

Compétition Internationale d'Informatique

Feuille de Problèmes du Tour de Qualification



Remarque : Ceci est une version traduite. Cliquez sur [ici](#) pour voir la feuille de problèmes originale en anglais.

Problèmes

Les problèmes ICSC consistent en des problèmes conceptuels et des problèmes de programmation. Le type de problème est indiqué par les symboles suivants :

💡 Problème conceptuel : Nécessite un raisonnement logique ou mathématique, résoluble à la main.

🔗 Problème de programmation : Nécessite la mise en œuvre d'une solution en code.

Tu peux marquer cinq points pour chaque problème. L'évaluation de la difficulté varie d'une étoile (le plus facile) à trois étoiles (le plus difficile).

Nous avons compilé du matériel de formation pour t'aider avec les concepts pertinents pour les problèmes posés. Tu peux y accéder ici : icscompetition.org/training.

Votre Solution

Tu dois fournir une description écrite de ta solution pour **tous les problèmes** dans la feuille de solution que tu soumetts (y compris le code). Tu peux soumettre des solutions manuscrites ou dactylographiées. Pour les problèmes de programmation, merci de télécharger également tes fichiers de code. Tu peux soumettre tes solutions de codage en Python (3.9), Java (OpenJDK 11) ou C++ (GCC 15.1). Pour t'assurer que ton code est évalué correctement, merci d'utiliser le squelette de code fourni : icscompetition.org/en/#qualification.

Tu voudras peut-être consulter les exemples de solutions pour nos problèmes de formation afin de voir à quoi pourrait ressembler une soumission claire et bien structurée : https://icscompetition.org/docs/solutions/2025/example/en/ICSC_example_2025.pdf.

Informations de Soumission

Lorsque tu soumetts ta solution, un compte de participant sera créé automatiquement pour t'aider à gérer ta soumission. Tu peux accéder à ton compte et à ta soumission en utilisant les identifiants que tu as définis lors du processus de soumission : <https://icscompetition.org/login>.

Merci de vérifier que ta solution et tes fichiers de code ont été téléchargés correctement. Tout problème de lecture ou d'exécution de ton code sera signalé dans ton compte de participant. Tu dois soumettre ta solution en ligne avant **Sunday, 24 August 2025, 23h59 UTC+0** à <https://icscompetition.org/submission>.

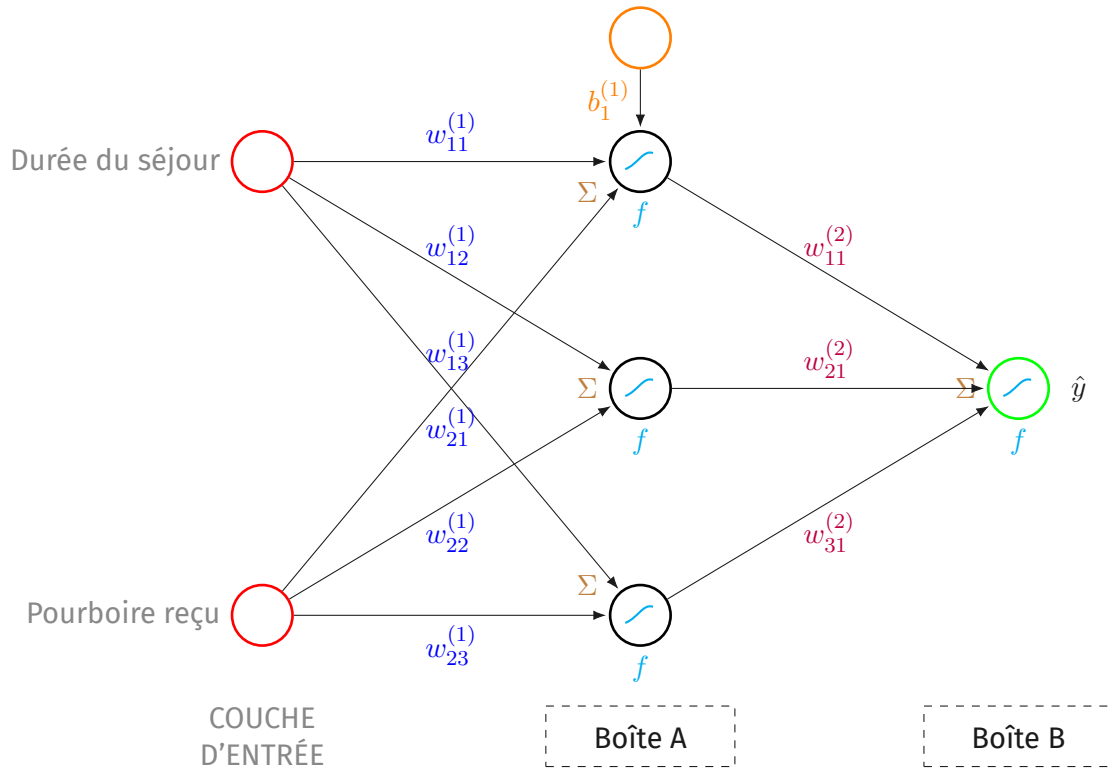
Pour te qualifier pour le tour de pré-finale, tu dois obtenir au moins 15 points (Junior), 17 points (Youth) ou 20 points (Senior). Si tu as des questions ou des commentaires, n'hésite pas à nous contacter par e-mail à l'adresse suivante : info@icscompetition.org.

Nous espérons que tu apprécieras de t'attaquer aux problèmes. Bonne chance !

🔦 Problème A : Composants de Réseau de Neurones



Considère cette illustration d'un réseau neuronal (Perceptron multicouche), qui prédit si un client de restaurant a été satisfait de sa visite :



Identifie et étiquette les composants suivants en te basant sur la figure ci-dessus :

$w_{21}^{(1)}$: Σ : f :

: : :

Boîte A: Boîte B: $\hat{y}$:

</> Problème B : Calculateur de Gâteaux

Un boulanger a une recette de gâteau préférée qui nécessite des quantités précises d'ingrédients : **100 unités de farine** et **50 unités de sucre**. Ta tâche est d'aider le boulanger à calculer :

- Combien de gâteaux peuvent être faits à partir d'un inventaire donné de farine et de sucre.
- Quelle quantité de farine et de sucre restera inutilisée après avoir fait autant de gâteaux que possible.

Écris la fonction suivante :

```
list<cake, remaining_flour, remaining_sugar> cake_calculator(flour, sugar).
```

- `flour` : Un entier supérieur à 0 spécifiant la quantité de farine disponible.
- `sugar` : Un entier supérieur à 0 spécifiant la quantité de sucre disponible.
- La fonction doit renvoyer une liste (ou un tableau) de trois entiers (`cake`, `remaining_flour`, `remaining_sugar`) avec les indices suivants : (0) le nombre de gâteaux qui peuvent être faits, (1) la quantité de farine restante, (2) la quantité de sucre restante.

Indice : Il existe plusieurs façons valides d'aborder ce problème. Pour t'aider avec le problème, nous te montrons une solution possible ci-dessous en pseudocode.¹ Tu peux utiliser le pseudocode pour ton programme, ou trouver une solution différente.

CakeCalculator

Input: Ingrédients : farine, sucre

Output: cakeCount, flourLeft, sugarLeft

```

1: flourNeeded ← 100                                /* La recette nécessite 100 de farine */
2: sugarNeeded ← 50                                  /* La recette nécessite 50 de sucre */
3: cakeCount ← 0                                     /* Nombre de gâteaux réalisés */
4: while true do
5:   if flour < flourNeeded OR sugar < sugarNeeded then
6:     break                                          /* Arrête si nous ne pouvons plus faire de gâteaux */
7:   end if
8:   flour ← flour - flourNeeded
9:   sugar ← sugar - sugarNeeded
10:  cakeCount ← cakeCount + 1
11: end while
12: flourLeft ← flour
13: sugarLeft ← sugar
14: return cakeCount, flourLeft, sugarLeft

```

¹Le pseudocode est une manière abstraite de décrire des algorithmes sans se concentrer sur les spécificités d'un langage de programmation particulier. C'est une forme d'abstraction utile. Apprenez à lire le pseudocode ici : <https://www.geeksforgeeks.org/what-is-pseudocode-a-complete-tutorial/>

💡 Problème C : L'Application de Messagerie Scolaire



Toi et tes amis avez développé une application de messagerie qui fonctionne sur le réseau de l'école. Pour transmettre des messages, l'application encode les caractères en séquences binaires (combinaisons de zéros et de uns). Cependant, en raison de la politique de données stricte de l'école, chaque élève ne dispose que d'une petite quantité de données par jour. Pour tirer le meilleur parti de cette bande passante limitée, tu souhaites mettre en œuvre une stratégie d'encodage plus efficace.

L'entropie de l'information aide à déterminer le nombre minimum théorique de bits nécessaires en moyenne pour encoder les messages. Pour un ensemble de symboles avec des probabilités $p_1, p_2, \dots, p_n$, l'entropie H est donnée par :

$$H = - \sum_{i=1}^n p_i \times \log_2(p_i)$$

Tu sais, grâce au trafic de messages précédent, que les caractères ont les fréquences suivantes :

Caractère	Probabilité	Caractère	Probabilité
A	0.20	G	0.05
B	0.15	H	0.05
C	0.12	I	0.04
D	0.10	J	0.03
E	0.08	K	0.02
F	0.06	L	0.10

Question 1 : Les encodages de texte standard utilisent le même nombre de bits pour chaque caractère. Pourquoi l'utilisation de codes de longueur différente pour les caractères ayant des probabilités différentes t'aidait-elle à transmettre plus de messages dans ta limite de données ? Donne un exemple.

Question 2 : Calcule l'entropie pour l'ensemble de 12 caractères ci-dessus. Que représente cette valeur en termes d'encodage optimal ?

La méthode de Fano est une façon de créer des codes préfixes de longueur variable efficaces.² Tu as implémenté cet algorithme et généré le code suivant pour ton ensemble de caractères :

Caractère	Probabilité	Code de Fano	Caractère	Probabilité	Code de Fano
A	0.20	000	G	0.05	001
B	0.15	100	H	0.05	1011
C	0.12	010	I	0.04	0111
D	0.10	1100	J	0.03	1101
E	0.08	0110	K	0.02	1111
F	0.06	1010	L	0.10	1110

²Des détails sur les codes de Fano sont disponibles sur : https://cs.stanford.edu/people/eroberts/courses/soco/projects/1999-00/information-theory/fano_codes_4.html

Question 3 : Calcule la longueur moyenne du code de ton code de Fano et compare-la à la limite théorique de l'entropie. Quelle est l'efficacité de ton code de Fano ?

</> Problème D : Jeu de Mots Croisés



Un jeu de mots cachés est un jeu éducatif courant dans lequel les joueurs recherchent des mots donnés cachés dans une grille de lettres. Étant donné une grille 2D de lettres, il faut trouver tous les mots valides dans cette grille. Voici un exemple d'un tel puzzle contenant les mots *apprentissage*, *science* et *amusant* :

N	P	K	N	P	I	J	W	J	G
W	L	E	A	R	N	I	N	G	D
V	T	T	S	C	I	E	N	C	E
I	T	Q	T	Y	I	P	C	S	K
B	J	Q	M	A	M	F	U	N	W
M	S	B	P	B	I	O	A	I	X
Q	G	D	D	K	P	C	I	Q	T
C	X	W	N	Z	S	Q	M	C	O
P	I	L	Q	I	P	C	L	Y	X
C	E	X	S	K	I	K	F	G	F

Ta tâche est d'implémenter une fonction qui génère un puzzle de mots cachés de **taille 10 x 10** basé sur une liste de mots d'entrée. Écris spécifiquement la fonction :

```
list<list<char>> create_crossword(list<string> words)
```

- `words` : Une liste de mots qui doivent apparaître dans le puzzle de mots cachés.
- La fonction doit renvoyer un tableau 2D (ou une liste de listes) de caractères représentant le puzzle. Chaque mot doit apparaître comme une séquence continue dans la grille.
- Aucune règle de positionnement spécifique n'est requise. Tes mots peuvent être disposés horizontalement, verticalement ou en diagonale. Sois créatif ! Les solutions exceptionnelles seront récompensées.

Indice : Vous pouvez consulter les concepts de programmation suivants avant de résoudre ce problème : tableaux (bidimensionnels), manipulation de chaînes de caractères, itérations et boucles imbriquées.

💡 Problème E : Complétude Fonctionnelle de NAND



Les portes logiques sont des éléments de base fondamentaux des ordinateurs, car elles contrôlent la façon dont l'information binaire est transmise à travers un circuit. Une porte logique effectue une fonction binaire :

$$f(a, b) \rightarrow \{0, 1\}, \quad \text{où } a, b \in \{0, 1\}.$$

Certaines portes sont particulièrement importantes en raison d'une propriété connue sous le nom de *complétude fonctionnelle*. Une porte logique (ou un ensemble de portes) est dite fonctionnellement complète si toute fonction booléenne peut être construite en utilisant uniquement cette porte (ou ces portes).

L'une de ces portes est la porte **NAND** (NON ET), qui produit 0 uniquement lorsque $a = b = 1$. Ainsi, elle produit 1 si au moins une entrée est 0.

Prouve que la porte **NAND** est fonctionnellement complète.

Indice : Peux-tu exprimer les opérations *AND*, *OR* et *NOT* en utilisant uniquement *NAND* ?