Second Concours à l'ENS Paris-Saclay et l'ENS Rennes
Interrogation orale d'informatique
Déroulement de l'épreuve

Le sujet se compose de deux problèmes indépendants. Le jury recommande de traiter les deux sujets, au moins partiellement.

Le temps de préparation est de 2 heures. Ensuite la ou le candidat.e exposera les résultats obtenus pendant une heure à l'oral. Bien entendu, le jury posera des questions et pourra revenir sur des questions omises.

Il est autorisé d'admettre le résultat d'une question pour ne pas rester bloqué lors de la phase de préparation.

Il est enfin demandé de mentionner en début d'oral quelles questions ont été traitées dans les deux problèmes. Cela permettra au jury de gérer au mieux le temps et de permettre de présenter l'intégralité des résultats obtenus lors de la préparation.

La rigueur du raisonnement scientifique est un point capital dans l'évaluation de l'épreuve.

Protocoles de communication

Soient S un ensemble fini et f une application de $S \times S$ dans $\{0,1\}$. On donne à Alice $a \in S$ et à Bob $b \in S$. Le but d'Alice et Bob est de calculer $f(a,b)$. Pour cela, ils coopèrent en s'échangeant des messages (suites de bits), leur but étant de calculer f en envoyant le moins de bits possible.

Formellement, un **protocole** P est la donnée :

— d'une fonction $\mathsf{tour}\colon \{0,1\}^* \to \{A, B, \bot\}$ qui décide qui doit envoyer le prochain bit, en fonction des bits envoyés jusqu'alors ;

— d'une fonction $\mathsf{msg}\colon \{0,1\}^* \times S \to \{0,1\}$ qui détermine le bit à envoyer par la personne qui doit envoyer le prochain bit, en fonction des bits envoyés jusqu'alors ;

— d'une fonction $\mathsf{res}\colon \{0,1\}^* \to \{0,1\}$ qui détermine le résultat si le calcul s'arrête.

Sur l'entrée (a,b), le protocole P fonctionne comme suit. On pose $M_0 = \varepsilon$. À l'étape $i \in \mathbf{N}$,

— si $\mathsf{tour}(M_i) = A$ alors Alice envoie le bit $m_i = \mathsf{msg}(M_i, a)$ à Bob et on pose $M_{i+1} = M_i m_i \in \{0,1\}^*$;

— si $\mathsf{tour}(M_i) = B$ alors Bob envoie le bit $m_i = \mathsf{msg}(M_i, b)$ à Alice et on pose $M_{i+1} = M_i m_i \in \{0,1\}^*$;

— sinon, $\mathsf{tour}(M_i) = \bot$, le protocole s'arrête alors et le résultat du calcul est $\mathsf{res}(M_i)$.

Un protocole P **calcule** une fonction f si, pour toute entrée (a,b), le protocole finit par s'arrêter à une étape i donnée et le résultat du calcul est $f(a,b)$. Le nombre i représente le nombre de bits échangés lors du protocole. On note $s_P(a,b) = m_0 m_1 \dots m_{i-1}$ la **communication** échangée sur l'entrée (a,b). On note $D_P(a,b)$ la **longueur** $|s_P(a,b)|$ de la communication, en adoptant la convention que si sur une entrée (a,b) le protocole ne s'arrête pas, on pose $D_P(a,b) = +\infty$.

Pour un protocole P, on note

$$D_P = \max_{(a,b) \in S \times S} D_P(a,b)$$

et pour une fonction f, on note

$$D(f) = \inf\{D_P \mid P \text{ calcule } f\}$$

1. Soient $n \in \mathbf{N}$ et f la fonction définie pour tout $a = a_0 a_1 \cdots a_{n-1} \in \{0,1\}^n$ et $b = b_0 b_1 \cdots b_{n-1} \in \{0,1\}^n$ par $f(a,b) = \sum_{i=0}^{n-1} (a_i + b_i) \mod 2$. Calculer $D(f)$ en précisant le protocole associé.

2. Montrer que pour toute fonction $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, on a l'inégalité

$$D(f) \leq n + 1$$

3. En déduire que pour toute fonction $f\colon S \times S \to \{0,1\}$, on a l'inégalité

$$D(f) \leq \lceil \log_2 |S| \rceil + 1$$

où $\lceil \log_2 |S| \rceil$ dénote la partie entière supérieure du logarithme en base 2 du cardinal de S.
4. Soit L un langage régulier sur un alphabet Σ reconnu par un automate fini déterministe à k états. Soit $f\colon \Sigma^n \times \Sigma^n \to \{0,1\}$ définie pour tout $(a,b) \in \Sigma^n \times \Sigma^n$ par $f(a,b) = 1$ si et seulement si $ab \in L$. Montrer que $D(f) \leq \lceil \log_2 k \rceil + 1$.
5. Montrer que toute fonction f admet un protocole de communication P tel que $D_P = D(f)$ et pour tout $(a,b) \in S^2$, $D_P(a,b) = D_P$.

Un **rectangle** est un sous-ensemble de $S \times S$ de la forme $U \times V$ avec $U, V \subseteq S$. Un rectangle $U \times V$ est **f-uniforme** si f est constante sur $U \times V$. Il est f-vrai si $f = 1$ sur le rectangle, et f-faux sinon.
6. Soit P un protocole de communication pour une fonction f donnée. Montrer par récurrence sur le mot M de $\{0,1\}^*$ que $R_M = \{(a,b) \mid s_P(a,b) \text{ commence par } M\}$ est un rectangle.
7. Soit P un protocole de communication pour une fonction f donnée. Soit M un mot de $\{0,1\}^*$. Déduire de la question précédente que $\{(a,b) \mid s_P(a,b) = M\}$ est un rectangle f-uniforme.

On appelle partition f-uniforme de $S \times S$ un ensemble de rectangles f-uniformes $R_i = U_i \times V_i$ disjoints recouvrant $S \times S$. On note $C(f)$ la taille minimale d'une partition f-uniforme de $S \times S$.
8. Expliquer pourquoi $C(f)$ est bien définie.
9. Montrer que $D(f) \geq \log_2 C(f)$. *On pourra s'aider de la question* 5.
10. Soit S dont le cardinal est une puissance de 2 et $f\colon S \times S \to \{0,1\}$ la fonction définie par $f(x,y) = 1$ si et seulement si $x = y$.

(a) Déterminer $D(f)$.

(b) En déduire, pour tout $n \geq 1$, une estimation du nombre d'états nécessaire pour tout automate déterministe reconnaissant le langage $L_n = \{ww \mid w \in \{0,1\}^n\}$. Fournir un automate déterministe convenable ayant un nombre d'états proche de votre estimation.

On considère une fonction $f\colon S \times S \to \{0,1\}$ quelconque et une partition f-uniforme de $S \times S$ où l'on distingue les rectangles f-vrais des rectangles f-faux. On note donc

$$S \times S = \bigcup_{i \in I} R_i \cup \bigcup_{j \in J} R'_j$$

où $\{R_i = U_i \times V_i \mid i \in I\}$ est l'ensemble des rectangles f-vrais et $\{R'_j \mid j \in J\}$ est l'ensemble des rectangles f-faux.
11. Soient $(a,b) \in S \times S$ et $K \subseteq I$. On suppose que $f(a,b) = 0$. Montrer qu'il existe un rectangle f-faux $U' \times V'$ tel que
— soit $a \in U'$ et $|\{i \in K \mid U_i \cap U' \neq \emptyset\}| \leq |K|/2$
— soit $b \in V'$ et $|\{i \in K \mid V_i \cap V' \neq \emptyset\}| \leq |K|/2$
12. Montrer que $D(f) = \mathcal{O}\big((\log_2 |I|)(\log_2 |J|)\big)$, et en particulier $D(f) = \mathcal{O}\big((\log_2 C(f))^2\big)$.

Répartition de tâches entre deux personnes

Considérons un responsable qui doit gérer le travail d'une équipe de deux personnes Alice et Bob. Chaque tâche i s'effectue en $\ell(i) > 0$ minutes. L'objectif du responsable est de répartir les tâches entre Alice et Bob **le plus équitablement possible**, c'est-à-dire de partitionner l'ensemble L de tâches en deux sous-ensembles L_{Alice} et L_{Bob} telle que $\max(\sum_{i\in L_{Alice}} \ell(i), \sum_{i\in L_{Bob}} \ell(i))$ soit minimal.

Algorithme offline

Chaque jour, le responsable reçoit **une liste complète de tâches** à faire dans la journée qui sont numérotées de 1 à n. Le responsable propose l'algorithme glouton suivant :

1. trier les tâches en les numérotant de façon décroissante en fonction de ℓ :
$$\ell(1) \geq \ell(2) \geq \cdots \geq \ell(n)$$
2. initialiser les sous-ensembles $L_{Alice} \leftarrow \emptyset$ et $L_{Bob} \leftarrow \emptyset$
3. pour i allant de 1 à n faire
   si $\sum_{j\in L_{Alice}} \ell(j) < \sum_{j\in L_{Bob}} \ell(j)$ alors $L_{Alice} \leftarrow L_{Alice} \cup \{i\}$
   sinon $L_{Bob} \leftarrow L_{Bob} \cup \{i\}$
4. retourner L_{Alice} et L_{Bob}

Question 1. Evaluer précisément la complexité de l'algorithme.

Question 2. Montrer que l'algorithme glouton retourne une solution optimale pour $n \leq 4$.

Question 3. Le responsable pense que l'algorithme retourne toujours la solution optimale. Est-ce le cas ?

Considérons une solution optimale qui partage L en deux sous-ensembles L^*_{Alice} et L^*_{Bob} telle que $OPT = \max(\sum_{i\in L^*_{Alice}} \ell(i), \sum_{i\in L^*_{Bob}} \ell(i))$ soit minimal. Notons $W = \sum_{j=1}^{n} \ell(j)$.

Question 4. Donner un minorant de OPT en fonction de W, un autre en fonction de $\ell(i)$. pour toute tâche i.

Soit L_{Alice} et L_{Bob} les sous-ensembles de L fournis par l'algorithme glouton.

Pour les questions suivantes de 5 **à** 8, nous supposerons que

- $\sum_{i\in L_{Alice}} \ell(i) \geq \sum_{i\in L_{Bob}} \ell(i)$;
- k est la dernière tâche insérée dans L_{Alice} par l'algorithme glouton.

Question 5. Montrer que $(\sum_{i\in L_{Alice}} \ell(i)) - \ell(k) \leq \sum_{i\in L_{Bob}} \ell(i)$.

Question 6. Montrer que $\sum_{i\in L_{Alice}} \ell(i) \leq \frac{W+\ell(k)}{2}$.

Question 7. Montrer que

1. si $\ell(k) \leq \frac{OPT}{3}$, alors $\sum_{i\in L_{Alice}} \ell(i) \leq \frac{7}{6}OPT$
2. si $\ell(k) > \frac{OPT}{3}$, alors $k \leq 4$

Question 8. Déduisez-en que $\max(\sum_{i\in L_{Alice}} \ell(i), \sum_{i\in L_{Bob}} \ell(i)) \leq \frac{7}{6}OPT$. La borne est-elle atteinte ?

Calcul de la solution optimale

Le responsable veut désormais un algorithme calculant une solution optimale. Pour cela, on construit une matrice booléenne T dont les lignes sont indexées par les tâches (de 1 à n) et les colonnes par les coûts (de 0 à W). L'élément $T[i, b]$, en ligne i et colonne b, contient 1 si et seulement s'il existe un sous-ensemble des i premières tâches dont le coût total vaut b.

Question 9. Que vaut $T[1, j]$ pour $j \in \{0, \ldots, W\}$?

Question 10. Pour $i \in \{2, \ldots, n\}$, donner la formule de récurrence calculant $T[i, b]$ en fonction de $T[i-1, j]$ pour $j \in \{0, \ldots, W\}$.

Question 11. Concevoir l'algorithme qui remplit la matrice T. Donner sa complexité. Est-elle polynomiale ?

Question 12. On suppose disposer de la matrice T remplie. Pour $i \in \{1, \ldots, n\}$ et $b \in \{0, \ldots, W\}$ tels que $T[i, b] = 1$, concevoir un algorithme qui calcule un sous-ensemble des i premières tâches dont le coût total vaut b. Donner sa complexité.

Question 13. Écrire un algorithme qui retourne la solution optimale. Evaluer sa complexité.

Algorithme en ligne

On suppose maintenant que **les tâches arrivent au fur et à mesure de la journée**.

Question 14. Comment adapter l'algorithme glouton à ce contexte ?

Comparons la solution retournée par l'algorithme de la question 14 à la solution optimale lorsque l'ensemble des tâches est connu à l'avance. Nous noterons par L_{Alice} et L_{Bob}, les deux listes calculées par l'algorithme de la question 14. Nous reprenons la notation OPT. Nous supposons toujours que $\sum_{i \in L_{Alice}} \ell(i) \geq \sum_{i \in L_{Bob}} \ell(i)$.

Question 15. Montrer que $\sum_{i \in L_{Alice}} \ell(i) \leq OPT + \frac{OPT}{2}$. En déduire que

$$\max(\textstyle\sum_{i \in L_{Alice}} \ell(i), \sum_{i \in L_{Bob}} \ell(i)) \leq \frac{3}{2} OPT$$

Peut-on trouver un exemple où l'inégalité précédente est une égalité ?

Protocoles de communication - Correction

Soient S un ensemble fini et f une application de $S \times S$ dans $\{0,1\}$. On donne à Alice $a \in S$ et à Bob $b \in S$. Le but d'Alice et Bob est de calculer $f(a,b)$. Pour cela, ils coopèrent en s'échangeant des messages (suites de bits), leur but étant de calculer f en envoyant le moins de bits possible.

Formellement, un **protocole** P est la donnée :

— d'une fonction $\mathsf{tour}\colon \{0,1\}^* \to \{A, B, \bot\}$ qui décide qui doit envoyer le prochain bit, en fonction des bits envoyés jusqu'alors ;

— d'une fonction $\mathsf{msg}\colon \{0,1\}^* \times S \to \{0,1\}$ qui détermine le bit à envoyer par la personne qui doit envoyer le prochain bit, en fonction des bits envoyés jusqu'alors ;

— d'une fonction $\mathsf{res}\colon \{0,1\}^* \to \{0,1\}$ qui détermine le résultat si le calcul s'arrête.

Sur l'entrée (a,b), le protocole P fonctionne comme suit. On pose $M_0 = \varepsilon$. À l'étape $i \in \mathbf{N}$,

— si $\mathsf{tour}(M_i) = A$ alors Alice envoie le bit $m_i = \mathsf{msg}(M_i, a)$ à Bob et on pose $M_{i+1} = M_i m_i \in \{0,1\}^*$;

— si $\mathsf{tour}(M_i) = B$ alors Bob envoie le bit $m_i = \mathsf{msg}(M_i, b)$ à Alice et on pose $M_{i+1} = M_i m_i \in \{0,1\}^*$;

— sinon, $\mathsf{tour}(M_i) = \bot$, le protocole s'arrête alors et le résultat du calcul est $\mathsf{res}(M_i)$.

Un protocole P **calcule** une fonction f si, pour toute entrée (a,b), le protocole finit par s'arrêter à une étape i donnée et le résultat du calcul est $f(a,b)$. Le nombre i représente le nombre de bits échangés lors du protocole. On note $s_P(a,b) = m_0 m_1 \dots m_{i-1}$ la **communication** échangée sur l'entrée (a,b). On note $D_P(a,b)$ la **longueur** $|s_P(a,b)|$ de la communication, en adoptant la convention que si sur une entrée (a,b) le protocole ne s'arrête pas, on pose $D_P(a,b) = +\infty$.

Pour un protocole P, on note

$$D_P = \max_{(a,b) \in S \times S} D_P(a,b)$$

et pour une fonction f, on note

$$D(f) = \inf\{D_P \mid P \text{ calcule } f\}$$

1. Soient $n \in \mathbf{N}$ et f la fonction définie pour tout $a = a_0 a_1 \cdots a_{n-1} \in \{0,1\}^n$ et $b = b_0 b_1 \cdots b_{n-1} \in \{0,1\}^n$ par $f(a,b) = \sum_{i=0}^{n-1}(a_i + b_i) \mod 2$. Calculer $D(f)$ en précisant le protocole associé.

Solution : Remarquons que $f(a,b) = \sum_{i=0}^{n-1} a_i \mod 2 + \sum_{i=0}^{n-1} b_i \mod 2$, donc il suffit qu'Alice envoie le bit $\sum_{i=0}^{n-1} a_i \mod 2$ et Bob le bit $\sum_{i=0}^{n-1} b_i \mod 2$ pour qu'on puisse calculer le résultat. Ainsi, $D(f) \leq 2$. Par ailleurs, $D(f) > 1$ puisqu'avec un seul message, seul un bit provenant d'un des deux interlocuteurs est échangé, ce qui ne permet pas de conclure (la fonction f dépend de ses deux arguments). On a donc $D(f) = 2$.

2. Montrer que pour toute fonction $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, on a l'inégalité

$$D(f) \leq n+1$$

Solution : L'idée est qu'Alice envoie la suite de n bits de l'entrée A à Bob, qui calcule et envoie alors le résultat. Formellement,

- $\mathsf{tour}(M) = A$ si $|M| < n$;
- $\mathsf{tour}(M) = B$ si $|M| = n$;
- $\mathsf{tour}(M) = \bot$ si $|M| > n$;
- $\mathsf{msg}(M, a) = a_i$ si $M = i < n$ (Alice envoie le bit d'indice i) ;
- $\mathsf{msg}(M, b) = f(M, b)$ si $|M| = n$ (Bob envoie le résultat) ;
- $\mathsf{res}(Mx) = x$ (le résultat est le bit envoyé par Bob).

3. En déduire que pour toute fonction $f\colon S \times S \to \{0,1\}$, on a l'inégalité

$$D(f) \leq \lceil \log_2 |S| \rceil + 1$$

où $\lceil \log_2 |S| \rceil$ dénote la partie entière supérieure du logarithme en base 2 du cardinal de S.

Solution : Il suffit de commencer par coder les éléments de S en base 2, ce qui nécessite $\lceil \log_2 |S| \rceil$ bits. Il faudra alors envoyer ces bits à Bob pour qu'il décode la valeur de a, grâce au protocole décrit en question précédente.

4. Soit L un langage régulier sur un alphabet Σ reconnu par un automate fini déterministe à k états. Soit $f\colon \Sigma^n \times \Sigma^n \to \{0,1\}$ définie pour tout $(a,b) \in \Sigma^n \times \Sigma^n$ par $f(a,b) = 1$ si et seulement si $ab \in L$. Montrer que $D(f) \leq \lceil \log_2 k \rceil + 1$.

Solution : Alice envoie (en le codant en binaire) l'état atteint après la lecture du mot a depuis l'état initial de l'automate déterministe. Bob n'a alors plus qu'à poursuivre le calcul dans l'automate en lisant le mot b, puis envoyer si oui (1) ou non (0) l'état atteint est acceptant.

5. Montrer que toute fonction f admet un protocole de communication P tel que $D_P = D(f)$ et pour tout $(a,b) \in S^2$, $D_P(a,b) = D_P$.

Solution : $D(f)$ est définie comme une borne inférieure à valeur dans $\mathbf{N}$, espace discret : cette borne inférieure est donc un minimum, c'est-à-dire qu'il existe un protocole P tel que $D_P = D(f)$. Pour tout $(a,b) \in S^2$, on a alors $D_P(a,b) \leq D_P$, par définition de D_P. Cependant, on peut perdre du temps en ajoutant des étapes inutiles de communication, pour faire en sorte qu'on ne calcule le résultat qu'à l'étape d'indice D_P. On obtient un protocole P' tel que $D_{P'}(a,b) = D_P = D(f)$. On a toujours $D_{P'} = D(f)$.

Un **rectangle** est un sous-ensemble de $S \times S$ de la forme $U \times V$ avec $U, V \subseteq S$. Un rectangle $U \times V$ est **f-uniforme** si f est constante sur $U \times V$. Il est f-vrai si $f = 1$ sur le rectangle, et f-faux sinon.

6. Soit P un protocole de communication pour une fonction f donnée. Montrer par récurrence sur le mot M de $\{0,1\}^*$ que $R_M = \{(a,b) \mid s_P(a,b) \text{ commence par } M\}$ est un rectangle.

Solution : Montrons par récurrence sur le mot M que $R_M = \{(a,b) \mid s_P(a,b) \text{ commence par } M\}$ est un rectangle. C'est vrai pour $M = \varepsilon$ puisque $R_\varepsilon = S \times S$. Supposons la propriété vraie pour un mot M, et de sorte que $R_M = U \times V$. Soit $x \in \{0,1\}$. Alors $R_{Mx} = \{(a,b) \mid s_P(a,b) \text{ commence par } Mx\}$. Si $\mathsf{tour}(M) = \bot$, alors $R_{Mx} = \emptyset$ est un rectangle. Sinon, supposons par exemple que $\mathsf{tour}(M) = A$. Alors

$$\begin{aligned} R_{Mx} &= \{(a,b) \mid s_P(a,b) \text{ commence par } M \text{ et } \mathsf{msg}(M,a) = x\} \\ &= \{(a,b) \mid s_P(a,b) \text{ commence par } M\} \cap \{(a,b) \mid \mathsf{msg}(M,a) = x\} \\ &= U \times V \cap \{a \mid \mathsf{msg}(M,a) = x\} \times S \\ &= (U \cap \{a \mid \mathsf{msg}(M,a) = x\}) \times V \end{aligned}$$

qui est bien un rectangle.

7. Soit P un protocole de communication pour une fonction f donnée. Soit M un mot de $\{0,1\}^*$. Déduire de la question précédente que $\{(a,b) \mid s_P(a,b) = M\}$ est un rectangle f-uniforme.

Solution : D'après la question précédente, $\{(a,b) \mid s_P(a,b) = M\} = R_M \cap \{(a,b) \mid \mathsf{tour}(M) = \bot\}$ qui vaut soit R_M soit $\emptyset$ qui sont bien des rectangles. Puisque le résultat ne dépend que des messages échangés, le rectangle $\{(a,b) \mid s_P(a,b) = M\}$ est f-uniforme.

On appelle partition f-uniforme de $S \times S$ un ensemble de rectangles f-uniformes $R_i = U_i \times V_i$ disjoints recouvrant $S \times S$. On note $C(f)$ la taille minimale d'une partition f-uniforme de $S \times S$.

8. Expliquer pourquoi $C(f)$ est bien définie.

Solution : L'existence d'une partition f-uniforme est assurée par la partition $(\{s\} \times \{s'\})_{s,s' \in S}$. Tout ensemble non vide d'entiers naturels admet alors un minimum.

9. Montrer que $D(f) \geq \log_2 C(f)$. *On pourra s'aider de la question* 5.

Solution : Soit P un protocole optimal où toutes les paires s'arrêtent après exactement D_P étapes de communication (existe d'après la question 5). Pour chaque mot M de longueur D_P, on a alors un rectangle R_M f-uniforme, et les R_M sont disjoints. Donc on obtient une partition de l'espace en rectangles disjoints. Comme il y a 2^{D_P} mots de taille D_P, on obtient une partition avec moins de 2^{D_P} rectangles. Donc la plus petite partition en a aussi moins, soit $C(f) \leq 2^{D_P} = 2^{D(f)}$. Dit autrement, $D(f) \geq \log_2 C(f)$.

10. Soit S dont le cardinal est une puissance de 2 et $f \colon S \times S \to \{0,1\}$ la fonction définie par $f(x,y) = 1$ si et seulement si $x = y$.

(a) Déterminer $D(f)$.

Solution : D'après la question 2, on sait que $D(f) \leq \log_2 |S| + 1$. De plus, toute partition en rectangles uniformes disjoints contient au moins $|S|+1$ rectangles (puisque chaque couple (x,x) est dans un rectangle différent et il faut au moins un rectangle faux). D'après le résultat de la question précédente, $D(f) > \log_2 |S|$, d'où $D(f) = \log_2 |S| + 1$.

(b) En déduire, pour tout $n \geq 1$, une estimation du nombre d'états nécessaire pour tout automate déterministe reconnaissant le langage $L_n = \{ww \mid w \in \{0,1\}^n\}$. Fournir un automate déterministe convenable ayant un nombre d'états proche de votre estimation.

Solution : En utilisant les notations et le résultat de la question 4 pour $L = L_n$, remarquons que la fonction f est la même que celle de la question 4 dans ce cas. En notant k_n le nombre minimal d'état d'un automate déterministe reconnaissant le langage L_n, on a $D(f) \leq \lceil \log_2 k_n \rceil + 1$. D'après la question précédente, $D(f) = n+1$. Ainsi, $\lceil \log_2 k_n \rceil \geq n$, soit $k_n \geq 2^n$. Il existe en effet un automate déterministe acceptant L_n possédant $3 \times 2^n - 2$ états : il enregistre dans $2^{n+1} - 1$ états les n premières lettres du mot (sous la forme d'un arbre avec 2^n feuilles, donc $2^n + 2^n - 1$ nœuds), puis vérifie que les n dernières lettres sont correctes, avec une copie identique de l'arbre (nécessitant $2^n - 1$ nœuds supplémentaires).

On considère une fonction $f\colon S \times S \to \{0,1\}$ quelconque et une partition f-uniforme de $S \times S$ où l'on distingue les rectangles f-vrais des rectangles f-faux. On note donc

$$S \times S = \bigcup_{i \in I} R_i \cup \bigcup_{j \in J} R'_j$$

où $\{R_i = U_i \times V_i \mid i \in I\}$ est l'ensemble des rectangles f-vrais et $\{R'_j \mid j \in J\}$ est l'ensemble des rectangles f-faux.

11. Soient $(a,b) \in S \times S$ et $K \subseteq I$. On suppose que $f(a,b) = 0$. Montrer qu'il existe un rectangle f-faux $U' \times V'$ tel que

— soit $a \in U'$ et $|\{i \in K \mid U_i \cap U' \neq \emptyset\}| \leq |K|/2$

— soit $b \in V'$ et $|\{i \in K \mid V_i \cap V' \neq \emptyset\}| \leq |K|/2$

Solution : Considérons le rectangle f-faux $U' \times V'$ qui contient (a,b) et montrons qu'il satisfait la propriété. On a à la fois $a \in U'$ et $b \in V'$ donc seules les propriétés de cardinal sont à prouver. Supposons par l'absurde que

$$|\{i \in K \mid U_i \cap U' \neq \emptyset\}| > |K|/2 \text{ et } |\{i \in K \mid V_i \cap V' \neq \emptyset\}| > |K|/2$$

Par lemme des tiroirs, il existe donc $i \in K$ tel qu'on ait à la fois $U_i \cap U' \neq \emptyset$ et $V_i \cap V' \neq \emptyset$. Ceci signifie que les rectangles $U_i \times V_i$ et $U' \times V'$ s'intersectent, ce qui n'est pas possible puisque l'un est f-vrai et l'autre f-faux.

12. Montrer que $D(f) = \mathcal{O}\big((\log_2 |I|)(\log_2 |J|)\big)$, et en particulier $D(f) = \mathcal{O}\big((\log_2 C(f))^2\big)$.

Solution : On propose un protocole de communication qui part de tous les rectangles f-vrais et qui essaie d'en éliminer le plus possible à chaque étape, à l'aide de l'idée de la question précédente.

On part donc de l'ensemble K de tous les rectangles f-vrais. À chaque étape du protocole :

- Alice essaie de trouver un rectangle faux $U' \times V'$ avec $a \in U'$ pour lequel U' intersectent beaucoup des rectangles de K. Si Alice en trouve un :
  - elle l'envoie à Bob, à l'aide de $\lceil \log_2(|J|) \rceil$ bits ;
  - Alice et Bob remplacent alors K par $\{i \in K \mid U_i \cap U' \neq \emptyset\}$.

  Si Alice n'en trouve pas, elle le précise à Bob en envoyant $\lceil \log_2(|J|) \rceil + 1$ bits 0. Bob procède alors de même. S'il trouve un rectangle f-faux, il l'envoie à Alice et Alice et Bob remplacent K par $\{i \in K \mid V_i \cap V' \neq \emptyset\}$. Sinon, d'après la question précédente, Bob sait que $f(a,b) = 1$ et en informe Alice.

Au bout de $\log_2 |I|$ étapes, $|K|$ est de taille nulle et donc Alice et Bob concluent que $f(a,b) = 0$. Chaque étape échange de l'ordre de $2\lceil \log_2(|J|) \rceil + 2$ bits, soit $\mathcal{O}\big((\log_2 |I|)(\log_2 |J|)\big)$ bits au total.

On en déduit donc que $D(f) = \mathcal{O}\big((\log_2 |I|)(\log_2 |J|)\big)$. Comme $|I|$ et $|J|$ sont bornés par $C(f)$, on a en particulier $D(f) = \mathcal{O}\big((\log_2 C(f))^2\big)$.

Répartition de tâches entre deux personnes

Considérons un responsable qui doit gérer le travail d'une équipe de deux personnes Alice et Bob. Chaque tâche i s'effectue en $\ell(i) > 0$ minutes. L'objectif du responsable est de répartir les tâches entre Alice et Bob **le plus équitablement possible**, c'est-à-dire de partitionner l'ensemble L de tâches en deux sous-ensembles L_{Alice} et L_{Bob} telle que $\max(\sum_{i\in L_{Alice}} \ell(i), \sum_{i\in L_{Bob}} \ell(i))$ soit minimal.

Algorithme offline

Chaque jour, le responsable reçoit **une liste complète de tâches** à faire dans la journée qui sont numérotées de 1 à n. Le responsable propose l'algorithme glouton suivant :

1. trier les tâches en les numérotant de façon décroissante en fonction de ℓ :
$$\ell(1) \geq \ell(2) \geq \cdots \geq \ell(n)$$
2. initialiser les sous-ensembles $L_{Alice} \leftarrow \emptyset$ et $L_{Bob} \leftarrow \emptyset$
3. pour i allant de 1 à n faire
   si $\sum_{j\in L_{Alice}} \ell(j) < \sum_{j\in L_{Bob}} \ell(j)$ alors $L_{Alice} \leftarrow L_{Alice} \cup \{i\}$
   sinon $L_{Bob} \leftarrow L_{Bob} \cup \{i\}$
4. retourner L_{Alice} et L_{Bob}

Question 1. Evaluer précisément la complexité de l'algorithme.

Entrée : une liste L de n tâches : chaque tâche i s'effectue en $\ell(i)$ minutes.
Sortie : deux parties disjointes L_{Alice} et L_{Bob} de L

1. Trier les tâches en les numérotant de façon décroissantes en fonction de ℓ :
$$\ell(1) \geq \ell(2) \geq \cdots \geq \ell(n)$$
$\mathcal{O}(n \log n)$ opérations
2. Initialiser les sous-ensembles $L_{Alice} \leftarrow \emptyset$ et $L_{Bob} \leftarrow \emptyset$; $\mathcal{O}(1)$ opérations
3. pour i allant de 1 à n faire — la boucle est faite $\mathcal{O}(n)$ fois
   Si $\sum_{j\in L_{Alice}} \ell(j) < \sum_{j\in L_{Bob}} \ell(j)$ alors $L_{Alice} \leftarrow L_{Alice} \cup \{i\}$
   sinon $L_{Bob} \leftarrow L_{Bob} \cup \{i\}$
4. Retourner L_{Alice} et L_{Bob}

Donc la complexité de l'algorithme est de $\mathcal{O}(n \log n + n) = \mathcal{O}(n \log n)$ si le test dans la boucle se fait en $\mathcal{O}(1)$ opérations. Pour cela, il suffit d'avoir deux variables intermédiaires qui stockent les deux sommes $\sum_{j\in L_{Alice}} \ell(j)$ et $\sum_{j\in L_{Bob}} \ell(j)$.

Question 2. Montrer que l'algorithme glouton retourne une solution optimale pour $n \leq 4$.

1. Pour $n = 1$, il existe deux solutions optimales : soit $(L_{Alice}, L_{Bob}) = (\emptyset, \{1\})$ ou $(L_{Alice}, L_{Bob}) = (\{1\}, \emptyset)$. La première est renvoyée par l'algorithme.
2. Pour $n = 2$, il existe deux solutions optimales : soit $(L_{Alice}, L_{Bob}) = (\{1\}, \{2\})$ ou $(L_{Alice}, L_{Bob}) = (\{2\}, \{1\})$. Remarque : les solutions $(L_{Alice}, L_{Bob}) = (\{1, 2\}, \emptyset)$ et $(L_{Alice}, L_{Bob}) = (\emptyset, \{1, 2\})$ ne sont pas des solutions optimales. La seconde solution optimale est renvoyée par l'algorithme.
3. Pour $n = 3$, nous avons $1 \in L_{Bob}$ et $2 \in L_{Alice}$, par définition de l'algorithme glouton. Comme $\ell(2) \leq \ell(1)$, alors $3 \in L_{Alice}$. Donc la solution retournée par l'algorithme glouton est $(L_{Alice}, L_{Bob}) = (\{2, 3\}, \{1\})$.
   Remarquons, que pour toute solution, une des listes L_{Alice} et L_{Bob} contient au moins deux éléments. Comme $\ell(2) + \ell(3) \leq \ell(1) + \ell(3)$, et $\ell(2) \leq \ell(1) \leq \ell(1) + \ell(3)$, nous avons
   $$\max(\ell(1), \ell(2) + \ell(3)) \leq \max(\ell(2), \ell(1) + \ell(3))$$
   Donc la solution retournée par l'algorithme glouton est optimale.
   $\max(L_{Alice}, L_{Bob}) = \max(\ell(1), \ell(2) + \ell(3))$.
4. Pour $n = 4$, en reprenant les arguments précédents, nous avons $\{1\} \subseteq L_{Bob}$ et $\{2, 3\} \subseteq L_{Alice}$. Donc, la solution optimale $OPT \geq \max(\ell(1), \ell(2) + \ell(3))$
   — Si $\ell(1) < \ell(2) + \ell(3)$, alors $4 \in L_{Bob}$. Donc, l'algorithme glouton fournit une solution telle que $\max(L_{Alice}, L_{Bob}) = \max(\ell(2) + \ell(3), \ell(1) + \ell(4))$. Remarquons que par hypothèse, une solution dont une liste contient 3 éléments n'est pas optimale. Ensuite par énumération de cas : on a
   — $\max(\ell(1) + \ell(3), \ell(2) + \ell(4)) > \max(\ell(2) + \ell(3), \ell(1) + \ell(4))$ car
   $$\ell(1) + \ell(3) \geq \ell(1) + \ell(4)$$
   $$\ell(1) + \ell(3) \geq \ell(2) + \ell(3)$$
   — $\max(\ell(1) + \ell(2), \ell(4) + \ell(3)) > \max(\ell(2) + \ell(3), \ell(1) + \ell(4))$ car
   $$\ell(1) + \ell(2) \geq \ell(1) + \ell(4)$$
   $$\ell(1) + \ell(2) \geq \ell(2) + \ell(3)$$
   L'algorithme glouton fournit donc une solution l'optimale.
   — Si $\ell(2) + \ell(3) \leq \ell(1)$, alors $4 \in L_{Alice}$. Donc, l'algorithme retourne une solution telle que $\max(L_{Alice}, L_{Bob}) = max(\ell(2) + \ell(3) + \ell(4), \ell(1))$.
   Soit une solution S telle que la liste qui contient la tâche 1 contient au moins un autre projet.
   $$\ell(2) + \ell(3) + \ell(4) \leq \ell(1) + \ell(4) \leq \ell(1) + \ell(3) \leq \ell(1) + \ell(2)$$
   Donc
   $$\max(\ell(2) + \ell(3) + \ell(4), \ell(1)) \leq \ell(1) + \ell(4) \leq \ell(1) + \ell(3) \leq \ell(1) + \ell(2)$$
   L'algorithme glouton fournit donc une solution l'optimale.

Dans tous les cas, l'algorithme glouton fournit donc une solution l'optimale.

Question 3. Le responsable pense que l'algorithme retourne toujours la solution optimale. Est-ce le cas ?

Considérons 5 tâches avec $\ell(1) = 3$, $\ell(2) = 3$, $\ell(3) = 2$, $\ell(4) = 2$, $\ell(5) = 2$. Voici les différentes solutions à considérer.

- Solution retournée par l'algorithme glouton : $L_{Bob} = \{1, 3, 5\}$ et $L_{Alice} = \{2, 4\}$. $\max(L_{Alice}, L_{Bob}) = \max(5, 7) = 7$.
- Solution optimale : $L^*_{Alice} = \{1, 2\}$ et $L^*_{Bob} = \{3, 4, 5\}$. Donc $\max(L^*_{Alice}, L^*_{Bob}) = \max(6, 6) = 6$.

Considérons une solution optimale qui partage L en deux sous-ensembles L^*_{Alice} et L^*_{Bob} telle que $OPT = \max(\sum_{i \in L^*_{Alice}} \ell(i), \sum_{i \in L^*_{Bob}} \ell(i))$ soit minimal. Notons $W = \sum_{j=1}^{n} \ell(j)$.

Question 4. Donner un minorant de OPT en fonction de W, un autre en fonction de $\ell(i)$. pour toute tâche i.

Pour toute tâche i, $OPT \geq \ell(i)$ car la tâche i est contenue dans l'une des deux listes.

Soit a et b deux réels positifs tels que $a + b = W$. On a $2\max(a, b) \geq a + b = W$. Donc on a $OPT \geq \frac{W}{2}$.

Soit L_{Alice} et L_{Bob} les sous-ensembles de L fournis par l'algorithme glouton.
Pour les questions suivantes de 5 **à** 8, nous supposerons que

- $\sum_{i \in L_{Alice}} \ell(i) \geq \sum_{i \in L_{Bob}} \ell(i)$;
- k est la dernière tâche insérée dans L_{Alice} par l'algorithme glouton.

Question 5. Montrer que $(\sum_{i \in L_{Alice}} \ell(i)) - \ell(k) \leq \sum_{i \in L_{Bob}} \ell(i)$.

Notons par L'_{Bob} la liste de la variable L_{Bob} lorsque k est inséré par l'algorithme glouton. Puisque k est inséré par l'algorithme glouton en dernier dans L_{Alice}, on a :

$$\sum_{i \in L_{Alice}} \ell(i) - \ell(k) \leq \sum_{i \in L'_{Bob}} \ell(i)$$

Après, l'algorithme insère uniquement des tâches dans la variable L_{Bob}. À la fin, on a la propriété. $\sum_{i \in L'_{Bob}} \ell(i) \leq \sum_{i \in L_{Bob}} \ell(i)$. Donc

$$\sum_{i \in L_{Alice}} \ell(i) - \ell(k) \leq \sum_{i \in L'_{Bob}} \ell(i) \leq \sum_{i \in L_{Bob}} \ell(i)$$

Question 6. Montrer que $\sum_{i \in L_{Alice}} \ell(i) \leq \frac{W + \ell(k)}{2}$.

Tout d'abord, remarquons que $\sum_{i \in L_{Alice}} \ell(i) + \sum_{i \in L_{Bob}} \ell(i) = W$.

$$
\begin{aligned}
(\sum_{i \in L_{Alice}} \ell(i)) - \ell(k) &\leq \sum_{i \in L_{Bob}} \ell(i) \\
(\sum_{i \in L_{Alice}} \ell(i) + \sum_{i \in L_{Alice}} \ell(i)) - \ell(k) &\leq \sum_{i \in L_{Bob}} \ell(i) + \sum_{i \in L_{Alice}} \ell(i) \\
(\sum_{i \in L_{Alice}} \ell(i) + \sum_{i \in L_{Alice}} \ell(i)) - \ell(k) &\leq W \\
2(\sum_{i \in L_{Alice}} \ell(i)) &\leq W + \ell(k)
\end{aligned}
$$

Question 7. Montrer que

1. si $\ell(k) \leq \frac{OPT}{3}$, alors $\sum_{i \in L_{Alice}} \ell(i) \leq \frac{7}{6} OPT$
2. si $\ell(k) > \frac{OPT}{3}$, alors $k \leq 4$

1.

$$
\begin{aligned}
\sum_{i \in L_{Alice}} \ell(i) &\leq \frac{W + \ell(k)}{2} \\
\sum_{i \in L_{Alice}} \ell(i) &\leq OPT + \frac{OPT}{2 \cdot 3}
\end{aligned}
$$

2. Nous allons raisonner par contradiction : nous supposerons que $k > 4$. Comme les tâches sont triés dans l'ordre décroissant en fonction du temps d'exécution, il y aurait au moins 5 tâches ayant un temps d'exécution supérieur à $OPT/3$. Dans toute répartition des 5 premières tâches, l'une des listes contient 3 de ces tâches. Sans perte de généralité, supposons que cela soit L^*_{Alice}. Nous avons $\sum_{i \in L^*_{Alice}} \ell(i) > 3\frac{OPT}{3}$ et cela mène à une contradiction.

Question 8. Déduisez-en que $\max(\sum_{i \in L_{Alice}} \ell(i), \sum_{i \in L_{Bob}} \ell(i)) \leq \frac{7}{6} OPT$. La borne est-elle atteinte ?

Considérons uniquement le sous-ensemble $\mathcal{P}'$ de tâches qui contient $k \leq 4$ tâches. Par la question 2, l'algorithme glouton retourne la solution optimale.

Retournons à la solution gloutonne en considérant maintenant l'ensemble $\mathcal{P}$ des tâches. La liste L_{Alice} n'a pas évolué par définition de la tâche k. Comme nous avons supposé que $\sum_{i \in L_{Alice}} \ell(i) \geq \sum_{i \in L_{Bob}} \ell(i)$, la qualité de solution dépend de L_{Alice}. Et comme c'est une solution optimale d'un sous-ensemble de tâches de $\mathcal{P}$ l'algorithme glouton retourne la solution optimale. Donc

— si $\ell(k) \leq \frac{OPT}{3}$, alors $\sum_{i \in L_{Alice}} \ell(i) \leq \frac{7}{6} OPT$

— si $\ell(k) > \frac{OPT}{3}$, alors $\sum_{i \in L_{Alice}} \ell(i) = OPT$

on peut en déduire

$$\max(\sum_{i \in L_{Alice}} \ell(i), \sum_{i \in L_{Bob}} \ell(i)) \leq \frac{7}{6} OPT$$

L'algorithme glouton est une 7/6-approximation. Cette borne est atteinte à cause de l'exemple de la question 3.

Calcul de la solution optimale

Le responsable veut désormais un algorithme calculant une solution optimale. Pour cela, on construit une matrice booléenne T dont les lignes sont indexées par les tâches (de 1 à n) et les colonnes par les coûts (de 0 à W). L'élément $T[i, b]$, en ligne i et colonne b, contient 1 si et seulement s'il existe un sous-ensemble des i premières tâches dont le coût total vaut b.

Question 9. Que vaut $T[1, j]$ pour $j \in \{0, \ldots, W\}$?

Il suffit de considérer que les sous-ensembles parmi la première tâche de $\mathcal{P}$ est soit l'ensemble vide soit l'ensemble $\{1\}$: $T[1, 0] = T[1, \ell(1)] = 1$.

Donc

$$T[1, j] = \begin{cases} 1 & si \quad j = 0 \ ou \ j = \ell(1) \\ 0 & sinon \end{cases}$$

Question 10. Pour $i \in \{2, \ldots, n\}$, donner la formule de récurrence calculant $T[i, b]$ en fonction de $T[i-1, j]$ pour $j \in \{0, \ldots, W\}$.

Il suffit de considérer qu'un sous-ensemble parmi les i premières tâches de $\mathcal{P}$ est soit un sous-ensemble parmi les $i-1$ tâches de $\mathcal{P}$ soit un sous-ensemble parmi les $i-1$ premières tâches de $\mathcal{P}$ union la tâche i.

Donc

$$T[i, j] = \begin{cases} 1 & si \quad T[i-1, j] = 1 \\ 1 & si \quad \ell(i) \leq j \ et \ T[i-1, j-\ell(i)] = 1 \\ 0 & sinon \end{cases}$$

Question 11. Concevoir l'algorithme qui remplit la matrice T. Donner sa complexité. Est-elle polynomiale ?

1. Pour j allant de 0 à W faire $T[1, j] \leftarrow 0$;
2. $T[1, 0] \leftarrow 0$; $T[1, \ell(1)] \leftarrow 0$
3. Pour i allant de 2 à n faire
   (a) Pour j allant de 0 à W faire
      i. $T[i, j] \leftarrow T[i-1, j]$
      ii. Si $(\ell(i) \leq j)$, alors $T[i, j] = \max(T[i, j], T[i-1, j-\ell(i)])$

4. retourner T

La complexité de l'algorithme est $\mathcal{O}(n \cdot W)$. L'algorthme est polynomial si les entiers sont codés en unaire. Si ce n'est pas le cas, il est exponentiel.

Question 12. On suppose disposer de la matrice T remplie. Pour $i \in \{1, \ldots, n\}$ et $b \in \{0, \ldots, W\}$ tels que $T[i, b] = 1$, concevoir un algorithme qui calcule un sous-ensemble des i premières tâches dont le coût total vaut b. Donner sa complexité.

1. $S \leftarrow \emptyset$; $j \leftarrow b$
2. Pour i allant de k à 2 faire
   (a) Si $(b \geq \ell(i))$ et $T[i-1, j-\ell(i)] = 1$ faire
   $S \leftarrow S - \{j\}$; $j \leftarrow j - \ell(i)$
3. retourner S

La complexité de l'algorithme est $\mathcal{O}(n)$.

Question 13. Écrire un algorithme qui retourne la solution optimale. Evaluer sa complexité.

1. $T \leftarrow remplir{-}tableau(\mathcal{P}, W)$;
2. Trouver le j^* tel que $T[n, j^*] = 1$ et que pour tout $j > j^*$ $T[n, j^*] = 0$ c'est-à-dire
   (a) $j^* \leftarrow W/2$
   (b) Tant que $(T[n, j^*] == 0)$ faire $j^* \leftarrow j^* - 1$;
3. $S \leftarrow calculer{-}solution{-}tableau(T, \mathcal{P}, n, j^*)$ (algorithme de la question précedente)

La complexité de l'algorithme est $\mathcal{O}(n \cdot W)$.

Algorithme en ligne

On suppose maintenant que **les tâches arrivent au fur et à mesure de la journée**.

Question 14. Comment adapter l'algorithme glouton à ce contexte ?

1. au début de chaque journée, Alice et Bob n'ont aucune tâche affectée

$$L_{Alice} \leftarrow \emptyset \;\; et \;\; L_{Bob} \leftarrow \emptyset$$

2. A l'arrivée de la tâche i,
   Si $\sum_{j \in L_{Alice}} \ell(j) < \sum_{j \in L_{Bob}} \ell(j)$ alors $L_{Alice} \leftarrow L_{Alice} \cup \{i\}$
   sinon $L_{Bob} \leftarrow L_{Bob} \cup \{i\}$

Comparons la solution retournée par l'algorithme de la question 14 à la solution optimale lorsque l'ensemble des tâches est connu à l'avance. Nous noterons par L_{Alice} et L_{Bob}, les deux listes calculées par l'algorithme de la question 14. Nous reprenons la notation OPT. Nous supposons toujours que $\sum_{i \in L_{Alice}} \ell(i) \geq \sum_{i \in L_{Bob}} \ell(i)$.

Question 15. Montrer que $\sum_{i \in L_{Alice}} \ell(i) \leq OPT + \frac{OPT}{2}$. En déduire que

$$\max(\sum_{i\in L_{Alice}} \ell(i), \sum_{i\in L_{Bob}} \ell(i)) \leq \frac{3}{2}OPT$$

Peut-on trouver un exemple où l'inégalité précédente est une égalité ?

Les résultats des questions 5 et 6 restent vrai pour l'algorithme en ligne. En effet, aucune hypothèse sur l'ordre des tâches n'a été faite pour prouver ces deux questions. Comme $OPT \geq \frac{W}{2}$ et $OPT \geq \ell(k)$, on a

$$\sum_{i\in L_{Alice}} \ell(i) \leq \frac{W + \ell(k)}{2} \leq OPT + \frac{OPT}{2}$$

On a donc directement que

$$\max(\sum_{i\in L_{Alice}} \ell(i), \sum_{i\in L_{Bob}} \ell(i)) \leq \frac{3}{2}OPT$$

L'algorithme glouton est une 3/2-approximation.

Considérons 3 tâches avec $\ell(1) = 1$, $\ell(2) = 1$, $\ell(3) = 2$. Voici les différentes solutions à considérer.

- Solution retournée par l'algorithme en ligne : $L_{Bob} = \{1, 3\}$ et $L_{Alice} = \{2\}$. $\max(L_{Alice}, L_{Bob}) = \max(3, 1) = 3$.
- Solution optimale : $L^*_{Alice} = \{1, 2\}$ et $L^*_{Bob} = \{3\}$. Donc $\max(L^*_{Alice}, L^*_{Bob}) = \max(2, 2) = 2$.

Cette borne est atteinte à cause de l'exemple.