# Banque MP inter ENS - Session 2017
# Épreuve orale d'informatique fondamentale LCR

---

• Écoles concernées : ÉNS de Paris-Saclay, ÉNS de Lyon et ÉNS de Rennes.

• Coefficients (en % du total d'admission) :
  — Paris-Saclay : MPI : 23,1% - Info : 13,2 %
  — Lyon : MPI : 10,8% - Info : 12,7 % (pour les deux options)
  — Rennes : MPI : 23,1% - Info : 8,6 %

• Membres du jury : B. Bollig, V. Gripon, G. Naves

---

Ce rapport concerne l'épreuve orale d'informatique fondamentale LCR des écoles normales supérieures de Paris-Saclay, Lyon et Rennes, dans les concours MPI et I. Pour l'année 2017, 250 candidats ont été interrogés. L'exercice consiste en un oral de 45 minutes sans préparation. Les notes sont étalées entre 6 et 18, avec une moyenne de 11.7 et un écart-type de 2.3. Un histogramme est donné en fin de document.

Le jury a proposé plus de 40 sujets originaux cette année, typiquement conçus de la façon suivante :
— un préambule introduisant un concept nouveau pour le candidat,
— une ou plusieurs questions permettant au candidat de s'assurer qu'il a compris le concept étudié, et à l'examinateur d'évaluer les capacités du candidat à s'approprier des notions nouvelles,
— une succession de questions de difficulté croissante amenant à la démonstration d'un résultat d'informatique fondamentale lié au concept introduit.

Les sujets proposent en tout une dizaine de questions, les dernières questions étant en général considérées comme très difficiles. Il n'est pas attendu d'un candidat, même très bon, qu'il puisse atteindre la fin du sujet.

Si les concepts abordés sont nouveaux, ils reposent souvent sur des notions et résultats qui sont au programme d'informatique, en particulier en algorithmique, graphes, logique, automates et langages. Les candidats commettant des erreurs sur tout point du programme sont sévèrement sanctionnés. Nous conseillons aux candidats de lire intégralement le programme officiel.

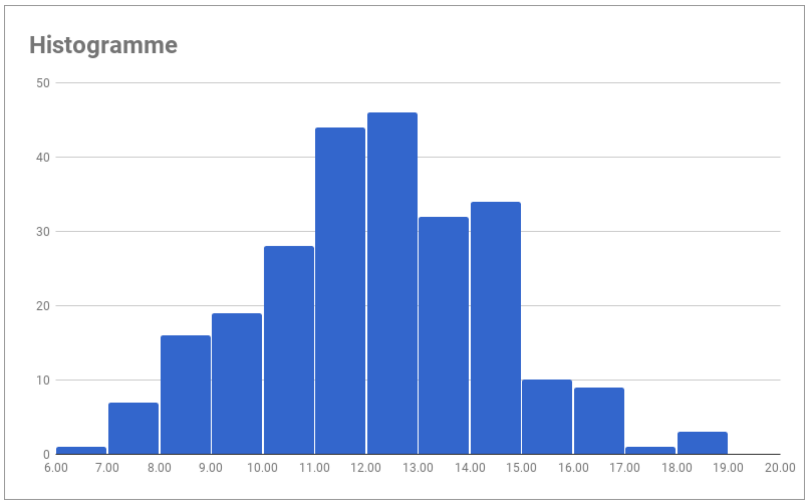
Par ailleurs, nous avons constaté que de nombreux candidats ont aussi acquis une culture en informatique dépassant le cadre du programme officiel. Si nous apprécions ces connaissances, lorsqu'elles sont bien maitrisées, nous déplorons trop souvent leur caractère superficiel, cette superficialité s'étendant parfois aussi au contenu du programme.

Nous avons aussi détecté des faiblesses typiques à de nombreux candidats :
— trop de candidats ne font pas la différence entre les définitions de marche et de chemin dans un graphe. Seul les chemins sont au programme, mais le fait que les arêtes d'un chemin doivent être distinctes a été presque systématiquement oublié. En particulier, la concaténation de deux chemins n'est pas un chemin en général. Une question typique était de montrer que les distances définies par les plus courts chemins d'un graphe vérifient l'inégalité triangulaire. Peu de candidats ont su le faire correctement. On peut aussi faire la distinction entre chemin et chaîne.
— Des candidats rencontrent des difficultés à utiliser les résultats au programme sur la représentation des nombres dans d'autres contextes que l'encodage de données, en particulier exploiter le fait que $n$ bits ne peuvent représenter plus de $2^n$ valeurs différentes. Un exemple typique est pour borner inférieurement le nombre de comparaisons nécessaires pour effectuer un tri, en utilisant comme mot binaire la séquence des résultats des comparaisons.

— les candidats ne font pas toujours la distinction entre un arbre enraciné, tels ceux utilisés pour les arbres binaires de recherche, et un arbre en théorie des graphes, c'est-à-dire un graphe connexe acyclique. Certains supposaient même que les arbres étaient nécessairement d'arité 2.
— proposer des formules sans réfléchir est fortement pénalisé par les examinateurs. On attend de toute façon que les réponses soient justifiées.

Le jury tient à insister sur le caractère théorique de l'épreuve, qui se concentre sur l'informatique fondamentale. En conséquence, on attend des candidats une approche mathématique des problèmes informatiques proposés plutôt qu'une approche "programmation", ce qui n'est malheureusement pas toujours bien compris.

***Nota bene* : A partir de 2018, la durée de l'épreuve orale d'informatique fondamentale (LCR) sera portée à une heure : 30' de préparation et 30' d'interrogation.**




**Ci-après sont donnés 3 exemples de sujets représentatifs.**
— Automates sur les images
— Détection d'agents fiables
— Group testing

# *Automates sur les images*

Fixons un alphabet fini $\Sigma = \{a, b, \ldots\}$. Une image $I$ (sur $\Sigma$) de dimension $h \times \ell$, où $h, \ell \geq 1$, est un tableau dont les entrées sont des éléments de $\Sigma$. Formellement, $I : \{1, \ldots, h\} \times \{1, \ldots, \ell\} \to \Sigma$. Voici une image de dimension $4 \times 8$ sur l'alphabet $\Sigma = \{a, b\}$ :

$$h \left\{ \underbrace{\begin{array}{|c|c|c|c|c|c|c|c|} \hline a & b & b & b & a & b & b & b \\ \hline b & a & a & b & b & a & a & b \\ \hline a & b & b & a & a & b & b & a \\ \hline b & b & a & a & b & b & a & a \\ \hline \end{array}}_{\ell} \right.$$

Alors $h$ est la hauteur et $\ell$ la largeur de l'image. On dénote $\mathit{Images}(\Sigma)$ l'ensemble des images sur $\Sigma$.

**Question 1.** Combien d'images y a-t-il de dimension $h \times \ell$ ?

Nous allons définir un modèle d'automates pour les images. Soit $I$ une image sur $\Sigma$ de dimension $h \times \ell$. La fonction $I^{\#} : \{0, \ldots, h+1\} \times \{0, \ldots, \ell+1\} \to \Sigma \uplus \{\#\}$ est définie par

$$I^{\#}(i,j) = \begin{cases} I(i,j) & \text{si } (i,j) \in \{1, \ldots, h\} \times \{1, \ldots, \ell\} \\ \# & \text{sinon} \end{cases}$$

Par exemple, pour l'image illustrée ci-dessus, on obtient :

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| # | # | # | # | # | # | # | # | # | # |
| # | $a$ | $b$ | $b$ | $b$ | $a$ | $b$ | $b$ | $b$ | # |
| # | $b$ | $a$ | $a$ | $b$ | $b$ | $a$ | $a$ | $b$ | # |
| # | $a$ | $b$ | $b$ | $a$ | $a$ | $b$ | $b$ | $a$ | # |
| # | $b$ | $b$ | $a$ | $a$ | $b$ | $b$ | $a$ | $a$ | # |
| # | # | # | # | # | # | # | # | # | # |

Un **automate d'images** est une paire $\mathcal{A} = (Q, T)$ où $Q$ est un ensemble fini non-vide d'états et $T$ est l'ensemble des transitions. Une transition est une image de dimension $2 \times 2$ sur l'alphabet $(\Sigma \times Q) \uplus \{\#\}$. Par exemple :

| | |
|---|---|
| # | # |
| # | $a, q$ |

Soit $I \in \mathit{Images}(\Sigma)$ et $R \in \mathit{Images}(Q)$, les deux de dimension $h \times \ell$. Soit $I \oplus R \in \mathit{Images}(\Sigma \times Q)$ l'image de dimension $h \times \ell$ définie par $(I \oplus R)(i,j) = (I(i,j), R(i,j))$. Donc, $I \oplus R$ est la "superposition" de $I$ et $R$. On dit que $R$ est un run sur $I$ si **chaque** sous-carré dans $(I \oplus R)^{\#}$ est une transition.

Le langage de $\mathcal{A}$, dénoté $L(\mathcal{A})$, est l'ensemble des images $I \in \mathit{Images}(\Sigma)$ sur lesquelles il existe un run. Un langage $L \subseteq \mathit{Images}(\Sigma)$ est dit *reconnaissable* s'il existe un automate d'images $\mathcal{A}$ tel que $L(\mathcal{A}) = L$.

**Question 2.** Soit $\Sigma = \{a\}$. Pour $h, \ell \geq 1$, soit $I_{h\times\ell}$ l'unique image sur $\Sigma$ de dimension $h \times \ell$.

a) Montrer que $\{I_{1\times 1}\}$ est reconnaissable.

b) Montrer que $\{I_{1\times 2k} \mid k \geq 1\}$ est reconnaissable.

c) Montrer que $\{I_{h\times h} \mid h \geq 1\}$ est reconnaissable.

**Question 3.** Pourquoi est-il justifié de dire que les automates d'images généralisent les automates finis ? Formaliser le fait que les automates sur les images reconnaissent tous les langages réguliers de mots.

**Question 4.** Soit $\Sigma = \{a, b\}$. Montrer que l'ensemble $\{I \in \mathit{Images}(\Sigma) \mid I$ contient un nombre pair de $a\}$ est reconnaissable.

Pour une image $I_1$ de dimension $h \times \ell_1$ et une image $I_2$ de dimension $h \times \ell_2$, soit $I_1 \cdot I_2$ leur "concatenation", qui est une image de dimension $h \times (\ell_1 + \ell_2)$.

**Question 5.** Montrer que si $L_1, L_2 \subseteq \mathit{Images}(\Sigma)$ sont des ensembles reconnaissables, alors $L_1 \cdot L_2 := \{I_1 \cdot I_2 \mid I_1 \in L_1$ et $I_2 \in L_2$ telles que $I_1$ et $I_2$ ont la même hauteur$\}$ est reconnaissable.

**Question 6.** Trouver un algorithme pour le problème suivant : Étant donné un automate d'images $\mathcal{A}$ sur $\Sigma$ et $h \geq 1$, y a-t-il une image $I \in \mathit{Images}(\Sigma)$ de hauteur $h$ telle que $I \in L(\mathcal{A})$ ?

**Question 7.** Soit $\Sigma = \{a\}$. Montrer que $\{I_{h\times 2^h} \mid h \geq 1\}$ est reconnaissable.

**Question 8.** Soit $\Sigma = \{a, b\}$ et $L = \{I \cdot I \mid I \in \mathit{Images}(\Sigma)$ est un carré$\}$.

a) Montrer que $L$ n'est pas reconnaissable.

b) Montrer que $\mathit{Images}(\Sigma) \setminus L$ est reconnaissable.

**Question 9.** En déduire une condition suffisante pour un langage de ne pas être reconnaissable. Appliquer cette condition au langage de la question précédente.

**Question 10.** Montrer que le langage des graphes planaires connexes est reconnaissable (sur un alphabet approprié).

# Détection d'agents fiables

Parmi un groupe de $n$ personnes (ou *agents*) se trouvent au plus $\frac{n-1}{2}$ personnes non-fiables, que l'on cherche à identifier. Pour cela, on peut demander à n'importe quelle personne si une autre personne est fiable. Une personne fiable dira toujours la vérité, et une personne non-fiable peut répondre arbitrairement. Nous cherchons à trouver le nombre minimum $Q(n)$ de questions permettant d'identifier toutes les personnes non-fiables du groupe. On pose les questions une par une et on obtient la réponse immédiatement.

**Question 1:** Dans le cas $n = 3$, comment identifier optimalement les personnes non-fiables ?

**Question 2:** Pourquoi demande-t-on que le nombre de personnes non-fiables soit au plus $\frac{n-1}{2}$ ?

**Question 3:** Montrer que $\left\lfloor \frac{3(n-1)}{2} \right\rfloor$ questions suffisent pour trouver tous les agents non-fiables. On pourra commencer par trouver une solution moins efficace.

On montre maintenant que le nombre de questions nécessaires pour trouver les agents non-fiables est $Q(n) \geq \left\lfloor \frac{3(n-1)}{2} \right\rfloor = q$. Pour cela, on conçoit un algorithme qui répond successivement à chaque question qu'on pose, de sorte que tant que moins de $q$ questions ont été posées, il existe (au moins) deux bipartitions possibles de l'ensemble des agents en *fiables* et *non-fiables*, compatibles avec l'ensemble des réponses données. On pose $k = \left\lfloor \frac{n-1}{2} \right\rfloor$.

Dans une première phase, l'algorithme répond *non-fiable* aux $k-1$ premières questions. Il passe ensuite en deuxième phase.

**Question 4:** On considère le graphe non-orienté $G$ dont les sommets sont les agents $V = \{1, \ldots, n\}$, deux agents sont adjacents si l'un des deux a donné son avis sur l'autre lors des $k-1$ premières questions. Borner inférieurement le nombre de composantes connexes de ce graphe en fonction de $k$.

Nommons $C_1, C_2, \ldots$ les composantes connexes.

**Question 5:** Construire un ensemble $T$ d'agents tel que la partition $T, V \setminus T$ en agents fiables et non-fiables est compatible avec les $k-1$ premières réponses.

**Question 6:** Prouver qu'on peut maintenir une partition cohérente $T, V \setminus T$ pendant les $n+1$ questions suivantes, quelles qu'elles soient.

**Question 7:** Montrer que $Q(n) = \left\lfloor \frac{3(n-1)}{2} \right\rfloor$.

**Question 8:** Supposons qu'une borne $m \leq \left\lfloor \frac{n-1}{2} \right\rfloor$ sur le nombre d'agents non-fiables est connue. Quel est le nombre de questions nécessaires et suffisantes pour déterminer l'ensemble des agents fiables ?

**Question 9:** Quel est le nombre de questions à poser s'il fallait poser toutes les questions avant d'obtenir toutes les réponses ?

# Group testing

On considère dans cet exercice $n$ personnes, numérotées de 1 à $n$. Une maladie grave a touché certaines personnes. Cette maladie peut-être détectée en réalisant un test sanguin, très complexe à mettre en œuvre.

On dénote $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ le vecteur d'état des personnes, tel que $x_i = 1$ si la $i$-ème personne est malade et $x_i = 0$ sinon.

Un *test* correspond à choisir un sous-ensemble $S$ de $\{1, \ldots, n\}$, pour décider si l'une de ces personnes est malade. Ainsi, la valeur du test $A(S)$ est définie comme :

$$A(S) = \begin{cases} 1 & \text{si } \sum_{k \in S} \mathbf{x}_k \geq 1 \\ 0 & \text{sinon} \end{cases}$$

Dans cet exercice, on se questionne sur le nombre minimum de tests à effectuer pour identifier les personnes malades.

On définit $t(d, n)$ comme étant le nombre minimal de tests concurrents à réaliser pour détecter $d$ individus malades dans une population de $n$ individus.

On définit $t^a(d, n)$ comme étant le nombre minimal de tests séquentiels à réaliser pour détecter $d$ individus malades dans une population de $n$ individus.

**Question 1** *Justifier que pour tout $d \geq 1$ et pour tout $n > d$, $1 \leq t^a(d, n) \leq t(d, n) \leq n$.*

On considère l'*anneau de Boole* $\mathcal{A} = (\{0, 1\}, \vee, \wedge, 0, 1)$.

**Question 2** *Montrer que si $d$ n'est pas fixé, identifier les individus malades à l'aide de tests concurrents peut être vu comme inverser un système linéaire dans $\mathcal{A}$.*

**Question 3** *Montrer que pour $d \geq 1$ et $n > d$, $t^a(d, n) \geq d \log_2(n/d)$.*

**Question 4** *Montrer que pour $d \geq 1$ et $n > d$, $t^a(d, n) \leq \mathcal{O}(d \log(n))$.*

**Question 5** *Montrer que pour $n > 1$, $t(1, n) \sim \log_2(n)$.*

On s'intéresse maintenant à majorer la valeur de $t^a(2, n)$.

Notons $k$ un entier strictement positif et $I_k$ tel que $\binom{I_k}{2} < 2^k < \binom{I_k+1}{2}$. On note $n_k$ le plus grand $n$ pour lequel $t^a(2, n) \leq k$.

**Question 6** *Justifier de l'unicité de $I_k$ et du fait que $I_k$ est un majorant de $n_k$.*

On note $u_k = I_k - 1$.

**Question 7** *Montrer que $u_k$ s'écrit $\lfloor 2^{(k+1)/2} - 1/2 \rfloor$.*

**Question 8** *Montrer que $\binom{u_{k+1}+1}{2} - \binom{u_k}{2} > 2^k$ (on se contentera de le vérifier pour $k$ pair).*

**Question 9** *Montrer que pour tout $k \geq 4$, $n_k \leq u_k$.*

**Question 10** *Montrer que pour $d \geq 1$ et $n > d$, $\Omega(d^2 / \log(d) \log(n)) \leq t(d, n) \leq \mathcal{O}(d^2 \log(n))$.*