Extraits des rapports et notices concrnant les oraux d'informatique X-ENS de la filière MPI

TP d'algo des ENS

- 3h30 de préparation seul sur machine, puis 20' de passage au tableau (sans son code) + restitution d'une fiche réponse
- consignes pour l'épreuve 2025 (pages 2 à 4)
- détails sur l'environnement disponible (pages 5 à 8)

Oral d'info théorique des ENS

- 30' de préparation puis 30' de passage au tableau (+ év. 30' d'attente après l'épreuve)
- rapport du jury 2024 (pages 9 et 10)

Oral d'info à polytechnique

- 50' au tableau (sans préparation préalable, sans ordi)
- rapport du jury 2024 (pages 11 et 12)

ADS à polytechnique

- 2h de préparation avec document puis 15 à 20mn d’exposé puis 20 mn d’échanges avec l’examinateur
- rapport du jury 2024 (pages 13 et 14)

Département Informatique - ENS Paris-Saclay
Concours/Épreuve pratique d'algorithmique et de programmation du concours commun des écoles normales supérieures

Pour la session 2025, l'épreuve aura lieu à l'ENS Paris-Saclay.

L'image disque utilisée est celle de la session 2024.

Modalités de l'épreuve (à lire attentivement)

Rappel : une pièce d'identité et la convocation sont **indispensables** pour passer les épreuves.

L'épreuve se déroulera à l'École normale supérieure de Paris-Saclay pendant **le mois de juin 2025 (les 13, 14, 20 et 21 juin pour la filière MPI et le 27 juin pour la filière MP)**.

Comme lors des éditions précédentes, l'épreuve comporte trois parties :

1. Installation devant l'ordinateur et familiarisation avec l'environnement (10 min)

> Ces 10 minutes permettent au candidat de se familiariser avec la machine et l'environnement de programmation. Il peut alors poser librement toutes les questions qu'il souhaite, en particulier sur les langages et interfaces.

2. L'épreuve sur ordinateur (3 h 30 min)

> Après les 10 minutes d'installation, les sujets sont distribués et l'épreuve démarre. Le texte du sujet est accompagné d'une **fiche réponse** où devront être consignés les résultats numériques demandés dans le sujet, dans les cases correspondantes. Le sujet est aussi accompagné d'une **clé USB**, qui faudra rendre avec la fiche réponse.
>
> Durant cette partie de l'épreuve, le candidat répond aux questions numériques en remplissant la fiche et prépare **l'interrogation orale** qui va suivre.
>
> Des feuilles de brouillon sont tenues à la disposition des candidats. Nous recommandons à chaque candidat de recopier proprement sur ces feuilles les algorithmes qu'il aura mis en œuvre pour pouvoir les présenter clairement et rapidement à la demande de l'examinateur lors de l'oral qui suivra.

3. Remise de la fiche réponse et présentation orale des résultats lors d'un entretien individuel au tableau (20 min)

> À l'issue des 3h30, le candidat est accompagné à la salle d'interrogation avec ses notes de brouillon et sa fiche. Il remet à l'examinateur sa fiche réponse et l'oral commence.
>
> La présentation orale du candidat se fait sous la conduite de l'examinateur qui pourra, par exemple, en fonction des résultats de la fiche, demander au candidat de décrire précisément les algorithmes mis en œuvre sur telle ou telle question, d'en donner leur analyse de complexité (en temps ou espace) et éventuellement proposer au candidat de corriger tel algorithme s'il s'avère faux.

Cet oral permet au candidat de pouvoir présenter les algorithmes qu'il a mis en œuvre. Il ne s'agit pas d'un oral d'algorithmique indépendant de l'implémentation sur machine. L'évaluation est basée avant tout sur la résolution effective des problèmes posés.

TRÈS IMPORTANT : étant donné la durée très courte de l'oral, chaque candidat doit impérativement préparer soigneusement cet oral pendant la partie sur machine *en rédigeant proprement sur les feuilles (de brouillon à sa disposition) les algorithmes qu'il a conçus* pendant la préparation sur machine, afin de n'avoir qu'à les recopier au tableau, si l'examinateur les lui demande. Les candidats peuvent (et doivent) emporter avec eux tout ce qu'ils ont écrit au brouillon pour passer l'oral.

Remarquez également que cet oral est trop court pour que le candidat puisse concevoir l'algorithme au vol, ou même prendre le risque de ne pas se souvenir correctement de ce qu'il a fait.

Objectifs de l'épreuve

L'objectif de cette épreuve est d'évaluer les capacités des candidats à mettre en œuvre la chaîne complète de résolution d'un problème informatique : analyse des spécifications abstraites, conception d'un algorithme et choix des structures de données, évaluation de sa complexité (coût en temps et en mémoire), programmation sur machine dans l'un des langages proposés, test des programmes sur des petites valeurs des paramètres, et exécution sur des valeurs précises des paramètres. À la fois algorithmique et pratique, cette épreuve est donc transversale et permet de tester la maîtrise du candidat sur la résolution concrète d'un problème informatique.

Déroulement de l'épreuve

À l'entrée dans la salle, les candidats sont invités à prendre place face à l'ordinateur où leur nom est indiqué (voir la description de la configuration logicielle plus bas).

L'épreuve dure 4h10 en tout. Les 10 premières minutes permettent au candidat de s'installer et de se familiariser avec le système. À l'issue de ces 10 minutes, l'épreuve commence par 3h30 de préparation sur machine pendant lesquelles le candidat répond aux questions sur la fiche réponse et prépare la présentation orale de ses résultats. Il doit sauvegarder son travail sur la clé USB. À la fin des 3h30, le candidat est accompagné avec sa fiche réponse, sa clé USB et ses notes à la salle d'interrogation orale où il présente ses résultats sous la conduite d'un examinateur.

Principe des sujets. Les questions portant sur les applications numériques sont posées de façon à pouvoir estimer la qualité et l'efficacité de la programmation : pour répondre juste aux questions portant sur des petites valeurs des paramètres, un programme simple et peu efficace suffit (voire pas de programme du tout) ; en revanche, pour les plus grandes valeurs des paramètres, un programme efficace (en temps et/ou en mémoire) est nécessaire.

Chaque candidat travaille sur des données différentes, générées à partir d'un nombre qui lui est donné individuellement au début de l'épreuve.

Dispositif de sauvegarde. Une clé USB (fixe) sera disponible sur chaque machine pour permettre aux candidats de sauvegarder régulièrement leur travail, afin de limiter les conséquences d'une éventuelle (bien que peu probable) défaillance matérielle ou logicielle. Les candidats doivent sauvegarder leur travail sur cette clé à la fin de la phase de préparation sur machine. La clé est récupérée par les examinateurs en fin d'épreuve.

Réinitialisation des machines. Les disques durs des ordinateurs sont systématiquement

restaurés dans leur état initial entre deux sessions. Ainsi tous les candidats travaillent sur une configuration identique quoi qu'ait pu faire le candidat précédent.

IMPORTANT. Il est strictement **interdit d'importer** des données (ou fichiers de configuration) sous quelque forme que ce soit (clés USB, bluetooth, airport,...) dans les ordinateurs du concours, **sous peine d'exclusion immédiate du concours**.

Configuration logicielle

Système : Linux Debian avec l'environnement graphique XFCE.

Les langages proposés sont les suivants :

- Python,
- Objective Caml,
- C.

Une documentation est disponible ici : langages.pdf.

Image disque d'entraînement

L'image disque suivante vous permettra de vous mettre en condition pour l'épreuve. **Il est vivement conseillé de l'utiliser afin d'éviter de perdre du temps sur des problèmes techniques lors de l'épreuve.**

Image disque : tp-algo-2024.iso (l'identifiant et le mot de passe sont tous les deux "tpalgo") et sa documentation : doc-image-iso.pdf (voir ci-dessus pour une documentation des langages et logiciels disponibles).

Langages et envionnements de programmation

Épreuve pratique de TP ALGO – Concours ENS

1 Choix du langage de programmation

Trois principaux langages de programmation principaux vous sont proposés : Python (**filière MP uniquement**), OCaml et C.

2 Langage Python (filière MP uniquement)

Par défault, la version utilisée est Python 3 (version 3.11.2). Mais Python 2 peut également être disponible (version 2.7.18) dans certaines configurations, voir ci-dessous.

2.1 Environnement de développement Pyzo

Python 3 est disponible dans l'environnement de développement Pyzo. Pour lancer Pyzo, utiliser "Applications" → "Développement" → "Pyzo".

2.2 Environnement de développement Spyder

Python 3 est disponible dans l'environnement de développement Pyzo. Pour lancer Spyder, utiliser "Applications" → "Développement" → "Spyder".

2.3 Éditeur Gedit avec exécution intégrée (greffon "External tools")

Vous trouverez l'éditeur Gedit dans le menu "Applications" → "Accessoires".

Pour exécuter votre code avec Python vous pouvez utiliser le raccourci suivant : touche *Majuscule* + touche *F5*. L'exécution se fera dans la partie inférieure de la fenêtre. Le code est automatiquement sauvegardé avant son exécution.

2.4 Visual Studio Code

Vous le trouverez dans le menu "Applications" → "Développement" .

Pour exécuter votre code Python, vous pouvez lancer Python sur votre fichier ("Run Python File"). Vous pouvez aussi exécuter une sélection de code dans une console Python (combinaison de touches *Majuscule* + *Entrée*, ou bien clique-droit → "Run Python" → "Run Selection/Line in Python Terminal").

2.5 Jupyter

Jupyter peut être lancé dans le menu “Applications” → “Développement”.

Avec Jupyter Notebook, dans la page qui apparaît dans le navigateur, cliquer sur le bouton “Nouveau” à droit de l’écran et choisir “Python 3 (ipykernel)”.

Avec JupyterLab, vous pouvez au choix utiliser un notebook Jupyter, ou lancer une console Python 3 (ipykernel).

2.6 Python 2

Pour passer en Python 2, il faut exécuter dans un terminal la commande `pyenv global 2.7.18`. La commande `python` lancera alors l’interpréteur Python en version 2.7.18.

Pour revenir à Python 3, il faut exécuter dans un terminal la commande `pyenv global system`. La commande `python` lancera alors l’interpréteur Python en version 3.11.2.

3 Langage OCaml

Ocaml est disponible en version 5.2.0, sauf avec Jupyter, dans lequel OCaml n’est disponible qu’en version 4.13.1.

3.1 Éditeur Gedit avec exécution intégrée (greffon "External tools")

Vous trouverez l’éditeur Gedit dans le menu “Applications” → “Accessoires”.

Pour exécuter votre code avec OCaml, vous pouvez utiliser le raccourci suivant : touche *Majuscule* + touche *F4*. L’exécution se fera dans la partie inférieure de la fenêtre. Le code est automatiquement sauvegardé avant son exécution.

3.2 Visual Studio Code

Vous le trouverez dans le menu “Applications” → “Développement” .

En ouvrant un fichier `.ml`, vous pouvez avoir accès aux fonctionnalités offertes par Visual Studio Code (typage, prototypes de fonction, etc.). Vous pouvez exécuter votre code dans un terminal intégré à l’aide de la combinaison de touches *Majuscule* + *Entrée*, ou le compiler avec `ocamlc` ou `ocamlopt` avant de l’exécuter.

3.3 Éditeur Emacs avec mode Tuareg

Vous pouvez utiliser l’éditeur Emacs avec le mode Tuareg. Avec cette option, vous éditerez votre fichier source, et l’exécuterez à travers l’éditeur.

Pour lancer Emacs, vous pouvez utiliser “Applications” → “Développement” → “Emacs (GUI)”. Vous pouvez également le lancer depuis un terminal, avec la commande `emacs`.

Emacs est un éditeur très puissant, mais qui peut dérouter en premier lieu, notamment car il n’utilise pas les mêmes raccourcis que ceux standards de Windows. Néanmoins, il est possible d’utiliser le menu pour toutes les opérations importantes (lire et enregistrer un fichier, copier et coller des zones de texte).

Le mode Tuareg s'active automatiquement si vous ouvrez un fichier avec l'extension `.ml`. (Sinon, pour forcer Emacs à passer en mode Tuareg, appuyez sur *Échap* puis `x`, et tapez `tuareg-mode` puis *Entrée*.)

Deux raccourcis sont importants à connaître :

— "ctrl-c" (touche *Control* conjointement avec la touche `c`) puis "ctrl-b" : exécute tout le fichier.

— "ctrl-c" puis "ctrl-e" : la ligne courante.

Lors de la première exécution, il vous demandera quel interpréteur choisir. Par défault, `ocaml` est utilisé.

3.4 Un éditeur quelconque, et exécution dans un terminal

Vous aurez deux fenêtres : un éditeur, et un terminal pour exécuter votre code. Vous pouvez utiliser l'éditeur de votre choix (par exemple gedit ou emacs). Pour lancer un terminal : "Applications" → "Émulateur de terminal".

Pour exécuter un code, enregistrez le fichier, et exécutez le dans le terminal avec la commande `ocaml votrefichier.ml`.

3.5 Boucle d'interaction OCaml

Vous pouvez directement lancer OCaml en mode interactif (commande `ocaml` dans un terminal). Il est possible de lancer `ledit ocaml` pour avoir des facilités d'édition (gauche/droite et historique). Vous pouvez combiner cette option avec l'option précédente en utilisant la commande `#use "votreficher.ml";;` dans l'interface interactive pour lire un fichier extérieur. (Attention à sauvegarder votre code si vous utilisez cette méthode !)

3.6 Jupyter

Jupyter peut être lancé dans le menu "Applications" → "Développement".

Avec Jupyter Notebook, dans la page qui apparaît dans le navigateur, cliquer sur le bouton "New" à droite de l'écran et choisir "OCaml default".

Avec JupyterLab, vous pouvez au choix utiliser un notebook Jupyter, ou lancer une console OCaml.

Attention : dans Jupyter, Ocaml est disponible uniquement en version 4.13.1.

4 Langage C

4.1 Visual Studio Code

Vous le trouverez dans le menu "Applications" → "Développement" .

Vous pouvez éditer vos fichiers `.c` et compiler vos programmes avec `gcc` dans un terminal intégré.

4.2 Un éditeur quelconque, compilation et exécution dans un terminal

Vous pouvez utiliser l'un des éditeurs disponibles (gedit, emacs, ...) puis compiler votre code à l'aide de `gcc` dans un terminal avant de l'exécuter.

4.3 Exemple de compilation et d'exécution dans un terminal

Si le fichier `main.c` est

```
#include <stdio.h>
int main (void)
{
        printf("Hello World!\n");
}
```

alors dans un terminal, on a

```
tpalgo@tpalgo:~$ gcc -o helloworld main.c
tpalgo@tpalgo:~$ ./helloworld
Hello World!
tpalgo@tpalgo:~$
```

Après avoir reçu un sujet, les candidat·e·s disposent de 30 minutes de préparation, suivies de 28 minutes d'interrogation devant un des examinateurs. En effet, deux minutes sont utilisées par l'examinateur pour aller chercher la/le candidat·e en salle de préparation. Nous rappelons également que jusqu'à quatre candidat·e·s peuvent passer l'épreuve à la suite sur le même sujet, auquel cas la première/le premier d'entre eux est invité·e à patienter 30 minutes dans la salle de préparation à l'issue de son oral, afin de garantir la confidentialité du sujet.

Comme il a été rappelé en préambule des sujets distribués :

> Le but de cette épreuve est d'évaluer la progression des candidates et candidats dans les questions, mais aussi la qualité de leur exposé des solutions, ainsi que l'autonomie dont elles ou ils font preuve pendant l'oral.

Le jury a proposé 17 sujets originaux (10 pour le concours MP et 7 pour le concours MPI), dont la liste est donnée en annexe de ce document. Entre 12 et 16 candidat·e·s étaient interrogé·e·s sur chaque sujet, ce qui permet d'harmoniser les évaluations d'un sujet donné.

Chaque sujet débute par un énoncé qui présente un problème d'informatique et introduit ses notations, puis comporte des questions de difficulté globalement croissante. Les premières questions permettent de démontrer que l'on a compris l'énoncé, ou d'aider à sa compréhension. Les dernières questions d'un sujet sont souvent des questions d'ouverture plus difficiles. En général, il n'est pas attendu que les candidat·e·s traitent l'intégralité des questions.

Les sujets proposés portent sur des thèmes variés de science informatique : langages, graphes, logique, *etc.*, en lien avec les programmes respectifs des filières MP et MPI. Ils nécessitent de s'approprier des concepts nouveaux, démontrer des résultats théoriques et construire des solutions techniques telles que des algorithmes. Bien que l'épreuve mette l'accent sur les concepts théoriques de l'informatique, on veille à garder à l'esprit le sens des objets que l'on étudie, et un certain sens pratique (par exemple dans la conception d'algorithmes ou l'estimation asymptotique de leur complexité) est apprécié.

Le jury tient à féliciter les candidates et candidats qui ont pour la plupart démontré des acquis très solides, et fait preuve d'idées excellentes malgré les contraintes de temps inhérentes à l'épreuve. Même celles et ceux ayant obtenu de moins bonnes notes ont montré des qualités certaines. En particulier, la plupart des candidat·e·s affichent une bonne maîtrise des concepts mathématiques essentiels (induction, disjonction de cas) ainsi qu'une bonne initiative (par exemple le fait de tester un algorithme sur un petit exemple avant d'en déduire le cas général).

Il est usuel que le jury engage une discussion sur certaines questions afin de guider les candidat·e·s dans leur résolution du problème. Une attitude constructive et positive est appréciée lors de ces échanges. A contrario, certaines attitudes peuvent être fortement pénalisées par le jury, comme par exemple ne pas tenir compte des conseils et indications fournis, ou encore essayer de profiter de ces échanges pour utiliser le jury comme "oracle" pour plusieurs questions.

Le jury adresse les conseils suivants aux futur·e·s candidat·e·s.

Concours et autres voies d'accès. Les Écoles Normales Supérieures sont destinées notamment aux personnes intéressées par la recherche ou l'enseignement. Il peut être utile pour les candidats de connaître les différentes voies d'accès à ces écoles, présentées sur la page suivante :

`https://diplome.di.ens.fr/informatique-ens/`

Gestion du temps et de l'espace. Répéter ou recopier l'énoncé lors de l'oral est une perte de temps, l'examinateur ayant lui aussi le sujet sous les yeux. Le jury déconseille aux candidat·e·s l'effacement du tableau à la main, qui le rend rapidement illisible pour l'examinateur.

Raisonnements. Afin de démontrer leur maîtrise du programme d'informatique de leur filière, mais aussi leur compréhension des nouveaux concepts introduits par le sujet traité, le jury invite les candidat·e·s à s'assurer de la cohérence de ce qu'ils décrivent.

Par exemple, si le jury demande un identifiant de variable OCaml, celui-ci ne peut pas, par définition, être un mot clé du langage, ou une application de fonction. Des erreurs d'attention ou de compréhension

peuvent arriver, mais la répétition de ces erreurs de syntaxe ou de typage donne l'impression très négative que la/le candidat·e n'a pas compris le sujet.

De manière générale, le jury conseille aux candidat·e·s d'utiliser les raisonnements par l'absurde uniquement lorsque ceux-ci sont nécessaires, et de préférer les inductions structurelles (par exemple sur un arbre) aux récurrences sur un critère numérique (hauteur de l'arbre). Celles-ci sont usuellement plus efficaces et / ou plus adaptées aux objets étudiés.

Pédagogie, intuition, formalisme. Si la situation le permet, il est souhaitable de décrire (en général oralement ou/et par un dessin) une preuve dans les grandes lignes avant de se lancer dans la preuve formelle. Cela permet à l'examinateur de s'assurer que la/le candidat·e a compris le principe de la preuve. De même, il est souhaitable de décrire un algorithme dans les grandes lignes avant de se lancer dans l'écriture du pseudo-code.

Si une question attend une réponse oui/non, il est conseillé d'annoncer cette réponse avant de se lancer dans la preuve. De même, si une question contient plusieurs résultats à prouver, annoncer lequel sera prouvé en premier, *etc.*

Certains sujets demandent d'absorber plusieurs notions nouvelles, parfois au moyen d'un formalisme assez lourd. On attend alors des candidat·e·s une capacité à s'affranchir du formalisme pour se concentrer sur la signification des objets. Cependant, si l'examinateur demande des précisions, la/le candidat·e doit être en mesure de justifier formellement son raisonnement.

Langage OCaml, programmation fonctionnelle. Lorsqu'une fonction OCaml doit être décrite, le jury suggère fortement aux candidat·e·s de réfléchir en premier abord à la signature de type de la fonction, qui leur permettra de clarifier leurs idées avant de proposer une implémentation.

Pour la filière MPI, le jury a remarqué les candidat·e·s voyaient difficilement le parallèle entre variables libres en logique et dans des expressions OCaml.

Remarques diverses. Le jury tient à souligner, comme l'an dernier, que la majorité des candidat·e·s oublient qu'il y a une différence entre entiers machine de langages comme OCaml et C, et les entiers mathématiques.

Plusieurs sujets demandaient des manipulations de formules logiques qui ont posé problème à certain·e·s candidat·e·s. Par exemple, lors d'un sujet manipulant des formules logiques construites à partir de conjonctions et d'implications, il était important de remarquer que $(a \wedge b) \implies c$ est équivalent à $a \implies b \implies c$. Certain·e·s candidat·e·s ont essayé de revenir à la définition de l'implication. Cela n'était malheureusement pas intéressant, en particulier car la négation et la disjonction n'apparaissaient pas dans les constructions du sujet. Un autre sujet demandait de manipuler $\oplus$ (OU exclusif) et $\wedge$ (ET) en utilisant leur associativité et le fait que $\wedge$ est distributif sur $\oplus$ (ce qui était rappelé). Le parallèle entre ces opérateurs logiques et les opérations d'addition et multuplication dans un corps fini à 2 éléments ne semblait pas connu.

Le concours des ENS s'intéresse à sélectionner des candidat·e·s qui ont majoritairement vocation à faire de la recherche ou de l'enseignement au terme de leurs études, devant un public divers. À ce titre, certains membres du jury souhaitent souligner qu'il est préférable d'utiliser les termes « enfants » et « parent » plutôt que « fils » et « père » lors de raisonnements sur des arbres.

Le jury s'est étonné du fait que certain·e·s candidat·e·s confondent les lettres grecques Φ (phi) et Ψ (psi), ce qui peut induire en erreur l'examinateur lorsque ces deux lettres interviennent dans le même raisonnement.

ÉCOLE POLYTECHNIQUE CONCOURS D'ADMISSION 2024

Épreuve orale d'informatique, Filière MPI

Coefficient (en pourcentage du total des points du concours) **:** 15,7 %

L'épreuve orale d'informatique fondamentale concerne les candidats à l'école polytechnique dans la filière MPI.

Cette année, trente-neuf candidats français et deux candidats étrangers ont participé à l'épreuve. On ne peut que regretter que, sur l'ensemble de ces quarante-et-un candidats, une seule soit une fille.

Les notes des candidats sont comprises entre 7 et 20, avec une moyenne de 12,36 et un écart-type de 3,7. L'histogramme de la figure 1 présente la distribution des notes de l'ensemble des candidats.

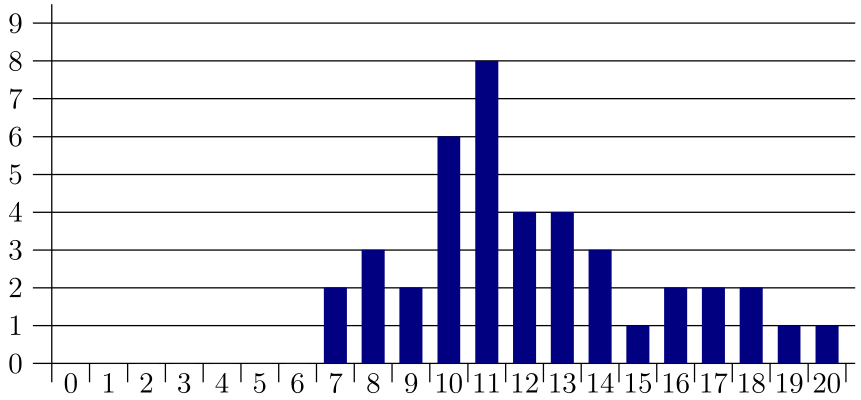

FIGURE 1 – Histogramme des notes de l'épreuve, arrondies à l'entier supérieur

Chaque interrogation durait quarante-huit minutes ; deux minutes étant utilisées par l'examinateur pour s'assurer de l'identité du candidat, lui permettre de signer la feuille d'émargement et lui rappeler les conditions de l'évaluation. Celle-ci se base principalement sur la progression des candidats dans les questions, mais tient également compte de la clarté de leur propos et de l'autonomie dont ils ont fait preuve. Rappelons ici qu'il est important d'être un minimum familier avec le format de l'épreuve, notamment en ce qui concerne l'absence de préparation avant l'oral lui-même : celui-ci commence au moment même où le candidat se voit proposer un sujet.

Le jury a proposé sept sujets originaux, dont deux exemples représentatifs sont présentés en annexe. Chaque sujet débute par un énoncé qui présente un problème d'informatique et introduit ses notations, puis comporte entre sept et onze questions de difficulté globalement croissante. Une ou deux premières questions relativement faciles d'accès permettent aux candidats de vérifier leur compréhension de l'énoncé et de se lancer dans l'interrogation orale. Les questions suivantes, progressivement plus difficiles, occupent la majeure partie du temps de l'interrogation, et visent à permettre de départager les candidats. La plupart des sujets terminent par une ou plusieurs questions considérées comme très difficiles.

En général, il n'est pas attendu des candidats qu'ils traitent l'ensemble des questions dans le temps imparti. Par ailleurs, l'évaluation du candidat tient bien sûr compte de la difficulté des questions qui lui étaient proposées. Ainsi, tel sujet conçu pour être plus difficile d'accès que tel autre est mieux valorisé ; telle question introductive simple mais chronophage est elle aussi valorisée à hauteur du temps qu'elle coûtera aux candidats.

Dans tous les cas, l'objectifs des examinateurs est de permettre au candidat d'avancer au maximum dans le sujet, quitte à lui donner des indications de temps à autre lorsque le candidat semble bloqué. Dans cette perspective, il est indispensable pour les candidats de trouver un équilibre parfois délicat : il convient à la fois d'être suffisamment précis pour convaincre l'examinateur que l'on a compris ce qui se passait, mais de ne pas passer trop de temps sur la question sauf si l'examinateur le demande. Par exemple, il ne faut pas hésiter à donner des éléments de preuve directement à l'oral plutôt que de systématiquement les écrire : si l'examinateur constate que le candidat a manifestement compris ce qu'il avance et que la solution proposée répond à la question, il pourra ainsi inviter le candidat à passer immédiatement à la question suivante.

Les sujets font appel aux différentes compétences nécessaires en science informatique : comprendre des concepts nouveaux, démontrer des résultats théoriques, et construire des solutions techniques telles que des algorithmes.

Le niveau général dont les candidats ont fait preuve pour cette seconde édition de l'épreuve orale d'informatique en voie MPI était excellent, confirmant ainsi les observations effectuées l'an dernier. Les candidats ont montré une très bonne maîtrise des algorithmes, structures de données, ainsi que d'éléments plus théoriques du programme tels que la logique et la théorie des langages. Le plus souvent, ils ont su s'appuyer sur cette maîtrise pour réfléchir de manière pertinente à des problèmes difficiles. Ainsi, les candidats ayant eu au moins 12/20, soit la moitié des candidats, ont proposé une prestation d'excellente facture.

Cette réitération du constat extrêmement satisfaisant déjà formulé en 2023 confirme tout le bien-fondé de la mise en place de la filière MPI pour l'apprentissage de l'informatique.

Enfin, voici quelques conseils méthodologiques à destination des candidats. Tout d'abord, afin de mieux manipuler les objets manipulés pendant l'oral, il est souvent très pertinent de faire des dessins et de regarder de petits cas : en sciences, intuition et rigueur sont toutes deux indispensables, et il ne faudrait pas sacrifier la première à la seconde. Ainsi, trop de candidats ont commencé à s'embourber en utilisant des notations parfois absconses, jusqu'au moment où l'examinateur leur a demandé de dessiner le graphe ou la fonction qu'ils étudiaient, ce qui les a très souvent débloqués. De même, face à une question fermée, plusieurs candidats sont partis bille en tête sur une conjecture formulée, semble-t-il, au hasard (puisqu'elle était fausse une fois sur deux en moyenne) au lieu de commencer par manipuler de petits exemples.

Par ailleurs, comme cela était déjà indiqué l'an dernier, il est évidemment nécessaire de connaître parfaitement son cours. Cette année, par exemple, plusieurs candidats se sont fourvoyés sur la complexité de l'algorithme de Dijkstra, ou encore sur le sens dans lequel doivent être effectuées les réductions visant à démontrer que tel problème est NP-difficile. De tels manquements, heureusement fort rares, ont eu pour double conséquence négative de faire perdre du temps aux candidats et de forcer les examinateurs à les sanctionner numériquement.

En annexe, nous présentons, corrigeons et commentons deux exemples de sujets représentatifs de cette année :
— Algorithme du tri lent
— Codage par couleurs

Épreuve orale d'Analyse de Documents Scientifiques Informatique,

Filière MPI

L'épreuve d'analyse de documents scientifiques en Informatique s'est déroulée pour la deuxième année en 2024. 26 candidat·e·s ont été interrogé·e·s. La moyenne des notes données fut de 11,5/20, et l'écart type de 3,42. L'objectif de ce rapport est d'informer les futurs candidat·e·s et leur professeur·e·s d'informatique sur les exigences de l'épreuve et les critères de notation.

Les textes donnés cette année étaient typiquement plus long que ceux donnés l'année dernière. Pour assurer une difficulté uniforme des textes, ils ont tous été choisis parmi les premiers chapitres de thèses en informatique. Les sujets portaient sur la parallélisation du calcul, les assistants à la preuve, le calcul formel, la cryptographie, la géométrie algorithmique, la calculabilité. Conformément au format de l'épreuve, les candidates et candidats avaient deux heures pour analyser le texte et préparer une présentation, puis 40mn en présence de l'examinatrice.teur. Sur ces 40mn, entre 15 et 20mn devaient être consacrées à l'exposé sur des feuilles, et projetées à l'aide d'une visionneuse. Le reste du temps est consacré à une discussion.

L'exposé Les critères de notation de l'exposé étaient la présentation des feuilles, l'adéquation avec le texte proposé, la pertinence et la cohérence des choix faits, la précision des faits exposés, le choix d'exemples.

Rappelons quelques aspects matériels : il faut venir à cette épreuve avec des crayons de différentes couleurs et une règle. Il faut utiliser ces couleurs pour mettre en valeur titres, exemples, définitions, dessins quand c'est possible. Il faut écrire de manière lisible, typiquement au moins deux fois plus grand que ce qu'on ferait sur une copie. On ne peut pas se permettre de négliger cet aspect.

Il faut pourtant trouver un équilibre, et ne pas consacrer trop de temps à la prépa- ration des feuilles en négligeant la compréhension du texte fourni. Nous conseillons aux candidates et candidats de parcourir rapidement tout le texte avant de choisir les parties à exposer. Les choix faits devront probablement être justifiés pendant la discussion. Sur tous les textes proposés cette année, des choix devaient être faits, et la plupart des candidates et candidats ont fait des exposés cohérents.

La présentation d'exemples, tirés du texte ou du programme, était un aspect important de l'exposé. Il fallait annoncer un plan, présenter les motivations et l'objectif du texte, et si possible donner une conclusion. Certain·e·s candidat·e·s ont présenté en conclusion des enjeux de leur choix reliés au sujet du texte : c'était une bonne amorce pour la discussion, sans garantie évidemment que l'examinatrice.teur les interroge sur ces sujets.

Les textes peuvent être long et ne sont pas forcément extrait du début d'un cours. Ils peuvent contenir des mots ou des notions qui ne sont pas précisément définies et sur lesquels il va falloir se forger une intuition. Nous conseillons de regarder la nature du document et les auteurs s'ils sont donnés. Cela peut aider à contextualiser le texte.

La discussion Les critères de notations pour la discussion étaient les suivants : la fluidité et la clarté des réponses du candidat ou de la candidate, les connaissances mobilisées pendant la discussion, la compréhension du texte en dehors de ce qui a été exposé. Rappelons qu'il s'agit avant tout d'un échange, et que l'enjeu n'est pas tant de tester la candidate ou le candidat sur ses connaissances que de lui permettre de mettre en valeur sa compréhension du texte, de ses enjeux, de restituer dans un contexte des connaissances apprises en cours, en TD ou à d'autres occasions. L'entretien a typiquement porté sur un aspect technique du texte, pouvant donner lieu à un exercice rapide au tableau, et à une discussion sur les enjeux du texte.

On ne demande pas au candidat ou à la candidate de connaître le contexte précis du texte donné, qui se situait souvent bien au-delà du programme de MPI. Par contre, la discussion permet de mettre en valeur le recul que la personne aura pu obtenir sur les connaissances acquises pendant ses études. On peut demander des intuitions sur les objets présents dans le texte : il ne faut pas hésiter à répondre, même si les intuitions ne sont pas parfaites cela permet d'amorcer la discussion.