce groupe était coordonné par Mme Sylvie Boldo, directrice de recherche, et composé de M. Xavier Blanc, professeur des universités ; M. Jean-Marie Chesneaux, inspecteur général de l'éducation, du sport et de la recherche ; Mme Isabelle Debled-Rennesson, professeure des universités ; M. Georges Da Costa, maître de conférence des universités ; M. Marc de Falco, professeur agrégé ; M. Jean-Christophe Filliâtre, directeur de recherche ; Mme Christine Froidevaux, professeure des universités émérite ; M. Rahim Kacimi, maître de conférence des universités ; M. Xavier Leroy, professeur au Collège de France ; Mme Alix Munier, professeure des universités ; Mme Amélie Stainer, professeure agrégée. Le programme complémentaire a été publié en ligne sur le site `devenirenseignant.gouv.fr` le 6 septembre 2021.

1.4 Remerciements

Le jury du concours 2022 souhaite saisir l'occasion de ce rapport pour remercier très vivement l'ensemble des collègues mentionnées et mentionnés ci-dessus pour leur participation à l'ensemble de ce processus dans un calendrier très resserré. Il souhaite également remercier celles et ceux qui auraient été oubliés par inadvertance (en les priant de bien vouloir l'excuser pour cet oubli), ainsi que, plus largement, tous ceux qui ont contribué à la création de cette agrégation et au succès de cette première session : en particulier, et de manière non limitative, la Société Informatique de France, les établissements ayant fait le pari d'ouvrir des préparations, ainsi que l'ensemble des préparateurs et préparatrices ayant construit des formations ambitieuses dans un délai là encore très resserré (et avec une information qui est arrivée de manière progressive et parfois hélas tardive).

Merci aux 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, et nous ont permis de tenir cette première session du concours dans des conditions optimales.

Le jury a reçu avec reconnaissance l'aide d'autres concours, à la fois en terme de conseils pratiques, en terme d'aide technique (sur l'environnement informatique pour les candidates et candidats), et même en terme de livres donnés. Un grand merci au jury de l'agrégation de mathématiques, en particulier à sa présidente et son précédent président. Merci aussi au jury du CAPES NSI, en particulier à sa présidente et à ses responsables informatiques.

La présidence du jury remercie également très chaleureusement les membres du jury pour leur travail dans la définition et la création de ces nombreuses épreuves, la qualité de la correction et de l'interrogation, leur ouverture d'esprit et leur implication sur cette édition où nous avons tout inventé.

Chapitre 2

Épreuves écrites

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.

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 *informaticiens complets*, capables à la fois de réflexes sains dans un ensemble de domaines assez larges de l'informatique (épreuve 1), et dotés 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 candidats, 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'un candidat aux fonctions de professeur qu'il sache rédiger une copie. En conséquence, un passage rayé ou hachuré ne sera jamais lu. De plus, quand plusieurs versions sont proposées par le candidat, le correcteur ne prend pas la meilleure. C'est au candidat de choisir ce qu'il présente à l'évaluation du jury, pour le meilleur et pour le pire.

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

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 un futur agrégé 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.

2.1 Épreuve 1

Le jury a été peu satisfait de cette épreuve. Beaucoup de candidats ont négligé une ou plusieurs des parties pour se concentrer sur d'autres où ils sont plus à l'aise. Le jury rappelle avec force son attachement au recrutement d'informaticiens *complets* et rappelle que tous les candidats doivent être à l'aise avec toutes les thématiques du programme.

La meilleure note ainsi que la moyenne de cette épreuve montrent que trop de candidats ont des lacunes. Les deux dernières parties, traitant de la logique et les réseaux, certes placées en dernière position, ont en particulier été trop souvent mal ou peu traitées.

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

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 candidats ayant une note supérieure à 5, 10 et 15.

— 259 présents
— Meilleure note : 14,84/20
— Moyenne : 6,37 ; écart-type : 2,86.

$\geq$ 4,85	75%
$\geq$ 5,00	64%
$\geq$ 6,00	50%
$\geq$ 8,30	25%
$\geq$ 10,00	12%
$\geq$ 15,00	0%

Notes sur l'architecture du sujet. À l'instar du sujet 0, le sujet proposé pour la session 2022 rassemble des exercices balayant largement l'informatique, allant de champs plutôt mathématisables (exercice 2, exercice 3) à des champs relevant plutôt de problématiques dites « de bas niveau » (exercice 1 et exercice 4). Il est important, pour un candidat, de démontrer des connaissances fondamentales dans ces différents champs plutôt que de chercher à traiter intégralement un seul ou même deux exercices (et ce d'autant plus que ces deux exercices sont « proches » thématiquement ou méthodologiquement parlant).

Notes sur la rédaction et la présentation. On attend d'un candidat aux fonctions de professeur qu'il sache rédiger une copie ; en outre, des informaticiens sont censés comprendre que la correction des copies étant un processus de masse, elle bénéficie du respect d'un format fixé et commun. Dans le cadre d'une épreuve comprenant des parties totalement indépendantes (notamment l'épreuve 1), cela signifie ne pas mélanger les réponses à ces différentes parties. La plupart des copies est bien organisée, mais le jury regrette de devoir préciser à l'intention de quelques candidats qu'il n'est pas convenable de répondre, dans cet ordre, à l'exercice 1 de la partie 1, puis à la partie 2, puis à la partie 4, puis à l'exercice 4 de la partie 1. Même au sein d'une partie ou d'un exercice, il convient d'éviter de changer l'ordre des questions.

2.1.1 Partie 1

Exercice 1

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.

Cependant, aucune des questions de cet exercice n'appelle de long développement. En particulier, la question h n'invite pas à rappeler l'histoire des systèmes de gestion de fichiers depuis 1960. De même, il est possible — si on y tient — de signaler, en prélude aux réponses données aux questions a et d, qu'on se place dans l'hypothèse d'un *prompt* standard, mais disserter sur les possibilités de configurer ce *prompt* est inopportun. Face à certaines copies présentant en partie 1 de tels développements surabondants mais faisant l'impasse sur la partie 2, 3 ou 4, le jury considère que le format de l'épreuve n'a pas été compris ou en tout cas pas respecté.

En question c, un petit nombre de candidats prétend que `untel` serait membre des groupes `guntel` et `otel`. En fait, rien dans les données fournies ne permet d'asseoir cette thèse.

En question f, le jury attend que soit précisée la nature de lien symbolique de `fc.txt` ; parler de lien n'est pas suffisant car il existe aussi des liens durs ; parler de raccourci n'est pas adapté. Le jury a été décontenancé par les réponses à cette question qui voient dans le premier triplet `rwx` les droits de `root` ou plus vaguement de l'« administrateur », ce qui laisse envisager des fichiers que `root` ne pourrait pas lire ou modifier. D'autres candidats parlent du « créateur » du fichier au lieu de son propriétaire, et on lit parfois qu'un triplet de droits (le premier ou le deuxième, selon les copies) représente les droits de l'utilisateur courant. C'est pour le moins surprenant.

En h, il ne suffit pas d'évoquer le caractère multi-utilisateur : il faut mentionner les conséquences de ce caractère soit pour la sécurité, soit pour la confidentialité, soit pour le partage.

Exercice 2

Répondre aux questions de cet exercice nécessite de calculer. Sans calculatrice, il est acceptable de laisser $12 + 256 + 256^2 + 256^3$ sous cette forme. Il est inutile d'exprimer ce résultat sous la forme $12 + 2^8 + 2^{16} + 2^{24}$. Il est très dommageable de donner comme résultat $12 + 256^6$.

En revanche, l'absence de calculatrice ne saurait justifier qu'on ne mène pas le calcul de la division euclidienne de 4567 par 2048, ni de celle de 573448 par 2048. De trop nombreux candidats trouvent dans ce dernier cas un résultat non seulement faux, mais d'un ordre de grandeur manifestement incorrect. Certains écrivent explicitement que, faute de calculatrice, il ne leur est pas possible de mener ce calcul ; une telle déclaration, dont on ne sait pas bien si elle constitue un aveu ou une critique, ne peut que susciter la consternation du jury. Les divisions posées sont en effet au programme de primaire.

Inversement, se borner à donner le résultat numérique sans indiquer la formule est risqué : aucun point n'est alors accordé s'il ne s'agit pas du résultat exact.

Exercice 3

Le texte étudié traite des disques durs. La plupart des candidats l'a compris. Certains y ont vu une discussion sur la mémoire vive. Cette interprétation est en effet possible, mais est plus difficile à manier correctement (elle est acceptée lorsque c'est le cas). En particulier, il est incongru de parler de fichiers si on choisit de se placer dans ce contexte.

En revanche, il n'est pas judicieux ici de parler à la fois de disque dur et de RAM, sans choisir : il en résulte dans le meilleur des cas un discours confus, non pédagogique. De même, il est pour le moins maladroit d'appeler première méthode celle de la deuxième secrétaire du texte, et inversement. Enfin, parler de mémoire sans précision est ambigu.

Les discussions sur les techniques associées aux disques durs magnétiques comparativement aux SSD ne sont pas sans pertinence dans l'absolu, mais elles font fi de la date de publication du texte étudié.

Il est nécessaire de bien analyser le sujet. Il est demandé d'indiquer ce que représente *la liste des tiroirs vides* et non pas ce que représentent *les tiroirs vides* : il ne suffit donc pas de parler de secteurs ou blocs libres ; il faut mentionner la table des inodes, qui permet de trouver ces blocs.

Concernant les exemples, le jury souhaite que l'on cite des systèmes de fichiers (FAT, NTFS, ext...) et non des systèmes d'exploitation.

Cet exercice est aussi l'occasion pour les candidats de montrer leur capacité à rédiger un texte en français, ce qui est indispensable à tout professeur. La plupart des candidats y est parvenu convenablement, mais dans certaines copies, l'orthographe tant lexicale que grammaticale ainsi que la ponctuation laissent franchement à désirer. C'est également dans ce genre d'exercice qu'une mauvaise écriture est pénalisante.

Certains candidats ont écrit des réponses longues de quatre pages, avec force digressions. C'est inutile, et la concision est aussi une qualité recherchée chez un professeur. Un tel développement est particulièrement mal perçu lorsqu'il voisine des parties non traitées.

On trouve des schémas dans certaines copies, avec parfois des éléments qui sont écrits verticalement. Cette dernière pratique est à éviter car, dans l'outil utilisé pour la correction, la rotation des pages est malaisée.

La citation de 1998 présente une vision stéréotypée des femmes, du fait de la métaphore de « la secrétaire » qui range un dossier dans des tiroirs. Le choix a été fait de garder le texte dans la forme sous laquelle il avait été publié. Le jury a été surpris de constater à cette occasion dans les copies d'autres stéréotypes probablement involontaires. Une minorité significative de candidats parlent de « petits garçons » pour la défragmentation, alors que cette épithète n'est pas dans le texte étudié. On trouve aussi quelques occurrences de « garçons musclés ». Le jury a ignoré ces ajouts pour noter cette question.

Exercice 4

En a.1, on doit observer que le programme crée une infinité de processus, ou crée des processus indéfiniment. Il est possible de critiquer, en plus, l'absence d'intérêt de ce programme, son indentation ou l'emploi de `for (;;)`. Mais il n'est pas utile de développer cette critique sur toute une page, surtout lorsque par ailleurs on fait l'impasse sur des parties entières de l'énoncé.

En a.2, le jury attend qu'on se place du point de vue de l'administrateur d'un système qui limite le nombre de processus qui peuvent être générés par les programmes qui s'y exécutent. Mais la question n'étant pas rédigée explicitement dans ce contexte, il a également accepté les solutions pertinentes consistant à modifier le programme proposé.

Le programme proposé en b n'est pas une façon pertinente de calculer x^n et la modification demandée en b.1 aboutit à une version pire encore, en raison du coût lié à la création des processus. Telle est l'analyse attendue en b.2. Cependant, il est hors sujet, en b.1, de remplacer le programme proposé par un autre sans rapport, même plus efficace : il est demandé de modifier le programme proposé.

Le jury loue l'effort de certains candidats qui proposent une forme de coloration syntaxique. Cependant, cet effort n'est pas nécessaire : une écriture soignée suffit à l'intelligibilité de la réponse. Le jury a apprécié en revanche qu'un changement de couleur soit utilisé pour signaler les parties modifiées du programme. Se limiter à écrire les éléments modifiés est risqué.

La plupart des candidats a fait des efforts pour la mise en page de ce long code, et le jury l'apprécie particulièrement. On a très souvent pris garde de ne pas commencer la réponse à b.1 en bas d'une page ; on ne doit pas pour autant caser la réponse à b.2 dans l'espace ainsi laissé libre en bas de cette page. Pour faire tenir le code sur une seule page, certains candidats adoptent une présentation en deux colonnes : elle doit alors être très nette ; il faut tracer une séparation à la règle.

Sur le fond, comme indiqué dans l'énoncé, le jury s'est montré tolérant avec les erreurs techniques. Deux points lui ont cependant paru rédhibitoires :

— la tentative de créer un « gros tube » par `int fds[3];` voire `int fds[4];` ;

— une séquence de la forme `pid1 = fork(); pid2 = fork();`, qui a pour conséquence que le premier fils effectue également le deuxième `fork`.

2.1.2 Partie 2

La partie 2 traite de l'arithmétique à virgule flottante. Le sujet donne tous les rappels nécessaires pour traiter l'exercice, de la répartition des bits du nombre à la définition des arrondis. Le but est ici de couvrir plusieurs propriétés et non-propriétés de l'arithmétique sur ordinateur.

L'ensemble de cette partie considère des nombres flottants exprimés en base 2. Aussi, il est incongru de rencontrer des puissances de 10, de e ou encore de 1 dans les réponses. Nous avons observé que le calcul en virgule flottante conserve une part de mystère pour les candidats.

— Le jury a noté beaucoup d'erreurs de calcul algébrique ou numérique de base.

— Aussi, beaucoup de confusions entre positif, strictement positif, négatif, strictement négatif, et leurs négations respectives, ont été observées.

— Les points de suspension sont à utiliser pour éviter de répéter un chiffre un grand nombre de fois. Aussi, il faut éviter d'écrire des nombres de 52 bits sans ellipse, surtout pour des nombres simples comme 0. Les notations ambiguës telles que $0\ldots1$ sont à éviter. Le nombre $1,\ldots1$ ne représente pas $1{,}0\ldots01$. Il est souhaitable aussi d'expliciter le nombre de bits compris dans l'ellipse : le nombre $0{,}0\ldots01$ écrit sans autre précision pourrait désigner n'importe quelle puissance négative de la base. Enfin, pour des nombres à plus de 1000 chiffres en binaire, on évitera une écriture binaire même avec des ellipses, et on préférera une expression algébrique sous formes de puissances de 2.

— Il est souhaitable de simplifier les résultats quand c'est possible, par exemple $2^{52} \cdot 2^{-1023}$, $2^{-52} - 2^{-53}$, $2/8$ ou 1×0.

D'autre part, on rappelle quelques résultats simples :

— L'opération $\oplus$ n'est pas l'addition entière. De plus, on ne peut pas additionner deux flottants terme à terme : ainsi, $((-1)^s \cdot 2^e \cdot f) + ((-1)^{s'} \cdot 2^{e'} \cdot f') \neq ((-1)^{s+s'} \cdot 2^{e+e'} \cdot (f + f'))$.

— Ajouter 1 à la représentation binaire d'un flottant interprétée comme nombre entier n'ajoute pas 1 à ce flottant.
— $2^1 \neq 1$, $2^0 \neq 0$, et $2^a + 2^b \neq 2^{a+b}$.

Certaines questions admettent plusieurs solutions, notamment 4a, 6, 7a, 8a, 9a. Lorsqu'un seul exemple est demandé, il est préférable de choisir le plus simple. Par exemple, le flottant 0 répond immédiatement aux questions 7a, 8a et 9a. Il est inutile et contre-productif de calculer l'ensemble complet des valeurs possibles quand un seul exemple suffit, d'autant plus si l'ensemble est incorrect. La réponse doit être formulée explicitement par un nombre : parler du « plus petit flottant strictement positif » n'est pas suffisant ; il faut donner sa valeur.

Pour cette première épreuve, le jury a été indulgent envers les réponses justes sans justification. Mais une justification brève est attendue, et peut permettre en tout état de cause de rattraper partiellement une erreur de calcul. Ainsi, ne pas justifier suffisamment ses réponses constitue une prise de risque. À l'inverse, il n'est pas nécessaire d'écrire une demi-page pour justifier un arrondi. On appréciera plutôt une justification brève citant le résultat exact et le nombre flottant le plus proche, et une application de la règle d'arrondi vers le flottant de fraction paire en cas de point milieu.

Question 1 Cette question a été bien traitée dans l'ensemble, mais beaucoup d'erreurs auraient pu être évitées par une lecture attentive de l'énoncé. Les notations $0.f$ et $1.f$ ont été mal comprises dans un nombre important de copies. Le point de $0.f$ représente le séparateur entre la partie entière et la partie fractionnaire. Ce n'est pas une multiplication. Le point a été choisi plutôt que la virgule utilisée en français par cohérence avec les langages de programmation utilisant cette notation.

Le sujet précise qu'il s'agit d'un nombre en binaire à virgule fixe, et que f est la fraction sur 52 bits. De fait, $0.f$ et $1.f$ représentent respectivement les nombres $0.f = \sum_{i=1}^{52} f_i \cdot 2^{-i}$ et $1.f = 1 + 0.f$ où f_i est le i-ième chiffre de f, de gauche à droite.

Ces nombres sont écrits en binaire et non en décimal : 0.001 représente 1/8 et non 1/1000 en décimal. Il n'est pas non plus correct d'éliminer des zéros en tête de f, qui sont significatifs : $0.001 \neq 0.1$.

Plusieurs candidats ont été surpris par l'ordre de grandeur très faible des nombres obtenus, de 0 à 2^{-1022}. Ce n'est pas une erreur de l'énoncé, l'exercice considérant volontairement des nombres flottants ayant une représentation simple en machine.

Question 2 Cette question a également été bien bien traitée. Elle demande d'écrire des nombres en binaire. Outre les erreurs d'interprétation de l'énoncé, les erreurs de conversion décimal/binaire ont constitué la principale difficulté rencontrée.

Le bit de poids fort d'un nombre entier sur n bits a pour poids $n-1$, et non n. Le nombre $1023 = 2^{10} - 1$ s'écrit en binaire sur 11 bits avec un zéro suivi de 10 uns, et non par deux zéros suivis de 9 uns, ni par 11 uns.

À la sous-question c, beaucoup de copies ont donné l'exposant correct 970 en décimal, mais n'ont pas su le convertir correctement en binaire. Le nombre 970 étant pair, voir une écriture binaire terminant par le bit 1 devrait alerter d'une erreur. Étonnamment, l'inverse a été également observé : une réponse correcte à la conversion « difficile » du nombre 970 mais des erreurs à la conversion « facile » de 1023.

Question 3 Cette question a été dans l'ensemble assez mal traitée. Un des exemples de l'énoncé contient une erreur que plusieurs candidats ont relevé. Le nombre flottant $0|0\ldots01|0\ldots0$ est interprété par l'entier 2^{52}, et non 2^{53} comme l'indique à tort l'énoncé. Cette coquille ne compromet pas la validité de l'énoncé et ne semble pas avoir affecté les réponses à la question.

Les réponses aux questions sont souvent peu précises et sans justification. Pour lever les ambiguïtés, il est important de nommer ses variables. En particulier, on demande de ne pas utiliser les mêmes noms de variables comme E et f pour désigner des nombres différents. De même, éviter d'écrire des égalités mathématiques telles que $E = E + 1$ ou encore $f = f$. Il suffit d'introduire de nouvelles variables $E' = E + 1$ et $f' = f$.

Question 4 La sous-question 4a a été bien traitée. Plusieurs décompositions sont possibles et sont également admises. Comme mentionné en préambule, nous conseillons toutefois de privilégier les solutions simples,

comme par exemple $1 = 1.2^0$, plutôt que $1 = 256.2^{-8}$. Il faut bien lire -2^{52} dans l'énoncé et non 2^{-52} ou -2^{-52} comme vu dans certaines copies.

La sous-question 4b a été un peu moins bien traitée. Attention à ne pas confondre l'entier n écrit sous forme d'une puissance de 2 avec l'exposant. L'entier immédiatement inférieur à 2^{53} est $2^{53} - 1$ et non 2^{52}.

Pour la sous-question 4c, la plupart des copies ont bien mentionné les flottants $1 - 2^{-52}$, 1 et $1 + 2^{-52}$. Le quatrième $1 - 2^{-53}$ est plus difficile à trouver. La réponse à la question précédente implique pourtant un nombre flottant qui est également de la forme $(1 - 2^{-53}) \cdot 2^k$.

Par dénombrement, il peut y avoir au plus 2^{64} flottants représentables sur 64 bits. Il ne peut donc pas exister 2^{1025} flottants distincts entre $1 - 2^{-52}$, 1 et $1 + 2^{-52}$ comme l'ont avancé certaines réponses. Enfin, 0 n'est pas non plus dans l'intervalle.

Question 5 Cette question a été bien traitée. La propriété qui n'est pas respectée par $\oplus$ ici est l'associativité. De nombreuses copies ont évoqué la commutativité, la distributivité, la transitivité ou la linéarité. Par ailleurs, $\oplus$ est une opération, et non une relation.

Question 6 Cette question a également été bien traitée. On rappelle cependant qu'une division par 0 n'entraîne pas en soi une boucle infinie ni la saturation de la mémoire.

Questions 7, 8 et 9 Ces trois questions ont été moyennement traitées. On observe que 0 répond aux questions 7a, 8a et 9a. Pour les parties b, nous avons noté beaucoup de bonnes réponses mal justifiées, et notamment peu de référence à l'arrondi pair. Des calculs de bornes non demandés ont été rajoutés. Pour la question 9b, la bonne réponse est 0. Cependant, plusieurs candidats ont supposé que cette valeur est à exclure. Enfin, plusieurs candidats ont donné la même réponse aux deux sous-questions a et b de chaque question, avec le risque de perdre tous les points des deux sous-questions.

Question 10 Peu de candidats ont obtenu des points sur cette question.

L'exercice demande de montrer que $x + y$ est un flottant selon la caractérisation introduite à la question 4, c'est-à-dire de l'exprimer sous la forme de $n \cdot 2^e$ avec n, et e des entiers tels que $|n| < 2^{53}$ et $-1074 \leq e \leq 970$. Quelques copies ont donné une démonstration élégante différente de celle que le jury attend.

L'inégalité $|x + y| \leq 2^{-1022}$ n'implique pas que $|x| \leq 2^{-1022}$ et $|y| \leq 2^{-1022}$. En effet, x et y peuvent être de signes différents. Par exemple, $x = (2^{52} + 1) \times 2^{-1022}$ et $y = -(2^{52}) \times 2^{-1022}$ satisfont les conditions de l'énoncé, mais ne sont pas sous-normaux ou bornés par 2^{-1022}.

La question demande de démontrer que $x + y$ est un flottant. Supposer *a priori* que $x + y$ est un flottant pour démontrer que $x + y$ est un flottant revient à une tautologie.

Question 12 Comme pour la question 10, il faut vérifier que le résultat est un flottant en montrant que ses composants sont dans les bornes.

2.1.3 Partie 3

La partie 3 a pour but de tester les connaissances autour de la logique et de la déduction naturelle. Dans l'ensemble, cette partie n'a pas été bien traitée. Peu de candidats ont abordé les questions reposant sur un raisonnement à partir de règles de déduction. Le manque de connaissances autour du raisonnement logique est très pénalisant car cette notion peut être complexe à appréhender pour les candidats. La déduction naturelle est au programme et les règles sont même rappelées. Il est inadmissible que tant de copies aient peu ou mal traité cet exercice. Malgré tout, il y a eu également de très bonnes copies. Cette partie est séparée en deux exercices.

Exercice 1 Cet exercice a pour but de modéliser en logique propositionnelle un ensemble de phrases écrites en langage naturel puis de prouver une propriété à partir de cette modélisation en utilisant la déduction naturelle.

Le cadre de la logique propositionnelle n'a pas posé de grandes difficultés, et la modélisation a été dans l'ensemble plutôt bien traitée. Cependant, certaines copies ont révélé un manque de connaissances sur la syntaxe nécessaire à la modélisation.

À l'inverse, comme évoqué plus haut, la déduction naturelle et la notion d'arbre de preuve ne sont pas maîtrisées dans un grand nombre de copies. Plusieurs copies ont proposé des preuves dans d'autres formalismes (mathématiques classiques, algèbre booléenne), ce qui n'est pas demandé. Il est important que les candidats répondent aux questions posées.

Exercice 2 Cet exercice est séparé en quatre questions. Trois des questions ont pour but de démontrer certaines tautologies en utilisant la déduction naturelle. La dernière demande un contre-exemple intuitif pour démontrer qu'une assertion (correspondant à une erreur de raisonnement) est fausse.

Dans l'ensemble, les trois premières questions ont été peu traitées, ou hors du cadre de la déduction naturelle comme demandé. Il y a eu des tentatives de prouver intuitivement les tautologies, mais souvent en utilisant dans les hypothèses les tautologies à démontrer. Cela montre un problème de rigueur de raisonnement dans ces copies.

La dernière question a été dans l'ensemble très bien traitée. Toutefois, plusieurs copies ont présenté des exemples que l'on peut difficilement qualifier d'intuitifs, comme par exemple concernant des limites de suite en mathématiques, et on aurait souhaité ici plus de clarté pédagogique.

Pour finir, certaines copies n'ont pas donné de contre-exemples mais une méta-explication des raisons de l'incorrection de l'assertion. Ceci n'est pas demandé.

Pour conclure, même si cette partie porte sur la logique et les arbres de déductions, il n'est pas interdit d'organiser les réponses à l'aide de phrases de présentation ou d'explications. Beaucoup de copies en manquent. La qualité pédagogique des copies est souvent insuffisante.

2.1.4 Partie 4

La partie 4 s'attache à tester les connaissances et compétences des candidats dans le domaine des réseaux, versant protocoles (questions 1 à 5) et routage (questions 6 et 7), et se conclut par une brève question autour de l'évaluation de performances.

Cette partie pose un problème de cohabitation de plusieurs trafics avec ou sans connexion dans un réseau de communication. Le réseau étudié est composé d'un segment de réseau local, d'un système de transport avec plusieurs nœuds (routeurs) et de serveurs distants. Les questions posées ont pour but d'évaluer les candidats sur plusieurs notions notamment l'encapsulation des services, le format de données (PDU), les modes de communication connecté et non-connecté, la fiabilité de la couche transports et les fonctions de la couche réseaux en particulier l'adressage, le routage et le contrôle d'erreur. Cette partie n'a pas été traitée par de nombreux candidats, peut-être par manque de temps.

Il est moins attendu des candidats des connaissances précises qu'une compréhension des mécanismes en œuvre et le jury a été globalement assez déçu. Beaucoup de candidats semblent avoir répondu au hasard sur les questions le permettant, ou ont paraphrasé les éléments donnés par l'énoncé sur DHCP ou DNS.

Question 1. S'agissant de DHCP, au-delà des trames demandées, la principale question est de savoir à quel moment l'adresse IP se détermine et, de ce fait, quels paquets sont émis en broadcast / unicast : c'est l'essentiel de ce qui est demandé. Subtilité ignorée des candidats, le serveur vérifie d'abord par un ping, avant de proposer une adresse, que cette adresse est bien disponible. En effet, les services encapsulés dans les trames ont été bien identifiés mis à part pour la trame 2 où le serveur DHCP vérifie quand même la disponibilité d'une adresse IP avant de l'offrir.

Question 2. Au-delà de la requête DNS nécessaire pour obtenir l'adresse IP du serveur, le jury s'intéresse à l'établissement de la connexion TCP entre les deux machines plus qu'aux requêtes HTTP proprement dites ; une bonne moitié des candidats n'a pas du tout décrit ce passage, passant directement de la requête DNS à une requête HTTP de type GET. Peu de copies ont identifié les bonnes trames post résolution DNS du non de domaine. L'objectif étant de décrire les étapes nécessaires à l'établissement d'une connexion WEB, peu

de candidats ont décrit le handshake de la connexion TCP. En effet la majorité s'est contentée de trames encapsulant la résolution du nom de domaine www.agreg-info.org.

Question 3. Beaucoup de candidats semblent répondre au hasard ; bien entendu, sans justification, ce n'est pas une option acceptable à une agrégation. On s'attend pourtant à ce que les candidats connaissent la différence entre UDP et TCP, qui figure explicitement au programme.

Question 4. Cette question explore différentes notions autour du format de paquet IPv4 et les fonctions de la couche réseau (IP) ; plusieurs questions admettent des réponses binaires tout en demandant une ou plusieurs justifications. Là encore, beaucoup de réponses « sèches » (oui/non) plus ou moins aléatoires : une réponse sèche sans justification ne saurait être valorisée. De manière générale, le jury regrette que ces notions soient peu comprises et peu maîtrisées par les candidats.

Question 5. Dans cette question, un nouveau trafic réseau s'ajoute avec une connexion FTP. Les justifications sont souvent insuffisantes.

HTTP et FTP sont tous deux fondés sur des connexion TCP. Il aurait fallu souligner la capacité de ce dernier à gérer de multiples connexions simultanément. En effet, plusieurs connexions sont possibles entre la machine A et les serveurs. Ces connexions diffèrent par le numéro de port local et par le numéro de port distant ; il n'y a donc pas de confusion possible. Le jury a remarqué aussi le très faible nombre de copies ayant fait référence aux Sockets des connexions. En effet, il est nécessaire de distinguer également les deux Sockets qui valent respectivement : < @IP-A, port x, @IP-ServeurHTTP, port 80 > et < @IP-A, port y, @IP-ServerFTP, port 21 >.

Question 6. Dans l'ensemble, la question a été largement traitée. Toutefois, très peu de copies ont donné le principe de fonctionnement avant de chercher le lien de sortie des différents paquets.

La seconde partie de la question concerne la variabilité de la latence. Il est important d'avoir à l'esprit que le protocole IP fonctionne en mode non connecté. Il s'agit donc d'une commutation de paquets avec la possibilité que les paquets empruntent des chemins différents (particulièrement lors d'un routage dynamique). Ensuite, même lorsque les paquets suivent le même chemin, une charge variable de trafic dans le réseau impacte le temps d'attente dans les équipements, ce qui fait varier aussi la latence.

Question 7. Cette question a pour but de tester les connaissances sur le routage utilisé dans RIP *(Routing Information Protocol)* qui s'appuie sur l'algorithme de détermination des routes décentralisé « Bellman-Ford ». De nombreuses copies expliquent très naïvement le comportement du réseau en cas de perte d'un routeur ou une rupture d'un lien et la différence entre les deux situations. Il aurait fallu donner l'information (vecteur de distance) transmise initialement par le routeur qui a détecté la panne, décrire la propagation et la prise en charge de cette information par les routeurs voisins. Ensuite, il aurait fallu présenter l'incidence sur les tables de routage et leur état final après convergence. Enfin, le jury a été très surpris de lire que la rupture d'un lien n'a aucune incidence sur les tables de routage.

Question 8. Le jury a voulu tester les connaissances des candidats sur les performances du réseau Ethernet. Il est à noter que la question sur le temps de transmission d'une trame a été bien traitée. En revanche, la question 8b sur la période de vulnérabilité a été mal traitée. Cette question est en lien avec la période de vulnérabilité et la fenêtre de collision. De manière générale, c'est la période qui représente la durée pendant laquelle une station éloignée peut détecter le canal libre et transmettre à son tour ; elle est au maximum égale au temps de propagation nécessaire entre les deux stations les plus éloignées sur le support. Suivant l'énoncé, cette période est au plus la durée de transmission de la trame puisque tout autre transmission faite jusqu'à la fin de transmission de la première aurait pu provoquer une collision.

Les réponses à la question 8c sont parfois hasardeuses. Le sujet demande de décrire la différence entre un débit support, un débit utile et un débit réel en s'appuyant sur tous les facteurs influençant les variations du débit. Le premier est la quantité d'information transmise ou reçue par unité de temps alors que le second ne

considère que la charge utile d'une trame (données utiles sans les entêtes). Le débit réel quant à lui, constaté au niveau des couches supérieures, est encore plus faible à cause des temps de traitement de protocoles, de la charge dans le réseau, du temps d'attente dans les routeurs et du temps serveur (connexions).

2.2 Épreuve 2

Données statistiques

— 256 présents
— Meilleure note : 19,06/20
— Moyenne : 6,11 ; écart-type : 5,02.

$\geq$ 1,70	75%
$\geq$ 5,00	50%
$\geq$ 9,00	25%
$\geq$ 10,00	22%
$\geq$ 15,00	7%

Le thème de cette épreuve est la construction d'une bibliothèque de grands entiers dans le langage Python, en alternative aux entiers natifs de Python. Le principe est tiré d'un article de Jean Vuillemin intitulé *Shared Integer Dichotomy*[3]. L'écriture en base 2 d'un entier est récursivement décomposée de façon dichotomique, sous forme de triplets (une partie haute, une partie basse et une dimension). Ces triplets sont construits de façon unique en mémoire, grâce à une technique de *hash consing*. Dès lors, un entier se trouve représenté par un graphe orienté acyclique. Les algorithmes sur ces entiers exploitent ce partage en mettant en œuvre de la mémoïsation, d'une façon très semblable à l'algorithmique des arbres de décision binaire. Pour cette raison, ces entiers sont baptisés IDD (pour *Integer Dichotomy Diagrams*).

La partie I a pour objectif de décrire la représentation des IDD et de familiariser les candidats avec cette idée. La partie II étudie sa mise en œuvre en Python. Le code de construction des IDD, et notamment la mise en place de la technique de *hash consing*, est intégralement donné, avec des questions pour en expliquer le fonctionnement. La partie III a pour sujet la conversion entre les IDD et deux formats différents : les entiers de Python d'une part et un fichier texte de sérialisation d'autre part. La partie IV traite d'opérations arithmétiques élémentaires sur les IDD et la partie V d'opérations logiques sur la représentation en base 2 sous-jacente. Enfin, la partie VI propose aux candidats d'explorer deux extensions possibles du format des IDD décrit dans le sujet.

Cette épreuve permet d'évaluer les connaissances des candidats sur un large spectre d'algorithmique et de programmation : tables de hachage, mémoïsation, parcours de graphes, terminaison, correction, complexité. Certaines questions demandent de proposer un code Python répondant à une spécification et une complexité données. D'autres questions, inversement, fournissent un code Python et en demandent une analyse de correction ou de complexité.

Les attentes sur le versant programmation vont au-delà de la simple écriture de programmes : la compréhension de la technique de *hash consing*, en particulier, nécessite parfois une compréhension précise du fonctionnement de certaines structures offertes par le langage (dictionnaires, en particulier). Ces attentes sont récurrentes : un professeur doit, au-delà de la pratique du langage, être capable de savoir ce qui se passe quand on « soulève le couvercle » des primitives offertes par le langage, en particulier en Python (fonctionnement, complexité, choses à faire, choses à éviter). C'est indispensable à la fois pour permettre à tous ses élèves d'éviter les écueils, mais aussi pour être capable de transmettre des explications aux plus curieux d'entre eux.

Pour obtenir la note maximale, il était nécessaire de traiter l'intégralité du sujet. Peu de copies ont traité toutes les questions, et aucune copie n'a traité toutes les questions correctement. Pour autant, il y a eu d'excellentes copies, montrant une excellente maîtrise de l'algorithmique et de la programmation. À l'opposé, beaucoup de copies ont montré de grosses lacunes, que ce soit en matière de raisonnement ou de programmation.

3. `https://hal.archives-ouvertes.fr/hal-01239120/document`

2.2.1 Remarques générales

Les réponses aux questions demandant de prouver la correction de fonctions ont été globalement faibles. Rappelons quelques faits à ce propos :

— Quand on demande de prouver la correction d'une fonction, on ne demande pas de se contenter de vérifier sur un ni même plusieurs exemples que la fonction donne le bon résultat.

— Le raisonnement par récurrence devrait être un réflexe dès lors que l'on cherche à prouver une propriété sur une fonction récursive. Il doit être indiqué comme tel, et formalisé.

— Le recours systématique à la preuve par induction, ici bien adaptée à la structure, est pertinent et correspond en général à des copies dont le discours est clair et la lecture agréable.

Cela semble une évidence, mais il est conseillé aux candidats qui expliquent dans leur copie que le code du sujet ne respecte pas les bonnes pratiques de programmation, ou que le concepteur ne comprend rien à la notion de portée des variables, d'argumenter ces points de manière très solide. Certaines copies veulent expliquer aux correcteurs des enjeux éloignés du sujet : elles perdent leur temps et font perdre leur temps aux correcteurs.

Beaucoup de candidats ont une vision très naïve de la complexité. D'une part, les constructions « avancées » de Python ne sont pas toutes de coût unitaire (recherche dans une liste, par exemple) ; d'autre part, tout appel de fonction cache, potentiellement, une complexité intrinsèque à cette fonction.

Beaucoup de copies confondent, dans leurs réponses aux questions de programmation, le type `int` des entiers natifs de Python et le type `IDD` des entiers construits dans ce sujet. L'énoncé fait pourtant l'effort de donner systématiquement les types d'entrée et de sortie de chaque fonction fournie ou demandée.

2.2.2 Remarques par question

Question 1. Il s'agit là d'une question très facile destinée à familiariser les candidats avec la notion d'IDD. Quelques copies échouent tout de même à décomposer 773 comme $3 \times 256 + 5$ ou à écrire correctement 256 sous la forme 2^{2^p}.

Question 2. Grouper les bits par octets est une bonne idée et montre le soucis de clarté et d'explicitation qui est une qualité pour un futur enseignant.

Question 3. Trop de copies éludent la question du plus grand élément, qui constitue pourtant la partie centrale de la question.

Questions 4 et 5. Il s'agit là de questions visant à vérifier la bonne compréhension de la notion de tables de hachage d'une part et de leur mise en œuvre en Python d'autre part. Les réponses ont été globalement très décevantes, avec un discours généralement peu structuré et peu convaincant. En particulier, pour argumenter que la recherche et l'insertion dans une table de hachage peuvent être considérées en temps constant, il faut prendre soin de vérifier en premier lieu que les fonctions `__hash__` et `__eq__` le sont bien, ce que beaucoup de copies ne font pas.

Pour la question 4, il semble naïf d'affirmer qu'une fonction est en temps constant car elle ne fait intervenir ni boucle ni récursivité. Il est même inquiétant de lire des choses comme « toute opération sur les listes Python s'effectue en temps constant » sur des copies qui par ailleurs peuvent être de bonne qualité.

Pour la question 5, peu de copies invoquent l'unicité en mémoire d'un même IDD, pourtant cruciale ici.

Question 6. Certaines copies, apparemment obsédées par la programmation orientée objet, s'obstinent à insérer le code des fonctions qu'elles écrivent dans la classe IDD décrite en introduction du sujet. Ce n'est ni nécessaire, ni attendu. On a vu aussi des copies créer une instance de classe factice pour appeler les méthodes de la classe IDD.

Le cas de base de la fonction `big` est souvent incorrectement traité, avec `zero` plutôt que `one`, ou encore `IDD.create(zero,zero,zero)`. Des copies font trois fois le même appel récursif, ce qui empêche grossièrement la complexité d'être linéaire en n.

Question 7. La complexité d'un parcours de graphe dépend aussi du nombre d'arcs, et pas uniquement du nombre de sommets : il faut invoquer ici le fait que les sommets ont un degré sortant borné. Le choix d'une liste pour repérer les sommets déjà visités encourt un surcoût d'un facteur linéaire : contrairement à ce que peut laisser penser la syntaxe Python, la recherche dans une liste (`x in l`) a un coût linéaire, et non constant !

Les correcteurs ont rencontré `def size(i:IDD, visite = set())` : il n'est pas possible de transformer le prototype d'une fonction demandée, même en donnant un argument par défaut — sans parler, ici, du partage catastrophique d'une valeur par défaut mutable.

Question 8. La notion d'invariant est très inégalement connue des candidats. En particulier, certains candidats confondent la notion d'invariant et la notion de variable non mutable. On apprécie les bonnes copies qui font l'effort de distinguer la fonction mathématique et la fonction programmée par le fragment de code.

Question 9. Quasiment toutes les copies qui traitent cette question oublient le coût de l'addition. Le premier paragraphe des préliminaires rappelle pourtant que les entiers Python sont de précision arbitraire et que l'addition de deux entiers m et n est en $O(\max(\log n, \log m))$.

Question 10. Les correcteurs ont rencontré trop souvent un exemple de calcul plutôt qu'une preuve. La plupart des démonstrations se contentent d'expliquer la formule sans pour autant penser à faire une récurrence ou une induction. Certaines copies ne jugent pas nécessaire d'expliquer ce que fait l'opérateur `<<` de Python.

Question 11. C'est là une question subtile, demandant de comprendre que la conversion vers le type `int` échoue faute de mémoire avant d'en arriver à faire déborder la pile. Seule une poignée de copies l'analysent correctement. D'une façon surprenante, ce point est mieux traité dans des mauvaises copies que dans des copies moyennes.

Question 12. On a vu beaucoup de confusions entre 2^p et 2^{2^p}. Un nombre non anecdotique de copies confondent les opérateurs `%` et `/` de Python. Le calcul de p à l'aide d'une boucle de condition d'arrêt $2^{2^p} < n$ avec calcul de la double puissance à chaque tour est assez regrettable, même si aucune complexité n'est attendue ici. Les candidats devraient avoir dans leurs automatismes le fait qu'un calcul de puissance se fait très bien de proche en proche au sein d'une boucle.

Question 13. Des copies donnent des exemples de fichier qui ne respectent pas la description donnée dans l'énoncé, avec des numéros faisant référence à des lignes plus loin dans le fichier. D'autres copies semblent penser que la question ne porte que sur l'exemple qui précède et non pas sur n'importe quel IDD.

Question 14. Les correcteurs ont croisé une réponse sous forme de logigramme. Pourquoi pas ! Néanmoins, il manque des précisions sur la gestion des appels empilés pour que la réponse soit exploitable. L'utilisation sans explication d'un sigle anglophone comme « DAG » ne semble pas judicieuse. Une minorité de copies proposent une méthode d'indexation des lignes correcte. Beaucoup de copies cherchent à indexer la ligne d'un IDD par l'entier qu'il représente, ce qui ne fait pas de sens vu la taille potentielle de cet entier.

Question 15. Il est nécessaire d'utiliser un dictionnaire ou une liste Python. Dans le second cas, la clé doit être un entier et non pas un caractère. Bravo aux copies qui ont vu ce point.

Question 16. Il est absurde de convertir les deux IDD vers le type `int` pour les comparer. Le but des IDD est justement de représenter des nombres que le type `int` ne permet pas de représenter !

Question 17. On a vu des réponses absurdes comme « la fonction `xp` ne comporte pas de boucle ni d'appels récursifs donc elle termine ».

On ne peut pas montrer la terminaison de deux fonctions mutuellement récursives f et g séparément (plusieurs copies supposent que f se termine pour montrer que g se termine, puis supposent que g se termine pour montrer que f se termine).

Inutile de faire un commentaire général semi-précis sur des quantités qui décroissent. La preuve passe-partout ne fonctionne pas ici ! Au contraire, on a vu des preuves précises s'appuyant sur le fait que $(\mathbb{N}^2, lex)$ est bien fondé. Le jury apprécie l'effort.

Question 18. Beaucoup de copies oublient le cas `one`.

Question 19. Un nombre non négligeable de copies ne pensent pas à montrer que les hypothèses impliquent $j \geq p$ et en déduisent une erreur (inexistante) dans l'énoncé.

Question 20. Il s'agit là d'une question très facile, bien traitée dans la majorité des copies.

Question 21. Trop de copies oublient de faire un cas particulier pour 0 et se retrouvent à appeler `rmsb` sur 0.

Question 22. On a vu de nombreuses solutions à cette question dans les copies, dont certaines très élégantes. Beaucoup de copies ont l'idée de se servir de `rmsb` pour itérer sur les chiffres 1 de l'écriture de `n` en base 2. C'est en effet une approche possible, mais il faut alors prendre soin de bien afficher les zéros de poids faible sans appeler `rmsb` sur 0.

Question 23. Il est relativement facile de s'inspirer des équations données pour la conjonction pour en dériver les équations demandées. Mais il faut tout de même réfléchir un peu pour ne pas reprendre naïvement la forme de l'équation (14). Beaucoup trop de copies ont fait cette erreur.

Question 24. Plusieurs copies proposent une fonction `log_and` en suivant les équations de l'énoncé, mais ce n'est pas ce qui est demandé ici. On attend ici un exemple explicite de suites qui mettent en évidence le nombre d'appel exponentiel et pas un commentaire disant que, lorsqu'il y a deux appels récursifs (ce qui n'est pas garantit systématiquement), on obtient alors une complexité exponentielle. Peu de copies ont compris l'enjeu de cette question et parmi ces dernières peu sont capables de donner un exemple concluant (garantissant que les deux suites ont des termes deux à deux distincts).

Question 25. La justification de la complexité est souvent oubliée.

Question 26. De trop nombreuses copies ne savent pas ce qu'est une différence ensembliste et la confonde avec une différence symétrique.

Question 27. De nombreuses copies confondent n et l'ensemble $\{n\}$.

Question 28. Une question facile (en se servant de `log_and`), plutôt bien traitée.

Question 29. Là encore une question plutôt facile, mais il faut prendre soin de justifier que chaque objet Python de la classe IDD occupe un espace borné (essentiellement trois pointeurs).

Question 30. S'il est facile de donner des majorants pour la représentation par un IDD et par une liste Python, très peu de copies ont fourni une comparaison précise montrant que chacune des deux options peut être asymptotiquement meilleure que l'autre.

Question 31. Beaucoup de réponses très informelles qui traduisent tout de même que l'enjeu est compris.

Question 32. S'il est facile de répondre au point 1 en utilisant la question précédente, quasiment aucune copie n'a su faire correctement le calcul de l'espace occupé par les listes d'entiers. Le bagage mathématique d'un candidat doit lui permettre de faire au minimum des calculs de complexité simples impliquant des sommes et des coefficients binomiaux.

Question 33. Les copies qui traitent cette question ont généralement compris l'idée générale, mais la description de sa mise en œuvre manque beaucoup de précision.

Question 34. Certains candidats ne comprennent pas le terme « entier relatif ». Par ailleurs, il est important de ne pas signer chaque sommet de l'IDD, mais d'ajouter un signe global. Le jury attend une réflexion sur le traitement de l'entier 0.

2.2.3 Quelques perles que le jury souhaiterait ne plus lire

Un petit florilège, qui n'est pas là pour jeter l'opprobre mais pour illustrer le désarroi qui a pu être celui du jury par moments :

- « La fonction n'est pas récursive donc elle est en temps constant. » et sa variante « La fonction n'est pas récursive et ne contient pas de boucle conditionnelle, donc elle s'exécute en temps constant. ».
- « La fonction `IDD.create` s'exécute en temps constant car elle n'est pas récursive. »
- « La fonction se termine bien et est donc correcte. »
- (Q10) « Prenons comme invariant "la fonction retourne un `int`" [...] Dans tous les cas, la fonction retourne un `int`, elle est donc correcte. »
- « La complexité de ces deux méthodes est en $O(n)$, donc elles s'exécutent en temps constant. »
- « Une addition en binaire se fait avec l'opération logique OU. », suivi d'une illustration avec $10 + 2$.

2.3 Épreuve 3

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

Données statistiques

Étude de cas informatique

- 150 présents
- Meilleure note : 19,12/20
- Moyenne : 6,14, écart-type : 3,48

$\geq$ 4,25	75%
$\geq$ 5,00	67%
$\geq$ 6,10	50%
$\geq$ 8,20	25%
$\geq$ 10,00	15%
$\geq$ 15,00	3%

Fondements de l'informatique

- 104 présents
- Meilleure note : 19,24/20
- Moyenne : 9,32, écart-type : 5,35.

$\geq$ 4,75	75%
$\geq$ 5,00	72%
$\geq$ 8,00	50%
$\geq$ 10,00	48%
$\geq$ 14,40	25%
$\geq$ 15,00	20%

Les deux épreuves ont été choisies de façon équilibrée avec 59% pour étude de cas informatique et 41% pour fondements de l'informatique. Nous avons eu d'excellentes copies dans les deux épreuves, avec des notes maximales comparables. Par contre, il semble que le vivier des deux épreuves ne soit pas identique et elles ont donné lieu à des moyennes assez différentes. Cela se reflète dans les autres épreuves : les candidats ayant choisi étude de cas informatique ont eu une moyenne inférieure de plus de 2 points sur l'épreuve 1 et de plus de 5 points sur l'épreuve 2 par rapport aux candidats ayant choisi fondements de l'informatique.

2.3.1 Étude de cas informatique

Présentation du sujet

Le sujet exploite le problème échiquéen des huit dames dont l'objectif est de placer huit dames sur un échiquier et de faire en sorte qu'aucune dame ne soit en prise. Les huit dames doivent donc être sur des lignes, colonnes et diagonales différentes. Le problème est connu pour être très complexe avec un nombre fini de solutions. Ce problème connaît de nombreuses variations avec différentes pièces et différentes tailles pour l'échiquier.

La première partie est la partie principale de l'épreuve. Elle propose de mettre en oeuvre une conception orientée objet. Il s'agit alors de définir les classes pour les concepts nécessaires à la réalisation du sujet (échiquier, pièce...).

La deuxième partie porte sur le développement web. Il s'agit de proposer une interface web au sujet en développant les pages web et en y associant les traitements nécessaires à la gestion des interactions.

La troisième partie porte sur les bases de données. Dans cette partie, on considère un fichier contenant plusieurs informations sur des problèmes échiquéens. L'objectif consiste à proposer une base de données pour structurer ces informations et permettre leurs traitements.

La quatrième partie porte sur l'architecture. L'objectif est de déployer une application qui permet de résoudre des problèmes échiquéens (du type problème des huit dames). Les questions posées visent à mesurer la capacité de proposer des architectures qui passent à l'échelle.

Analyse des résultats et commentaires généraux

Sur la première partie, le niveau en programmation est très hétérogène. Certaines copies sont très propres et vont directement à l'essentiel. D'autres copies manquent de précision et présentent des solutions très compliquées. Concernant la maîtrise des langages, quelques copies ont du code Python qui n'a rien de Python. On sent alors le plus souvent une influence de Java. De plus, les questions de conception sont généralement mal traitées. La relation objet / classe est trop souvent mal connue. La réutilisation de code n'est que très peu envisagée. Les copies traitent les questions les unes après les autres sans faire montre de vision d'ensemble.

Peu de copies traitent les questions relatives à la programmation web (deuxième partie). Les copies montrent des faiblesses importantes sur HTML, CSS et JavaScript. Il y a cependant quelques très bonnes copies sur cette partie.

Plusieurs copies traitent les questions relatives aux bases de données (troisième partie). Elles sont dans l'ensemble d'un très bon niveau.

Peu de copies traitent des questions relatives à l'architecture. Dans l'ensemble, ces copies manquent de précision et se limitent à des descriptions abstraites. Il y a cependant quelques très bonnes copies sur cette partie.

Commentaires par question

Dans cette section, nous présentons des commentaires question par question pour la première partie.

Question 1 Il faut présenter 3 tests et bien préciser pour chacun l'état initial, la suite d'instructions avec les données et le résultat attendu. Certaines copies ont listé les erreurs du code (parfois sans écrire de test), ce qui n'est pas demandé. Il y a même quelques copies qui ont proposé des corrections ; là encore, ce n'est pas demandé.

Question 2 Cette question a été bien traitée dans l'ensemble, même si certaines copies ont proposé des solutions très complexes. Les bonnes copies proposent des mécanismes pour réutiliser le code nécessaire à la gestion des cases.

Question 4 Il faut s'appuyer sur le code des questions précédentes pour obtenir un code simple. Certaines copies proposent du code très compliqué.

Question 5 Cette question cherche à discuter des avantages / inconvénients de deux conceptions : mettre la position des pièces dans l'échiquier ou dans les pièces. Cette question est relative à l'affectation des responsabilités (patron GRASP).

Question 6 Il s'agit ici de mettre en œuvre un mécanisme pour réutiliser du code (héritage ou autre). De nombreuses copies présentent des solutions fausses ou incomplètes. Elles montrent que ces notions de réutilisation de code sont mal maîtrisées.

Question 8 L'objectif est de programmer une vérification simple. Plusieurs copies proposent pourtant des solutions très compliquées.

Question 9 Cette question nécessite de mettre en place un patron fabrique (*factory method*). Il n'est pour autant pas nécessaire de connaître ce patron. Il s'agit juste de proposer un constructeur générique (avec plusieurs paramètres). Peu de copies ont bien traité cette question.

Question 10 Plusieurs copies ont bien traité cette question.

Question 11 Cette question peut être traitée avec une réponse simple mais qui va prendre beaucoup de temps à l'exécution. Pour autant, plusieurs copies ont tenté de proposer une réponse complexe pour parcourir l'ensemble des possibilités (trop complexe parfois).

Question 12 La principale difficulté est d'utiliser une structure de données pour pouvoir gérer et raisonner sur les matrices de permutations. Peu de copie ont traité cette question.

Question 13 L'objectif est ici d'utiliser une méthode pour tester les autres (principe utilisé en test logiciel).

Question 14 Il s'agit ici de proposer une conception permettant à un développeur de passer sa propre méthode de résolution. Cela nécessite que cette méthode soit manipulée (passage de la méthode comme paramètre ou exploitation des classes abstraites).

Question 15 Il faut proposer un mécanisme permettant d'évaluer dynamiquement des solutions. Peu de copies ont proposé des bonnes solutions.

Question 17 Répondre à cette question nécessite de mettre en place plusieurs vérifications. Il faut les identifier et les coder.

Question 18 Cette question nécessite la mise en place de la classe Game qui sert de point d'entrée. C'est cette classe qui centralise toutes les informations.

Question 19 Peu de copies ont présenté des séquences. Cela illustre que le fonctionnement des objets (avec échange de messages) est mal maîtrisé.

Question 20 Cette question est similaire à la question 17. Elle nécessite la mise en place de plusieurs vérifications.

Question 21 Cette question nécessite le support d'une suite de coups et la vérification que le joueur suit bien ces coups. Peu de copies ont traité cette question.

Question 22 Il faut présenter quelques éléments sur la complexité des approches (en temps et en espace).

Question 23 La généralisation nécessite plusieurs changements dans le code (pour que le paramètre N soit utilisé). Il faut ici lister tous les impacts dans le code et sur les tests.

Dans cette section, nous présentons des commentaires question par question pour la deuxième partie.

Question 24 Il faut proposer un formulaire web (balise FORM) et bien préciser l'attribut action.

Question 25 Il faut ici utiliser JavaScript pour générer le code HTML (en une boucle for). Peu de copies ont traité cette question.

Question 26 Il faut ici ajouter du code JavaScript sur le bouton et faire en sorte qu'une image soit ajouté sur la case. Peu de copies ont traité cette question.

Question 27 Cette question est très similaire à la question 26. On doit alors réutiliser du code. Peu de copies ont proposé la réutilisation de code.

Question 28 Cette question nécessite d'envoyer une requête via du code JavaScript. Peu de copies ont proposé une solution.

Question 29 Il s'agit ici de préciser que les contrôles faits dans le navigateur aident à la fluidité.

Dans cette section, nous présentons des commentaires question par question pour la troisième partie.

Questions 31-36 Ces questions traitent de la conception des bases de données. Les bonnes copies ont proposé un schéma et ont détaillé les réponses.

Questions 37-42 Ces questions demandent des requêtes SQL (avec pour les dernières questions des liens vers Python). Les copies sont d'un bon niveau.

Dans cette section, nous présentons des commentaires question par question pour la quatrième partie.

Question 43 Il s'agit ici de proposer un moyen pour répartir les requêtes entre plusieurs serveurs physiques. Un tourniquet peut être proposé mais il faut expliquer comment le lien est gardé avec le client.

Question 44 Il s'agit ici de proposer un mécanisme avec des serveurs virtuels. Il faut alors expliquer les règles permettant la gestion du cycle de vie des serveurs virtuels (création, suppression).

Question 44 Il s'agit ici de proposer un mécanisme pour demander à plusieurs serveurs d'effectuer un contrôle mais d'arrêter le traitement en cas de réception d'une bonne réponse.

Question 44 Il s'agit ici de proposer un mécanisme en parallèle et de discuter d'un compromis entre efficacité et coût.

2.3.2 Fondements de l'informatique

Présentation du sujet

Le sujet étudie la compilation des expressions arithmétiques sous le prisme des besoins en espace et en temps. La première partie s'intéresse au cas de la compilation vers une machine à pile ; on fait le lien entre le schéma de compilation d'Erchov et le nombre de Strahler d'une expression. La deuxième partie propose l'utilisation d'une machine à registres et démontre qu'il n'y a pas de gain d'espace par rapport à la machine à pile, même dans une compilation optimale. Enfin, la troisième et dernière partie étudie le partage de sous-expressions lors de la compilation et fait le lien avec le jeu du marquage (*peeble game*) dans un graphe orienté sans cycle.

Analyse des résultats et commentaires généraux

Le jury a pu constater qu'un grand nombre de candidats ont montré une capacité à aborder les questions posées de manière pertinente, à avancer dans le problème, et à produire des réponses satisfaisantes à des questions balayant algorithmique, complexité, sémantique... La moitié des candidats a généralement avancé assez loin dans le problème, abordant largement 3.1 et regardant en partie 3.2. Les toutes meilleures copies, réellement excellentes, ont traité quasiment tout le problème à l'exception des trois dernières questions qui n'ont pas été sérieusement abordées. Traiter intégralement et parfaitement les parties 1 et 2, finalement assez élémentaires, permettait d'obtenir un peu plus de 10/20.

On retrouve enfin dans les copies plus faibles deux traits récurrents : familiarité limitée, voire nulle, avec OCaml (voir aussi *infra*) ; et absence de maîtrise (ou au moins de précision) du raisonnement par récurrence (ici souvent par induction). C'est pourtant une technique de raisonnement indispensable dans toutes les questions d'informatique traitant de structures de données ou d'algorithmes récursifs, et en acquérir la maîtrise devrait être un passage obligé de la préparation de tout agrégatif.

Si la syntaxe OCaml est maîtrisée par certains candidats, d'autres ont choisi de sauter systématiquement les questions de programmation, d'écrire du code avec une syntaxe approximative s'inspirant de Python, et quelques rares se sont contentés de décrire les algorithmes en pseudo-code. Le jury rappelle que la maîtrise des trois langages de programmation C, OCaml et Python est requise pour le concours de l'agrégation. Si le jury sait se montrer clément pour des erreurs mineures de syntaxe, inévitables lorsqu'on écrit du code sur papier, il a sanctionné les copies où ces erreurs sont trop nombreuses ou se répètent. De manière non exhaustive, il est rappelé qu'en OCaml :

- les listes sont des objets non modifiables ; à ce titre, `x :: lst` ne modifie pas la liste `lst`, mais crée une nouvelle liste ayant `x` comme tête et `lst` comme queue. De même, il n'existe pas de méthode `append` ou `pop` permettant d'enlever ou de rajouter un élément à une liste ;
- les listes ne sont pas des tableaux et ne supportent pas de syntaxe permettant d'accéder facilement à un élément d'indice donné. Si `lst` est une liste, il n'existe pas de syntaxe `lst.(i)`, `lst[i]` ou autre permettant de récupérer l'élément d'indice i ;
- si le parenthésage est parfois facultatif, il est également souvent nécessaire. Les règles de priorité des différentes opérations dans une ligne de code sont mal connues. Les parenthèses autour des arguments d'une fonction manquent souvent, même dans les meilleures copies. Par exemple, `calcul j+1` ne fera pas le même calcul que `calcul (j+1)`. De même, `let x = ref -1 in` déclenchera systématiquement une erreur lors de l'analyse syntaxique.

Par ailleurs, une fonction dont l'écriture est uniquement un enchaînement de `List.fold_left`, `List.iter` et `List.map`, ou contenant plusieurs fonctions auxiliaires est généralement difficile à comprendre. Il est rappelé que l'agrégation est un concours visant à recruter des enseignants et qu'un code complexe doit être expliqué convenablement — il n'est au demeurant, même si le code est simple, jamais nuisible d'expliquer de manière claire en quelques mots comment il fonctionne ; c'est au contraire une préoccupation pédagogique qui est appréciable dans le contexte de ce concours.

Certaines questions de programmation sont accompagnées d'un calcul de complexité à effectuer. Les justifications données sont parfois incomplètes. Par exemple, pour justifier d'une complexité linéaire en la taille d'un arbre, affirmer « la fonction ne traite chaque nœud de l'arbre qu'une seule fois » est insuffisant si on ne précise pas que ce traitement ne fait que des opérations en temps constant.

Les questions théoriques, qui représentent la majorité du sujet, et en particulier les preuves par induction ou récurrence, ont parfois été bâclées. À ce sujet, les futurs candidats doivent retenir que :

- l'initialisation d'une induction ou récurrence ne doit pas être négligée. Les candidats qui se sont contentés de « c'est évident », « il n'y a rien à faire », « trivial » ou autre commentaire du même type ont été systématiquement sanctionnés. Même si ce n'est pas nécessairement la partie difficile d'une telle preuve, on attend une justification rigoureuse. Par ailleurs, l'initialisation doit être correctement identifiée : initialiser pour les arbres de hauteur 1 au lieu des arbres de hauteur 0 sera sanctionné ;
- une preuve par induction ou récurrence doit être correctement rédigée : le candidat doit expliquer clairement quel type de preuve il est en train d'effectuer, quelle propriété fait l'objet de la récurrence, préciser clairement où est traitée l'initialisation et où est traitée l'hérédité ; des preuves ont parfois

été limitées à des explications informelles à base de « ça marche pour 1, ça marche pour 2, et on voit bien qu'on peut étendre le raisonnement à n quelconque ». Ces explications n'obtiennent généralement qu'une faible partie des points attribués à la question.

Par ailleurs, les candidats sont encouragés à utiliser au maximum les notations de l'énoncé et à rédiger leurs réponses en articulant les différents éléments avec des phrases. Les preuves qui introduisent tout un système de notations formelles et qui sont constituées d'un enchaînement de formules sans phrases sont généralement très difficiles à suivre. Le jury ne décourage pas les candidats d'introduire des notations lorsque c'est nécessaire, mais cela doit se faire avec parcimonie.

Enfin, certains candidats ont choisi de gagner des points en allant picorer les questions les plus faciles tout au long du sujet. Cette pratique est généralement peu rémunératrice car les questions simples de milieu de sujet n'ont pour objectif que de faire comprendre les notions abordées aux candidats mais n'ont qu'un poids faible dans le barème de l'épreuve.

Commentaires par question

Dans cette section, nous présentons des commentaires question par question.

Question 2 Il faut bien penser ici à parcourir la liste d'instructions et à ne pas écrire du code ne traitant qu'une seule instruction.

Question 3 Dans plusieurs copies, le calcul de la consommation courante de l'espace est fait correctement, mais la valeur maximale n'est pas gardée en mémoire.

Question 5 Il s'agit de la première question où il est attendue une preuve par induction correctement rédigée.

Question 7 À nouveau, une preuve rigoureuse par récurrence est attendue. On peut se permettre de passer rapidement sur certains détails pour D_n une fois la preuve pour G_n faite correctement.

Question 11 On attend ici la distinction entre le cas $n = 0$ et $n \geqslant 1$.

Question 13 Une justification se contentant de « la taille maximale est atteinte pour un arbre binaire complet » est insuffisante. On attend une preuve rigoureuse par récurrence ou induction.

Question 14 Il faut bien penser à expliciter de quelle(s) question(s) on réutilise les résultats.

Question 15 De nombreuses copies ne donnent pas de méthode pour construire une telle expression et se limite à « on prend un arbre complet de hauteur 5 ».

Question 17 Il faut penser à garder en mémoire les résultats des appels récursifs pour éviter une complexité exponentielle.

Question 18 On n'attend pas ici d'optimisation particulière de la complexité temporelle. Une fonction correcte de complexité quadratique obtient généralement tous les points.

Question 19 Les copies qui indiquent un exemple explicite dans lequel la pire complexité est atteinte ont été valorisées. Il faut bien penser à citer tous les calculs qui augmentent potentiellement la complexité (concaténation, appels à `strahler`…).

Question 20 Même si la réécriture du code n'est pas demandée, le jury attend des explications précises pour l'implémentation de l'algorithme amélioré. On ne peut pas se limiter ici à dire « on utilise de la mémoïsation ».

Question 23 Un contre-exemple est attendu pour toute valeur de K, et pas pour une valeur de K particulière.

Question 25 La principale difficulté de cette question est la compréhension du formalisme des définitions précédentes.

Question 27 Cette question est plus difficile qu'elle n'en donne l'air. La principale difficulté est de justifier que $k+1$ registres sont nécessaires, même si les calculs des deux sous-expressions sont entremêlés.

Question 31 Une lecture attentive de l'énoncé est requise pour savoir ce qui est demandé. La précondition pour que `verif_aretes s lst` renvoie `true` ne doit pas supposer que `lst` est la liste d'adjacence de `s`.

Question 33 Il est inutile de déterminer le puits du graphe pour cette question. En revanche, mémoïser les résultats (dans un tableau par exemple) et utiliser le graphe transposé sont nécessaires pour obtenir la bonne complexité.

Question 34 Pour cette question, il faut calculer le puits. Même si aucune complexité n'est imposée, les codes particulièrement inefficaces ont été sanctionnés. Travailler avec des listes pour cette fonction n'est généralement pas une bonne idée.

Question 37 L'utilisation de la règle (iii) ne donne pas une stratégie adéquate. S'il faut se contenter de la règle (ii), il faut justifier que cela est toujours possible (par exemple par l'existence d'un ordre topologique).

Question 38 Beaucoup de copies ont décrit la stratégie par des schémas peu clairs sans utiliser les notations de l'énoncé. S'il est encouragé de faire des représentations graphiques, cela ne doit être qu'à titre d'illustration et ne peut pas se substituer à une preuve ou une description précise. On attend une justification pour le temps minimal, même si l'énoncé ne le demande pas explicitement.

Chapitre 3

Épreuves orales

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.

Les épreuves orales se sont déroulées du 16 au 24 juin 2022 au lycée Paul Valéry à Paris. Les candidates admissibles ont été convoquées pour les trois épreuves orales sur trois jours consécutifs. Le jury est bien conscient du stress et de la fatigue des candidates avec des temps de préparation longs et des horaires parfois décalés. Son objectif n'est pas de piéger les candidates mais de les mettre dans un contexte leur permettant de montrer le meilleur d'elles-mêmes, tout particulièrement en matière de connaissances disciplinaires et de compétences pédagogiques. **Le jury a jugé l'oral de l'agrégation d'un très solide niveau global, faisant de cette première session un concours très sélectif. Le jury a assisté à un grand nombre d'excellentes prestations, et a eu à déplorer peu d'oraux franchement ratés.**

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 2023. Les préparations à l'agrégation ont également mis à disposition des livres, également accessibles à toutes les candidates. L'environnement informatique à disposition des candidates (pour les trois épreuves) est celui téléchargeable sur le site du jury et les fichiers des candidates sont sauvegardés régulièrement sur un serveur. Il est à noter que les responsables informatiques ne sont pas là pour aider les candidates 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. Dans chacune des épreuves orales, le jury commence par laisser totalement l'initiative à la candidate et n'intervient qu'exceptionnellement (en leçon 30 minutes, en modélisation 35 minutes, en TP 30 minutes). Il attend que la candidate soit maîtresse de son organisation, notamment qu'elle ait prévu une montre ou tout autre dispositif plutôt que de demander l'heure au jury.

Le jury utilise l'ensemble du spectre entre 0 et 20 pour noter. La note 20 marque les meilleures prestations mais n'indique pas qu'une candidate est parfaite ; de même, une note très basse marque uniquement le fait que la candidate a été comparativement moins bonne que les autres. Il est aussi à rappeler que les examinatrices peuvent poser des questions sur un spectre thématique plus large mais en lien avec celui du sujet, par exemple sur l'impact de la hiérarchie mémoire sur le comportement d'un programme, même si le sujet est plus algorithmique. On vise à recruter des informaticiennes *complètes* et les questions peuvent permettre d'évaluer la capacité des candidates à faire des liens avec d'autres sous-domaines que celui étudié par le sujet.

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

1. Le chapitre 2 est écrit au masculin.

Il est compréhensible que les candidates soient en situation de stress, mais le jury attend d'elles une attitude de professeure face à sa classe. Il faut donc éviter de passer son temps à clipser/déclipser son marqueur ou de s'asseoir sur la table à côté du tableau. Quand elle se réfère à un document projeté, la candidate doit montrer le tableau, pas l'écran de l'ordinateur ; de même, il est inutile de pointer des éléments sur ses feuilles de brouillon. Les candidates peuvent se référer à leurs notes lorsqu'elles le souhaitent pendant l'ensemble des épreuves. Néanmoins, savoir prendre de la distance avec ses notes pour se concentrer sur la communication avec l'auditoire est une vraie démarche d'enseignante, et sera valorisé. Il est important de s'adresser à l'ensemble de son auditoire, en l'espèce l'ensemble du jury. Les candidates doivent avoir une connaissance suffisante des logiciels qu'elles utilisent pour être capables d'agrandir rapidement la police de caractères ou passer sur fond clair pour améliorer la lisibilité.

Le jury rappelle que la gestion du tableau est également évaluée par le jury. Un contenu propre et clair, éventuellement utilisant des couleurs, sera valorisé. Les candidates peuvent effacer le tableau ou une partie de tableau après avoir demandé l'autorisation du jury. Elles peuvent toutefois s'autoriser à effacer ce qui n'est pas du contenu (une faute d'orthographe, un trait mal tracé…).

3.1 Leçon

Données statistiques

— Présentes : 54
— Meilleure note : 20
— Moyenne : 10,93, écart-type : 5,51

$\geq$ 5,00	80%
$\geq$ 7,25	75%
$\geq$ 10,00	54%
$\geq$ 11,00	50%
$\geq$ 15,00	25%

Le jury attend des candidates qu'elles se projettent dans la fonction d'enseignante, notamment en adoptant, lors de leur oral, un ton et une posture proches de celles qu'elles adopteraient face à une classe.

Les candidates ne doivent pas transformer l'épreuve de leçon en une épreuve de TP ou de modélisation. Ainsi, bien qu'un vidéo-projecteur soit mis à leur disposition, son usage devrait être exceptionnel et bref dans cette épreuve (par exemple pour afficher un graphe avec beaucoup d'arcs qu'il serait inutilement long de dessiner pendant le développement). Évaluer l'aptitude des candidates à utiliser l'ordinateur pour illustrer des phénomènes ou des concepts (comme par exemple une animation montrant le déroulement d'un algorithme) n'est pas un objectif de cette épreuve. De même, la production de programmes informatiques est évaluée lors de l'épreuve de TP.

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

Plan

Le jury souhaite voir proposer par la candidate un plan de la leçon d'au plus trois feuillets A4 manuscrits. Un bon plan n'occupe pas nécessairement la totalité de ces trois pages, mais il est regrettable de voir des candidates n'en utiliser qu'une seule, surtout très aérée. Le plan doit être écrit de manière lisible et compréhensible, en tenant compte du fait qu'il sera photocopié. À la session 2023, les feuilles fournies comporteront un cadre en dehors duquel il faudra ne pas écrire. Il est recommandé de prendre ces feuilles en orientation paysage et d'écrire sur deux colonnes. Il convient de numéroter les pages.

Certains plans se limitent à une simple table des matières, telle qu'on pourrait la recopier depuis n'importe quel livre sur le sujet. Le jury attend un plan structuré (avec des parties et sous-parties) et détaillé, incluant des définitions, des exemples, des remarques, des illustrations..., et veillant notamment à la cohérence des notations entre ces éléments.

Ainsi, se contenter de donner le nom d'un théorème ou d'un algorithme (qu'il soit au programme de l'agrégation ou non) n'est possible que s'il s'agit d'une ouverture vers un thème voisin mais extérieur à la leçon. Par exemple, simplement citer l'algorithme de Dijkstra est peut-être possible dans la leçon *Arbres : représentations et applications*, si on choisit de parler des tas comme application des arbres et qu'on souhaite signaler que les tas ont des usages. Mais si on choisit de parler de cet algorithme dans la leçon *Implémentations et applications des piles, files et files de priorité*, il convient de le décrire, car il s'agit alors d'une application d'une structure de données, et il fait donc partie intégrante de la leçon.

Il est opportun de numéroter les éléments du plan afin de permettre au jury d'y faire référence au cours de la discussion. Une numérotation séquentielle globale est la plus pratique (c'est-à-dire : définition 1, exemple 2, définition 3, et non : définition 1, exemple 1, définition 2).

La présentation du plan ne doit pas consister en sa lecture ou sa paraphrase. Le jury apprécie que les choix pédagogiques et l'enchaînement logique du plan soient présentés oralement lors de la présentation du plan. Un plan « catalogue » (reprenant par exemple toutes les structures de données imaginables pour la leçon *Implémentations et applications des piles, files et files de priorité*) n'apparaît pas comme un choix judicieux. Il est tout à fait possible de signaler pourquoi on a choisi de ne pas inclure certains éléments dans la leçon : enseigner, c'est aussi savoir poser les limites de ce que l'on transmet à un public donné.

Il convient d'utiliser au mieux les dix minutes imparties à la présentation du plan. Il est tout à fait regrettable de voir des candidates terminer leur présentation au bout de cinq minutes.

Sans que cela soit exigible, il a été apprécié que la candidate précise le niveau auquel se place la leçon ainsi que les principales sources utilisées lors de son élaboration. Il est possible, lorsque le sujet s'y prête, de proposer un plan au niveau terminale NSI. Cependant, le jury pourra, dans la phase de questions, interroger sur la totalité du programme de l'agrégation en lien avec la leçon. Il est possible de proposer une leçon dont les différentes parties s'adressent à des niveaux variés.

Développement

Les deux développements proposés au jury doivent être introduits dans la présentation orale du plan et visibles sur le plan écrit. L'intitulé des développements doit permettre au jury de se faire une idée assez précise de leur contenu. Proposer un seul développement est nettement sanctionné. À cet effet, deux développements trop semblables comptent comme un seul et un développement manifestement hors sujet ou manifestement trivial ne compte pas. Cependant, préparer un seul développement et en proposer un deuxième pour la forme est encore plus risqué : le jury ne choisira pas systématiquement le développement le plus alléchant. Proposer trois développements ou davantage n'apporte aucun bénéfice.

Le développement doit être maîtrisé, ce qui ne signifie pas être appris par cœur et récité. Il est par exemple préférable de présenter un algorithme simple et bien maîtrisé qu'un algorithme complexe et mal compris. Il est rappelé que la candidate a le droit de consulter ses notes pendant son développement, sans demander aucune permission préalable ; s'abstenir d'utiliser ses notes n'entraîne pas, en soi, de bonification par le jury ; les consulter n'entraîne pas, en soi, de malus. Il faut faire de ses notes un usage raisonné : passer son temps à recopier sa feuille au tableau est clairement à ne pas faire, mais ne pas disposer de notes du tout ou ne pas s'y référer alors qu'on hésite visiblement sur un détail d'une formule compliquée n'est pas pertinent non plus. Reste la question des points qui sont explicitement au programme de l'agrégation : ils devraient être connus, mais à défaut, le jury préfère qu'une candidate s'aide de ses notes avec fluidité que de la voir faire un effort de mémoire pendant de longues secondes.

Le développement doit avoir un réel rapport avec la leçon et être convenablement inséré dans le plan. Ainsi, un développement sur un algorithme classique de calcul d'enveloppe convexe d'un nuage planaire n'a pas vraiment sa place dans la leçon *Implémentations et applications des piles, files et files de priorité* dès lors que, s'il utilise inévitablement des structures de données, celles-ci n'y jouent pas un rôle central.

Comme pour la présentation du plan, il est attendu que les candidates mettent à profit une part substantielle du temps maximal qui leur est donné (vingt minutes). Un développement de 13 minutes gagnerait à être renforcé.

Questions

Les candidates doivent s'attendre à des questions techniques (résoudre un exercice, détailler un calcul de complexité, répondre à une question de programmation...) mais aussi à des questions didactiques (pertinence de l'ordre dans lequel sont introduites des notions, discussions sur d'autres possibilités) et de pédagogie (quelle partie de la leçon sera abordée en TD plutôt qu'en TP ? est-il possible d'aborder ce concept avec des élèves de terminale ?...).

Il est bien sûr attendu des candidates qu'elles répondent avec honnêteté aux questions : il n'est pas scandaleux de se tromper, mais il est scandaleux de tenter de tromper le jury en affirmant avec aplomb des choses fausses ou pour le moins douteuses. Lorsqu'on ne sait pas, la bonne attitude est de réfléchir avec le jury, ou, en dernier recours, de dire « je ne sais pas ». Du reste, certaines questions posées par le jury (notamment au plan pédagogique) n'admettent pas *une* bonne réponse, et c'est davantage la capacité de la candidate à réfléchir et envisager plusieurs possibilités qui est évaluée.

Remarques sur certaines leçons

La listes de leçons pour 2023 est disponible depuis `https://www.devenirenseignant.gouv.fr/` [2]. Voici quelques remarques spécifiques à certaines leçons.

- Pour les leçons d'algorithmique, le jury appréciera que les candidates signalent les points-clés de l'implémentation.
- Pour la leçon *Programmation dynamique*, le jury appréciera une discussion avec le concept de mémoïsation.
- Pour la leçon *Exemples de méthodes et outils pour la correction des programmes*, le jury appréciera que la candidate évoque la spécification, le test, la documentation, le débuggage mais aussi des outils pratiques tel que valgrind.
- Pour la leçon *Implémentations et applications des ensembles et des dictionnaires*, le jury s'étonne de ne pas avoir eu de présentation des tables de hachage.
- Pour la leçon *Décidabilité et indécidabilité. Exemples*, on s'attend à ce que la candidate ait réfléchi à des moyens d'aborder ces notions restant dans le cadre du programme NSI, donc sans les machines de Turing.

3.2 Travaux pratiques

Données statistiques

- Présentes : 54
- Meilleure note : 20
- Moyenne : 10,85, écart-type : 4,73

$\geq$ 5,00	87%
$\geq$ 7,60	75%
$\geq$ 10,00	57%
$\geq$ 10,33	50%
$\geq$ 14,50	25%
$\geq$ 15,00	13%

Cet oral, composé d'une partie « Travaux Pratiques » et d'une partie « Revue de code », est avant tout centré sur la pédagogie et n'est en aucun cas une épreuve de virtuosité ou de rapidité. Le but est ici de montrer la capacité des candidates à, d'une part, produire et justifier du code clair et pédagogique et à, d'autre part, comprendre et expliquer comment corriger un code défectueux comme celui que pourrait produire une élève.

2. `https://media.devenirenseignant.gouv.fr/file/agreg_externe/14/1/p2023_agreg_ext_informatique_2_1428141.pdf`

Comme pour les deux autres épreuves orales, le jury attend un exposé structuré et mettant en perspective les pistes de travail proposées par l'énoncé. L'objectif de l'épreuve est de s'assurer que la candidate et future professeure est capable de construire et conduire des travaux pratiques avec ses élèves.

Ainsi, la candidate doit prendre quelques instants pour présenter la problématique du sujet, mais sans excès. De rares candidates ont voulu introduire leur exposé en replaçant très longuement le sujet du TP dans son contexte scientifique général et en apportant des connaissances qui ne sont pas dans l'énoncé du sujet. Elles ont perdu du temps inutilement. Mais à l'inverse, l'exposé ne doit pas se résumer à une énumération scolaire des différentes réponses aux questions.

L'usage du tableau doit être structuré et être au service de la compréhension du discours de la candidate. La vidéo-projection doit se concentrer sur les éléments de code, tests, d'exécution ou de mise sous forme graphique de résultats d'exécution. Une présentation basée sur des diapositives n'est pas souhaitée.

Écriture du code, lisibilité, organisation

Concernant la production de code, l'épreuve est avant tout, rappelons-le, une épreuve de pédagogie et non pas un concours de vitesse de développement. Le jury souhaite voir du code agréable à lire et invite les candidates à privilégier les choix de code didactiques. Pendant qu'elles préparent l'épreuve, les candidates devraient garder à l'esprit la question suivante : mon code est-il digne de figurer dans un bon manuel d'enseignement scolaire ?

Les candidates présentant un code clair ont d'ailleurs beaucoup plus de facilité à en faire l'exposé et parviennent mieux à se concentrer sur les points clés de leur implémentation, puisque leur code se comprend de lui-même. L'utilisation excessive des bibliothèques peut par ailleurs nuire à la compréhension du code. Le jury apprécie que les candidates respectent les usages de programmation de chaque langage.

Les très bonnes candidates ont même le réflexe d'organiser leur code sous la forme de plusieurs fichiers et précisent dans leur exposé comment ceux-ci interagissent entre eux. Cela leur permet de gagner du temps dans la présentation en reléguant des fonctions annexes peu intéressantes dans des fichiers auxiliaires.

On rappelle d'ailleurs que les candidates sont encouragées, durant leur présentation initiale, à mentionner tel ou tel point technique sans le développer, laissant au jury la possibilité d'y revenir plus longuement par la suite.

En plus de l'écriture de code lisible et pédagogique, l'épreuve de TP repose sur la transmission de l'idée qu'un code est correct. Et avant de pouvoir justifier qu'un code est correct, encore faut-il qu'il s'exécute correctement. Beaucoup de candidates moyennes se contentent de coder des fonctions sans les tester et n'en montrent aucune exécution. Présenter d'énormes paquets de code sans convaincre le jury qu'il fonctionne et qu'il est correct n'a pas d'intérêt. Au contraire, le jury s'attend à des exécutions des codes produits avec des exemples pendant la présentation. Pendant la phase de questions, le jury peut demander à ce que les codes produits soient recompilés ou réexécutés, éventuellement avec des exemples proposés par le jury, ces exemples pouvant être différents de ceux proposés par la candidate.

Pour revenir à la justification de la correction, une première approche passe par la notion de preuve : sans utiliser un formalisme excessif, la mention d'invariants bien choisis peut contribuer à justifier de la correction d'un algorithme.

L'autre principale approche repose sur l'usage de tests. La qualité des tests du code produit est une partie importante de l'épreuve de TP, même si l'explication des tests ne doit pas prendre une part prépondérante dans la présentation orale. Les tests proposés doivent suivre une démarche méthodologique rigoureuse. L'utilisation d'outils simples (options de compilation, valgrind...) est fortement appréciée.

Les tests conduits par la candidate doivent figurer dans sa production. Certaines candidates signalent qu'elles ont effectué des tests non conservés par des commandes en ligne au moment d'écrire leur code. Le jury ne peut prendre en compte que ce qu'il a la possibilité d'évaluer et donc de constater pendant l'interrogation.

Enfin, certaines candidates cherchent à impressionner le jury par des optimisations de codes excessives là où elles ne sont pas utiles. Par exemple, faire de la récursivité terminale pour éviter d'empiler une trentaine de valeurs n'est pas pertinent. Le jury conseille de se concentrer sur du code simple, bien maîtrisé et dont la sûreté de fonctionnement est assurée.

Revue de code

Concernant la revue de code, le jury a apprécié qu'elle soit bien comprise comme un exercice permettant à la candidate de se projeter dans sa future fonction de professeure. En général, la plupart des erreurs ou maladresses de code sont bien repérées.

Néanmoins, trop de candidates moyennes ne fournissent qu'une correction hasardeuse qu'elles n'ont visiblement pas exécutée elles-mêmes. Même dans l'exercice de revue de code, le jury attend de voir que le code a été testé et vérifié. Le jury apprécie que la candidate en profite pour rappeler et appliquer de bonnes pratiques de programmation lorsqu'elle présente sa correction.

Comme pour les autres épreuves orales, on attend un exposé structuré, même sommairement. Ainsi, de nombreuses candidates ont divisé leur revue entre différents types de malfaçons : bug, maladresse..., ce qui a souvent été un choix heureux. D'autres traitent les erreurs de manière séquentielle. Les bonnes candidates commencent par analyser rapidement l'intention du code. Certaines candidates manquent de systématisme : elles repèrent une erreur à sa première occurrence mais ne cherchent pas à voir si elle se répète ou oublient de la corriger ensuite.

Mentionnons enfin un comportement que le jury a pu rencontrer quelques fois : après avoir survolé les erreurs du code proposé (souvent de façon très partielle), la candidate propose sa propre version du code demandé, version qui n'a que peu de rapport avec le code proposé. Cette façon de procéder va à l'encontre de l'intention de cette épreuve dont l'objectif est d'illustrer la capacité de la candidate à comprendre et aider à corriger les erreurs d'un code écrit par une élève. Notons d'ailleurs que les algorithmes proposés pour correction dans cet exercice sont considérés comme simples, et donc que voir une candidate écrire une version correcte de l'algorithme demandé plutôt que de prendre soin de comprendre et corriger le code proposé est perçu comme très maladroit.

3.3 Modélisation

Données statistiques

— Présentes : 54
— Meilleure note : 20
— Moyenne : 10,13, écart-type : 4,39

$\geq$ 5,00	81%
$\geq$ 6,50	75%
$\geq$ 10,00	56%
$\geq$ 11,00	50%
$\geq$ 13,50	25%
$\geq$ 15,00	13%

Les candidates se voient remettre un sujet unique (pas de choix), qui part généralement d'un problème concret, pour lequel il propose une ou plusieurs modélisations informatiques, et esquisse l'analyse et la résolution, éventuellement partielle, du ou des problèmes sous-jacents. Cette analyse peut aboutir à un algorithme, une étude de complexité, mais aussi à l'ébauche d'une solution technique (logicielle ou matérielle) au problème initial. Les sujets sont construits de manière à permettre une certaine progressivité, et peuvent n'être traités que partiellement. Le texte se conclut par une liste de pistes de réflexion. Au moins une de ces pistes guide la candidate pour se saisir d'une dimension éthique, sociétale, environnementale, économique ou juridique.

Présentation

L'oral proprement dit débute par la présentation de la candidate, d'une durée d'au plus 35 minutes et incluant la présentation argumentée de la ou des illustrations sur ordinateur, ainsi que la discussion d'une dimension éthique, sociétale, environnementale, économique ou juridique telles que mentionnée par l'arrêté.

Cette discussion — courte et synthétique — ne suit pas nécessairement la piste proposée par le jury, toute initiative personnelle pertinente ayant d'ailleurs vocation à être valorisée.

Il est attendu des candidates 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. 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, la candidate 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. Le choix peut être fait de se concentrer sur seulement une partie du texte, mais dans ce cas, le traitement de cette partie doit être particulièrement approfondi. Dans tous les cas, la présentation doit s'adresser à un public d'étudiantes de niveau L1/L2 ou CPGE.

De même qu'à l'épreuve de leçon, le jury est attaché à la capacité des candidates 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 la candidate se sent plus à l'aise.

Le niveau disciplinaire des candidates est globalement très satisfaisant. Toutefois, le jury regrette que certaines candidates ne maîtrisent pas certains algorithmes du programme du lycée, même lorsqu'ils sont présentés dans le texte. Les études de complexité élémentaires ne doivent pas non plus poser de problèmes aux candidates. La complexité des opérations élémentaires des structures de données courantes, telle que `len(t)` ou `t[a:b]`, doit également être maîtrisée.

Plusieurs candidates ont mal géré leur temps et ont dû interrompre leur exposé au milieu d'une explication. Ce phénomène est préjudiciable à la candidate. Il est fortement conseillé aux candidates de surveiller le temps afin de pouvoir conclure leur exposé sereinement.

Plusieurs candidates ont choisi de faire des exposés exclusivement oraux. Certains de ces exposés sont de grande qualité, mais l'absence de l'utilisation du tableau est incompatible avec le format « cours » et au propos structuré qui est attendu par le jury. En particulier, ce choix n'est guère propice à la présentation d'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.

Lorsque l'on fait des études de complexité asymptotique d'algorithmes présentés pour être appliqués à des cas concrets, le jury attend que la candidate 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 ?

L'oral comprend un temps d'illustration informatique où la candidate s'appuie sur l'ordinateur pour illustrer son propos. Le jury a été particulièrement satisfait de cette partie. Les candidates ont proposé des illustrations conséquentes, et les ont bien présentées et intégrées dans leurs exposés. Certaines candidates ont même intégré des affichages animés dans leurs programmes pour introduire des éléments de correction des algorithmes. Toute initiative pédagogique pertinente est vouée à être valorisée. 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.

Ces 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. La candidate 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 la candidate exécute ses programmes durant la présentation afin de vérifier leur fonctionnalité.

La discussion autour de dimensions éthiques, sociétales, environnementales, économiques ou juridiques a été bien traitée par une grande majorité des candidates, sachant que tous les sujets contiennent une telle piste. Le jury regrette malgré tout quelques rares cas de refus d'obstacle et rappelle que cette discussion est obligatoire et fait partie du barème de cette épreuve.

L'exposé de la candidate doit prendre la forme d'un cours. L'évaluation de l'utilisation du tableau est partie intégrante de l'évaluation des compétences didactiques et pédagogiques des candidates. Le jury attend que la candidate montre sa capacité à utiliser le tableau pour transmettre des notions et expliquer des exemples bien choisis.

Comme tout oral d'un concours de recrutement d'enseignant, il s'agit, au moins partiellement, d'un exercice de mise en situation professionnelle. En particulier, les candidates ne doivent supposer du jury aucune connaissance préalable du texte. De manière plus large, une réflexion préalable des candidates sur cet aspect de l'épreuve semble indispensable : 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 à la session 2022 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 de la candidate, 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 candidates, de faire préciser ou corriger des points, de fournir des indications pour permettre à la candidate 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.

Les séances de questions se sont très bien déroulées, mais le jury regrette que plusieurs candidates ne réussissent pas à fournir des réponses concises. Il est primordial pour l'efficacité de ces discussions que la candidate ne monopolise pas la parole et laisse le jury rebondir sur ses réponses.