



# Olympiades d'Informatique

<http://uclouvain.acm-sc.be/olympiades>

## Exemples de questions pour les secondaires



Ce document propose des exemples de questions pour le concours destiné aux élèves du secondaire. La première section donne des exemples de questions de logique tandis que la seconde propose des exemples de questions algorithmiques. Nous vous conseillons de lire le document « *Introduction à l'algorithmique* » pour comprendre les exemples de solution donnés.

## 1 Logique



### 1.1 Les sabliers

Au moyen de deux sabliers (un de 7 minutes et un de 4 minutes), vous devez mesurer 9 minutes. Vous pouvez d'abord manipuler les sabliers avant de commencer à compter les 9 minutes. On vous demande de trouver la solution qui prenne le moins de temps en tout.

On dispose de deux sabliers, permettant de mesurer 4 et 7 minutes. On va appeler le premier  $A$  et le second  $B$ . On doit pouvoir mesurer 9 minutes en tout, mais on ne doit pas commencer directement à compter, on peut d'abord effectuer un certain nombre d'opérations avec les sabliers. Essentiellement, une seule opération est possible : *retourner*. On peut n'en retourner qu'un seul ou les deux à la fois. La question demande de trouver la solution qui prenne le moins de temps en tout. Voici une solution possible, chaque sablier est accompagné de deux chiffres  $x/y$ . Le chiffre  $x$  représente le temps qu'il reste à couler et  $y$  celui déjà écoulé.

Opération	Observation	 $A$	 $B$	
Retourner $A$ et $B$		4/0	7/0	
	$A$ est vide	0/4	3/4	(4 minutes écoulées)
Retourner $A$		4/0	3/4	<b>début du chrono</b>
	$B$ est vide	1/3	0/7	(3 minutes écoulées)
Retourner $A$ et $B$		3/1	7/0	
	$A$ est vide	0/4	4/3	(3 minutes écoulées)
Retourner $A$ et $B$		4/0	3/4	
	$B$ est vide	1/3	0/7	(3 minutes écoulées)
				<b>fin du chrono</b>

Cette solution prend donc 13 minutes en tout, ce n'est pas la meilleure. Une autre solution est possible et ne prend qu'exactly 9 minutes, c'est la meilleure. Elle est reprise ci-dessous :

Opération	Observation	 $A$	 $B$	
Retourner $A$ et $B$		4/0	7/0	
	$A$ est vide	0/4	3/4	(4 minutes écoulées)
Retourner $A$		4/0	3/4	
	$B$ est vide	1/3	0/7	(3 minutes écoulées)
Retourner $B$		1/3	7/0	
	$A$ est vide	0/4	6/1	(1 minute écoulée)
Retourner $B$		0/4	1/6	
	$B$ est vide	0/4	0/7	(1 minute écoulée)



## 1.2 Quelle est la question ?

Imaginons qu'il existe une machine qui soit capable de répondre correctement à n'importe quelle question fermée dont la réponse est binaire (oui ou non), et ce via deux LEDs de couleur (vert pour oui et rouge pour non). Vu la popularité évidente de cette machine, deux firmes chinoises (CHEMINDELAVÉRITÉ et SENTIERDELAVÉRITÉ) ont sans attendre créé leur propre modèle d'une telle machine.

Chaque firme a sa propre notion des couleurs. Une des deux firmes utilise le vert pour oui et le rouge pour non ; l'autre firme fait le contraire. Malheureusement, les machines ne sont livrées qu'avec un mode d'emploi en chinois et vous êtes incapables de comprendre quoi que ce soit et personne ne sait vous aider. De plus, les deux modèles sont tous les deux vendus par les mêmes détaillants.

Après quelques semaines d'attente, vous recevez enfin un modèle chinois de cette machine. Curieux comme vous êtes, vous vous posez plusieurs questions :

- (a) Quelle question faut-il poser à la machine pour savoir de quelle firme elle provient ?
- (b) Quelle est la question qu'il faut poser pour connaître le code de couleur utilisé par la firme CHEMINDELAVÉRITÉ ?
- (c) Quelle question faut-il poser à la machine pour allumer la LED verte, peu importe de quelle firme provient la machine ?

## 1.3 Les chiens et les chats

Jean a un certain nombre de chiens et de chats. Si l'un de ses chats devient un chien, Jean aura autant de chiens que de chats. Par contre, si l'un de ses chiens devenait un chat, Jean aurait deux fois plus de chats que de chiens. Combien de chiens et de chats Jean a-t-il ?

## 1.4 Paiement

Jean, Marc et Sébastien se partagent un repas. Jean apporte cinq miches de pain, Marc en apporte trois et Sébastien n'apporte rien du tout. Les trois hommes se répartissent équitablement toute la nourriture apportée. À la fin du repas, Sébastien propose de payer sa part et pose huit euros sur la table avant de partir. Jean, voyant les huit pièces de un euro, décide d'en prendre cinq pour les cinq miches de pain qu'il a partagées. Il reste donc trois euros pour Marc, qui a partagé trois miches de pain. Marc s'offusque et estime qu'il a droit à plus d'argent ! Quel est votre avis ? Quelle est la répartition équitable des huit euros donnés par Sébastien ?

## 1.5 Question d'âge(s)

Jean a été enfant pendant un quart de sa vie, jeune homme pendant un cinquième et père de famille pendant un tiers. Puis, il a encore vécu 13 ans jusqu'à aujourd'hui. Quel âge a-t-il aujourd'hui ?



## 2 Algorithmique

### 2.1 Le nombre de diviseurs

Étant donné un nombre entier  $n > 0$ , écrire un algorithme qui calcule le nombre de diviseurs entiers positifs que possède  $n$ .

On reçoit comme donnée un nombre entier strictement positif  $n$ . Il faut calculer le nombre de diviseurs entiers positifs que possède  $n$ . Il faut tout d'abord se rappeler de la définition de *diviseur*. Le nombre entier positif  $d$  est diviseur de  $n$  si  $d$  divise parfaitement  $n$ , c'est-à-dire qu'il existe un entier  $k > 0$  tel que  $n = k \times d$ . Les diviseurs de  $n$  sont donc obligatoirement compris entre 1 et  $n$ . À partir de ce constat, on peut écrire un algorithme qui teste, pour chaque entier  $d$  compris entre 1 et  $n$ , si celui-ci divise  $n$ . Une solution en français peut donc s'écrire comme :

```
Initialiser un compteur cnt à zéro.  
Pour chaque entier  $d = 1, 2, \dots, n$  :  
    Si  $d$  divise  $n$ , alors :  
        Ajouter 1 à cnt.  
→ Le nombre  $n$  possède cnt diviseurs entiers positifs.
```

On peut également écrire une solution plus rigoureuse en utilisant du pseudo-code comme montré ci-dessous. L'opération  $n \bmod d$  permet de connaître le reste de la division entière de  $n$  par  $d$ . Si ce dernier est égal à zéro, c'est que la division est exacte et que  $d$  est un diviseur de  $n$ .

---

**Algorithme 1** : Le nombre de diviseurs.

---

**Input** :  $n$ , un nombre entier positif

**Output** : Le nombre de diviseurs entiers positifs de  $n$

```
1   $cnt \leftarrow 0$   
2  for  $d \leftarrow 1$  to  $n$  do  
3      if  $n \bmod d = 0$  then  
4           $cnt \leftarrow cnt + 1$   
5  return  $cnt$ 
```

---

### 2.2 Le tapis de cartes

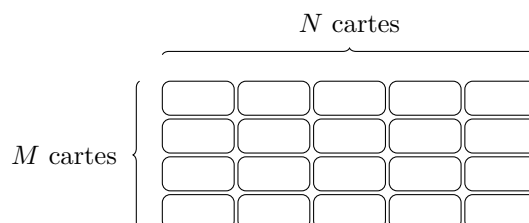
Alice et Bob disposent d'un très grand nombre (on le suppose illimité) de cartes à jouer rectangulaires, toutes de dimensions identiques. Ils désirent les disposer côte à côte, dans le même sens et sans les découper, afin de former un carré.

Aidez-les à trouver quelles sont les dimensions minimales de ce carré. Si  $L$  est la largeur d'une carte et  $H$  sa hauteur (toutes deux des valeurs entières strictement positives), on vous demande d'écrire un algorithme pour calculer  $X$ , le nombre de cartes nécessaires pour former le carré.

La première chose à faire est de bien comprendre le problème. On va donc disposer des cartes rectangulaires côte à côte, dans le même sens, pour former un tapis de cartes. Connaissant la largeur



et la hauteur d'une carte, on souhaite trouver le nombre de cartes qu'on va devoir utiliser pour former un carré de dimension minimale. Voici ce que ça donne en dessin si on place les cartes horizontalement :



La hauteur du tapis vaut donc  $M \times L$  et sa largeur  $N \times H$ . On souhaite que le tapis soit carré et donc que  $M \times L = N \times H$  et ce qu'on cherche, c'est le nombre minimal de cartes nécessaires, c'est-à-dire  $M \times N$ . Pour réussir à résoudre ce problème, il faut se rendre compte que pour trouver la solution, il suffit de calculer le plus petit commun multiple (PPCM) entre  $L$  et  $H$ . On trouve ensuite :

$$M = \frac{P}{H} \quad \text{et} \quad N = \frac{P}{L} \quad \text{avec } P = \text{PPCM}(L, H)$$

Le problème revient donc à trouver le PPCM entre deux nombres entiers strictement positifs  $L$  et  $H$ . Une façon de faire consiste à parcourir tous les multiples de  $L$  et tester s'ils sont également multiple de  $H$ . On garde bien entendu le plus petit. On peut écrire l'algorithme en français :

Pour chaque entier  $m = 1, 2, \dots, H$  :  
Si  $L \times m$  est un multiple de  $H$ , alors :  
→  $L \times m$  est le PPCM de  $L$  et  $H$ .

$L \times m$  est multiple de  $H$  revient à dire que  $H$  divise  $L \times m$ . On peut dès lors écrire un algorithme en pseudo-code en utilisant l'opérateur mod qu'on a vu à la question précédente. On ne teste que les entiers de 1 à  $H$ . En effet, on sait bien qu'au pire le PPCM entre  $L$  et  $H$  vaut exactement  $L \times H$  (ceci se produisant lorsque  $L$  et  $H$  n'ont aucun diviseur commun).

---

**Algorithme 2** : Le tapis de cartes.

---

**Input** :  $L$  et  $H$ , nombres entiers strictement positifs, largeur et hauteur d'une carte

**Output** : Nombre de cartes nécessaires pour former un tapis carré de côté de longueur minimale en plaçant des cartes côte à côte dans le même sens

```
1 // Calcul du PPCM entre L et H
2 m ← 1
3 found ← false
4 while not found and m ≤ H do
5     if L × m mod H = 0 then
6         found ← true
7         m ← m + 1
8 // Calcul du nombre de cartes
9 ppcm ← L × m
10 M ← ppcm/H
11 N ← ppcm/L
12 return M × N
```

---



## 2.3 La gestion d'un hangar de stockage

Vous êtes en charge de la gestion du hangar de stockage d'une grande surface pouvant stocker  $X$  produits. Tous les jours, un camion vient, apporte  $Y$  nouveaux produits et enlève les  $Y$  produits les plus anciens du stock. Les produits livrés par le camion sont toujours considérés comme neufs. Les produits du stock et ceux du camion sont représentés par des listes  $stock = \langle s_1, s_2, \dots, s_X \rangle$  et  $camion = \langle c_1, c_2, \dots, c_Y \rangle$ .

- Écrivez un algorithme qui prend en entrée l'état du stock et le contenu du camion et renvoie en sortie l'état du stock en fin de journée.
- Étendez l'algorithme précédent afin de gérer le cas où le camion apporte des produits d'âges différents. Vous pouvez connaître l'âge d'un produit  $a$  en utilisant la fonction  $age(a)$ .
- Chaque soir, un ouvrier vérifie que les produits ne sont pas trop vieux pour être vendus. Un produit âgé de plus de 126 jours (au moment de la vérification) est considéré comme trop vieux et doit être détruit. Écrivez une fonction qui prend en entrée l'état du stock et renvoie l'état du stock après vérification et éventuelle destruction de certains produits.

## 2.4 Longueur d'un nombre

Écrivez un algorithme qui calcule le nombre de chiffres d'un entier strictement positif  $n$ .

## 2.5 Classement de voitures

Vous êtes rédacteur en chef d'un magazine automobile et souhaitez proposer un classement de différents modèles de voitures à vos lecteurs. Pour ce faire, vous avez choisi cinq critères numériques pour lesquels vous possédez les valeurs pour chaque modèle de voiture :

- longueur ( $L$ ) ;
- largeur ( $W$ ) ;
- hauteur ( $H$ ) ;
- nombre de sièges ( $S$ ) ;
- nombre d'airbags ( $A$ ).

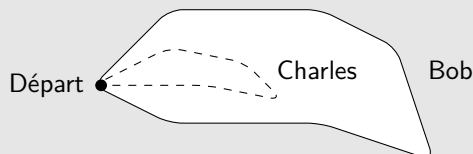
Chaque modèle de voiture est identifié par une liste de cinq valeurs  $\langle L, W, H, S, A \rangle$  correspondant à ces cinq critères. Les modèles de voiture seront classés en fonction des valeurs de la liste **criteres**. Par exemple, si la liste contient  $\langle 5, 1, 4 \rangle$ , cela signifie que les modèles seront classés selon le nombre d'airbags (5), en cas d'égalité, on utilisera la longueur (1) et enfin le nombre de sièges (4).

Écrivez un algorithme qui permet de trier une liste de modèles de voiture en fonction d'une liste de critères donnée. Votre algorithme doit être le plus simple et efficace possible.



## 2.6 On va courir un peu ?

Bob et Charles ont décidé qu'ils allaient courir un peu pour garder la forme. Ils ont chacun leur parcours favori qui part du même point et qui y revient également.



Le trajet de Bob fait  $A$  km de long tandis que celui de Charles fait  $B$  km de long. En supposant que  $A$  et  $B$  sont des naturels non nuls, écrivez un algorithme qui calcule le nombre de tours que vont faire Bob et Charles avant de tous les deux se recroiser au point de départ.

## 2.7 Jeu de Nim

On dispose sur la table un nombre  $n$  d'allumettes. Le jeu se joue à deux et se déroule par tours. Chaque joueur va pouvoir retirer 1, 2 ou 3 allumettes. Le perdant est le joueur qui joue le dernier tour. Écrivez un algorithme qui décrit le comportement qu'un joueur doit avoir pour toujours gagner. Le joueur doit décider s'il commence ou laisse la main ; ensuite, il décide combien d'allumettes il retire à chaque tour.

Vous pouvez utiliser les trois fonctions suivantes :

- `setBegin(i)` permet de décider si vous commencez ( $i = 0$ ) ou si vous laissez la main à l'adversaire ( $i = 1$ ).
- `withdraw(i)` permet de retirer  $i$  allumettes (avec  $i = 1, 2$  ou  $3$ ).
- `getAction()` permet de savoir combien d'allumettes ont été retirées par l'adversaire lors de son dernier coup.

## 2.8 Pyramide

Construisons une pyramide de boules, toutes les boules se trouvant à un même étage étant de la même couleur (blanches ou noires). Les étages contiennent successivement (en partant du haut vers le bas de la pyramide)  $1, 4, 9, 16, \dots, i^2, (i+1)^2, \dots$  boules.

Deux étages successifs ne peuvent avoir la même couleur. Il y aura donc une alternance entre les étages blancs et noirs. Écrivez un algorithme qui prend comme entrée un nombre de boules blanches  $n_B$  et un nombre de boules noires  $n_N$  et qui renvoie :

- $-1$  s'il est impossible de construire une pyramide avec les nombres  $n_B$  et  $n_N$  ;
- $0$  si la boule au sommet de la pyramide est blanche ;
- $1$  si la boule au sommet de la pyramide est noire.



## 2.9 Problème de Joséphus

Nous allons procéder à l'élimination de  $N$  soldats, numérotés de 0 à  $N - 1$  et assis en cercle. Nous commençons par tuer le soldat portant le numéro  $D$ . Ensuite, nous tuons le soldat vivant assis  $J$  places plus loin dans le cercle, en tournant toujours dans le même sens, et ainsi de suite jusqu'à ce qu'il ne reste qu'un seul survivant.

Par exemple, pour  $N = 10$ ,  $D = 3$  et  $J = 4$ , les morts se feront dans l'ordre suivant : 3, 7, 1, 6, 2, 9, 8, 0 et 5. Le survivant est donc le soldat portant le numéro 4.

Écrivez un algorithme qui, pour des naturels strictement positifs  $N$ ,  $D$  et  $J$  donnés, tel que  $0 \leq D < N$ , calcule le numéro du soldat survivant.

## 2.10 Le juste prix

Un joueur est face à un grand nombre d'objets, ayant chacun un prix. Le but du joueur est de ramener dans sa base, située à cinq mètres des objets, la plus grande valeur possible, en terme d'objets, pour grossir sa cagnotte. Par exemple, si le joueur ramène un téléviseur à € 2000, un toaster à € 30 et un laptop à € 1500, sa cagnotte s'élève à € 3530. Lorsqu'il ne transporte aucun objet, le joueur se déplace à 1,5 m/s. Lorsqu'il transporte un objet de  $x$  kg, il se déplace à  $-0,5x + 1,5$  m/s.

En supposant qu'il y a  $N$  objets et qu'on peut connaître le prix du  $i^{\text{e}}$  objet avec `getPrice(i)` et le poids de ce même objet avec `getWeight(i)`, écrivez un algorithme qui calcule la plus grosse cagnotte que le joueur pourra rassembler en  $T$  secondes.

# 3 QCM

## 3.1 Puissance « tunée »

Que vaut  $i$  après exécution de l'algorithme suivant ?

- (a)  $i = b^{\lfloor b/a \rfloor}$                       où  $\lfloor x \rfloor$  représente le plus grand entier plus petit que ou égal à  $x$   
(b)  $i = b^{\lfloor a/b \rfloor}$   
(c)  $i = b^{\lceil a/b \rceil}$                       où  $\lceil x \rceil$  représente le plus petit entier plus grand que ou égal à  $x$

---

**Algorithme 3 :** Puissance « tunée ».

---

**Input :**  $a, b$ , deux entiers strictement positifs

---

```
1   $i \leftarrow 1$ 
2  while  $a > 0$  do
3     $i \leftarrow i * b$ 
4     $a \leftarrow a - b$ 
5  return  $i$ 
```

---



### 3.2 Factorielle étrange

Complétez l'algorithme suivant qui prend en entrée un entier  $n > 0$ , pour qu'il calcule la factorielle de  $n$ , c'est-à-dire le produit  $1 \times 2 \times \dots \times n$ . Vous devez choisir une des instructions suivantes :

- (a)  $j \leftarrow j + 1$
- (b)  $j \leftarrow n$
- (c)  $j \leftarrow j + k$

---

**Algorithme 4** : Factorielle étrange.

---

**Input** :  $n$ , un entier strictement positif

**Output** : la factorielle de  $n$ , c'est-à-dire le produit  $1 \times 2 \times \dots \times n$

```
1  $i \leftarrow 1$ 
2  $j \leftarrow 0$ 
3  $k \leftarrow n + 1$ 
4 while  $k > 0$  do
5   // À compléter
6    $i \leftarrow i * k$ 
7    $k \leftarrow k - 1$ 
8 return  $i/j$ 
```

---