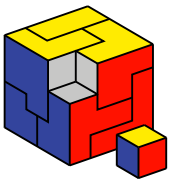
Prologin 2023



Concours national d'informatique

Épreuve écrite d'algorithmique

NIM - Ni IMproved

1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (3 heures et 30 minutes).

Conseils

— Lisez bien tout le sujet avant de commencer.

— **Soignez la présentation** de votre copie.

— N'hésitez pas à poser des questions.

— Si vous avez fini en avance, relisez bien.

— N'oubliez pas de passer une bonne journée.

Remarques

— Le barème est donné à titre indicatif uniquement.

— Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.

— Lorsqu'un algorithme est demandé, vous pouvez le décrire avec suffisamment de précision, le pseudo-coder ou l'implémenter dans le langage de votre choix. Dans le dernier cas, veuillez néanmoins préciser le langage que vous utilisez.

— Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe.

— Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.

— Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

2 À propos du sujet

Dans ce sujet écrit, nous allons étudier la création d'un nouvel éditeur de texte révolutionnaire pour coder en Befunge : « NIM — Ni Improved ». Le Befunge est un langage ésotérique se codant en deux dimensions, dans une grille « cyclique » : à droite du dernier caractère d'une ligne se trouve le premier caractère de la ligne, en dessous du premier caractère de la dernière ligne se trouve le premier caractère de la première ligne, et vice-versa. Le code est ensuite interprété par un curseur constamment redirigé et réorienté. Ainsi, le code peut être exécuté de gauche à droite, mais parfois aussi de droite à gauche, de haut en bas ou de bas en haut.

— Dans la Section 3, nous étudierons une méthode de déplacement à travers un fichier Befunge. Cette section peut rapporter un total de **13 points**.

— Dans la Section 4, nous étudierons l'implémentation d'une fonction de recherche de chaîne de caractères dans un fichier Befunge. Cette section peut rapporter un total de **10 points**.

— Dans la Section 5, nous étudierons les différentes stratégies afin de placer le dernier caractère d'un code Befunge. Cette section peut rapporter un total de **17 points**.

3 NIM Bindings

Dans cette section, nous allons étudier les déplacements dans notre éditeur de texte. Étant donné que notre éditeur de texte pourra être utilisé pour coder dans des langages ésotériques tels que Befunge, nous voulons implémenter des déplacements adaptés pour leur propriété cyclique : en se déplaçant vers la droite depuis la dernière colonne du fichier, nous revenons sur la première colonne.

Nous allons utiliser quatre touches directionnelles assez particulières :

— Une touche A permettra de se déplacer d'une case vers la droite (ce qui fait revenir sur la première colonne dans le cas où nous nous situerions sur la dernière colonne) ;

— Une touche B permettra de se déplacer d'une case vers le bas (ce qui fait revenir sur la première ligne dans le cas où nous nous situerions sur la dernière ligne) ;

— Une touche C permettra de se déplacer de X cases vers la droite (tout en conservant la propriété cyclique : cette touche a le même effet que d'appuyer X fois sur la touche A) ;

— Une dernière touche D permettra de se déplacer de Y cases vers le bas (similairement, cela a le même effet que d'appuyer Y fois sur la touche B).

3.1 Sur une ligne

Dans cette première partie, nous nous concentrerons uniquement sur l'effet de nos touches directionnelles sur une unique ligne de N caractères. Ainsi, uniquement les touches A et C nous sont d'intérêt. Supposons par exemple une ligne composée de 8 caractères, ainsi que de deux touches :

— A, « se déplacer d'une case vers la droite » ;

— C, « se déplacer de 5 cases vers la droite » ($N = 8$, et $X = 5$).

Alors, il est possible d'atteindre la 4^{e} colonne en partant de la première, en appuyant deux fois sur la touche C, et enfin en appuyant une fois sur la touche A. La Figure 1 montre ce déplacement. Lors du deuxième appui sur la touche C, on se déplace « cycliquement » de cinq cases vers la droite, depuis la 6e case. On atterit ainsi sur la 3^{e} colonne.

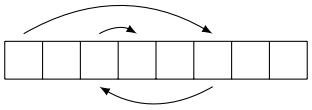


FIGURE 1 – Une manière possible d'atteindre la 4^{e} colonne, en appuyant sur les boutons C, C, puis A.

Question 1 (1 point)

> En supposant toujours une ligne composée de 8 caractères, ainsi que deux touches A « se déplacer d'une case vers la droite » et C « se déplacer de 5 cases vers la droite » ($N = 8$, et $X = 5$). Énumérer trois manières différentes d'atteindre la 5^{e} colonne depuis la première.

Question 2 (1 point)

> En supposant maintenant une ligne composée de 8 caractères, ainsi que deux touches A « se déplacer d'une case vers la droite » et C « se déplacer de 7 cases vers la droite » ($N = 8$, et $X = 7$). Quelle est la colonne la plus longue à atteindre depuis la première colonne ?

Question 3 (2 points)

> Décrire un algorithme qui, prenant en entrée deux entiers N et X, retourne un unique entier : le nombre d'appuis minimal permettant d'atteindre n'importe quelle colonne. En d'autres termes, vous devez trouver le plus petit entier K tel que pour toute colonne, vous soyez certain de pouvoir l'atteindre en K déplacements ou moins. Une complexité temporelle de $O(N)$ est attendue. [1]

Question 4 (1 point)

> Supposons toujours une ligne composée de 8 caractères, ainsi que deux touches A « se déplacer d'une case vers la droite » et C « se déplacer de X cases vers la droite » ($N = 8$). Quelles valeurs de X (inférieures à N) permettent de minimiser le nombre maximal d'appuis nécessaires afin de pouvoir atteindre n'importe quelle colonne ?

Question 5 (1 point)

> De la même manière, quelles valeurs de X (inférieures à N) permettent de minimiser le nombre maximal d'appuis nécessaires afin de pouvoir atteindre n'importe quelle colonne si $N = 16$?

Question 6 (2 points)

> Décrire un algorithme qui, prenant en entrée un entier N, retourne une valeur optimale de X permettant de minimiser le nombre maximal d'appuis nécessaires afin de pouvoir atteindre n'importe quelle colonne. Vous pouvez appeler un algorithme donné lors d'un exercice précédent.

3.2 En deux dimensions

Question 7 (2 points)

> Décrire un algorithme qui, prenant en entrée deux entiers N, M, retourne une valeur optimale de (X, Y) permettant de minimiser le nombre maximal d'appuis nécessaires afin de pouvoir atteindre n'importe quelle position dans un fichier de taille $N \times M$. Vous pouvez appeler un algorithme donné lors d'un exercice précédent.

Question 8 (3 points)

> Afin de compiler un fichier Befunge, le compilateur commence par effectuer une vérification syntaxique. Cette vérification syntaxique parcourt le fichier ligne par ligne, en partant de la première ligne, et s'arrête à la première erreur détectée. Cependant, ayant été implémentée à la va-vite, la vérification syntaxique n'affiche pas où se trouve l'erreur quand il y en a une ! Vous venez de tenter de compiler un fichier de 42 lignes, et vous remarquez lors de la compilation qu'il y a une erreur de syntaxe. Vous décidez alors de repérer la ligne fautive en tentant à plusieurs reprises de compiler certaines lignes isolées du fichier, vous indiquant si la faute se situait oui ou non dans ces lignes.
>
> Indiquer une stratégie sur les lignes à compiler pour localiser l'erreur de syntaxe en compilant le moins de fois possible le fichier, sachant que **vous ne devez en aucun cas provoquer plus de deux nouvelles erreurs de compilation**. [2]
>
> Vous ne devez pas non plus altérer le contenu du fichier. Simplement, tour à tour, sélectionnez certaines lignes et envoyez-les à la compilation. Vous saurez alors si la ligne provoquant une erreur de compilation se situe dans l'une de celles-ci.

1. Cependant, il est possible de résoudre le problème en une complexité temporelle inférieure. Un point bonus est attribué si vous y parvenez.
2. Si vous provoquez trois erreurs de compilations, la Terre explose.

4 Recherche de texte

Nous nous concentrerons ici sur l'implémentation d'une fonction de recherche dans l'éditeur de texte NIM. Cependant, cet éditeur de texte étant potentiellement utilisé pour coder dans des langages ésotériques tels que AsciiDots ou Befunge, nous voulons implémenter la fonctionnalité « rechercher du texte dans plusieurs directions ».

4.1 L'algorithme de Joseph Marchand

Dans cette partie, nous étudierons le cas d'une recherche de sous-chaîne classique, sur une ligne et sans se préoccuper des cycles. Joseph Marchand, votre collaborateur sur la création de NIM, vous propose son algorithme pour rechercher une sous-chaîne [3] :

Algorithme 1 : Recherche de sous-chaîne ?

```
Entrées : Une chaîne de caractères S, une potentielle sous-chaîne de caractères T
Résultat : Vrai si T est inclus dans S, Faux sinon
L ⟵ la longueur de S ;
M ⟵ la longueur de T ;
pour i ← 1 à L − M faire
    j ⟵ 1 ;
    tant que j < M et S[i + j − 1] = T[j] faire
        j ⟵ j + 1 ;
    fin
    si j = M alors
        retourner Vrai
    fin
fin
retourner Faux
```

Question 9 (2 points)

> Indiquer tout d'abord si l'algorithme est correct. Dans le cas échéant, expliquer la ou les erreurs, les corriger **en apportant les moindres modifications**. Il n'est pas ici question d'optimisation.

Question 10 (2 points)

> Estimer la complexité temporelle pire cas et cas moyen de cet algorithme. On supposera pour le cas moyen que les chaînes de caractères sont composées de caractères aléatoires de valeur ASCII comprise entre 32 et 126 (caractères imprimables).

Question 11 (1 point)

> Donner un exemple d'entrée qui **maximise** le temps d'exécution de son algorithme, avec $L = 9$ au maximum. L'objectif est de faire effectuer à l'algorithme le plus de comparaisons possibles.

Question 12 (1 point)

> Proposer un autre algorithme de recherche de sous-chaîne ayant une meilleure complexité pire cas.

4.2 Fonction recherche

Question 13 (2 points)

> Indiquer un algorithme qui permet de rechercher efficacement une sous-chaîne dans un fichier, horizontalement ou verticalement, à l'endroit ou à l'envers, sans prendre en compte la propriété cyclique de la grille.
>
> La Figure 2 montre ces quatres configurations différentes d'une sous-chaîne de caractères. La fonction doit retourner Vrai si la sous-chaîne est présente au moins une fois d'une quelconque manière, et Faux sinon. Vous pouvez appeler les fonctions des questions précédentes. Estimer le plus précisément possible la complexité temporelle de votre algorithme.

3. On considère ici les chaînes de caractères indicées à 1 (le premier caractère de S est $S[1]$)

S	A	X	U	T	N
M	N	I	M	W	I
U	D	R	E	P	M
X	U	A	X	M	S
M	I	N	T	I	G
U	E	J	X	N	S

FIGURE 2 – Un exemple de grille de lettres, et 4 potentielles positions pour la sous-chaîne « NIM ».

P	R	E	V	E	N
T	A	N	N	O	G
M	N	L	M	N	I
J	A	E	V	I	G
G	M	I	N	T	M
P	U	U	O	Y	I

FIGURE 3 – Un exemple de grille de lettres cyclique, et 3 potentielles positions pour la sous-chaîne « NIM ».

Question 14 (2 points)

> Dans certains langages comme Befunge, la grille est cyclique, à la manière d'un tore : une chaîne peut commencer à la fin d'une ligne et se terminer au début de la même ligne, de même pour les colonnes. La Figure 3 montre quelques configurations possibles d'une sous-chaîne de caractère. Indiquer un algorithme permettant de rechercher efficacement une sous-chaîne dans un fichier, horizontalement ou verticalement, à l'endroit ou à l'envers, et en prenant en compte la propriété cyclique du fichier. Estimer le plus précisément possible la complexité temporelle de votre algorithme.

5 NIM Online

On rappelle que l'éditeur de texte NIM se base sur l'idée qu'un fichier est une grille rectangulaire de largeur M et de longueur N, où chaque case de la grille peut contenir un caractère.

Notre nouvel éditeur de texte possède également une fonctionnalité permettant à plusieurs utilisateurs de coder sur le même fichier en même temps ! Cependant, cela peut causer divers problèmes de synchronisation si deux personnes tentent d'éditer en même temps la même partie du code. Alors, pour éviter tout problème, on a décidé que :

— Initialement, le fichier est entièrement vide. Les $N \times M$ cases du fichier ne contiennent rien.

— Il est impossible d'écrire sur une case déjà écrite, par nous ou par quelqu'un d'autre.

— Il est impossible d'écrire sur une case étant sur la même ligne et adjacente à une autre case déjà écrite, par nous ou par quelqu'un d'autre. C'est à dire qu'il est impossible d'écrire juste à droite ou juste à gauche d'un autre caractère. Cela prend en compte la propriété cyclique du fichier : si le dernier caractère d'une ligne est écrit, alors il est impossible d'écrire sur le premier caractère de cette ligne, et vice-versa.

Vous êtes connecté avec Joseph Marchand pour terminer votre TP Befunge sur NIM Online. Votre professeur est très strict concernant la répartition du travail, et tient absolument à ce que tous les membres du groupe participent autant au TP. Alors, vous avez décidé que vous écrirez, chacun votre tour, un caractère. Cependant, vous avez tous les deux remarqué que le professeur avait tendance à mieux noter celui ayant modifié en dernier le fichier. Vous voulez donc tous les deux écrire le dernier caractère à tout prix.

La Figure 4 montre un exemple de déroulé du TP pour un fichier de taille $(N, M) = (2, 5)$. Dans cet exemple, Joseph Marchand écrit le premier caractère, et vous écrivez le dernier caractère. Vous remportez donc le bonus.

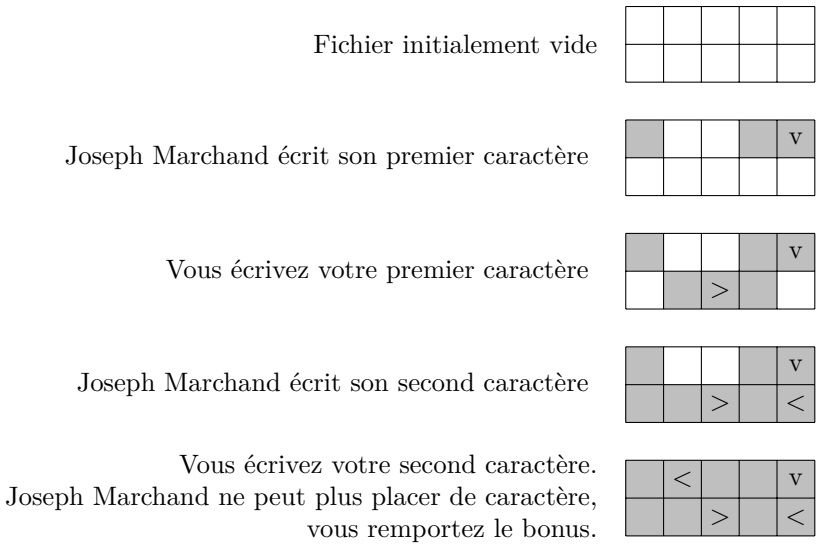


FIGURE 4 – Exemple de déroulé du TP. Les cases grisées correspondent aux cases où il est impossible décrire.

5.1 One-liner

Dans cette partie, vous avez décidé de tenter de réaliser l'entièreté de votre TP sur une unique ligne, principalement par vantardise. On fixe donc $N = 1$ pour l'entièreté des questions de cette partie.

Question 15 (1 point)

> Supposons que la ligne fasse 10 caractères ($M = 10$). Afin d'obtenir le bonus, est-il préférable d'écrire le premier caractère ou de laisser Joseph Marchand écrire en premier ? Justifier en détaillant votre stratégie.

Question 16 (1 point)

> Supposons maintenant que la ligne fasse 11 caractères ($M = 11$). Afin d'obtenir le bonus, est-il préférable d'écrire le premier caractère ou de laisser Joseph Marchand écrire en premier ? Justifier en détaillant votre stratégie.

Question 17 (2 points)

> Pour chaque largeur (M) allant de 1 à 9, indiquer sans justifier s'il est préférable de commencer à écrire ou d'attendre que Joseph Marchand écrive le premier caractère.

5.2 Analyse de Sprague-Grundy

On peut affecter à chaque situation possible un *nombre de Grundy*, définit comme suit :

— Le nombre de Grundy d'une situation finale, où personne ne peut plus rajouter de caractère, est 0.

— Le nombre de Grundy de toute autre situation donnée est **le plus petit entier (positif ou nul) qui n'apparaît pas dans la liste des nombres de Grundy des situations qui suivent immédiatement la situation donnée.**

Par exemple, s'il existe trois coups possibles depuis une certaine situation, menant à trois potentielles situations différentes A, B et C, que le nombre de Grundy de la situation A est 0, le nombre de Grundy de la situation B est 2 et que le nombre de Grundy de la situation C est 3, alors le nombre de Grundy de notre situation sera 1.

Dans cette partie, nous étudions toujours des fichiers composés d'une unique ligne ($N = 1$). **Cependant, nous allons ici ignorer la propriété cyclique du fichier**. Il est donc ici temporairement possible de poser un caractère dans la première colonne d'une ligne malgré la présence d'un caractère dans la dernière colonne, et vice-versa.

La Figure 5 montre alors les nombres de Grundy de toutes les situations pour un fichier de taille $(N, M) = (1, 3)$.

— Les trois situations représentées en feuille sont des situations finales car personne ne peut plus rajouter de caractère. Leur nombre de Grundy est donc bien 0.

— Les deux situations qui ne peuvent mener qu'à une feuille au coup d'après doivent posséder comme nombre de Grundy le plus petit entier postifi ou nul n'étant pas le nombre de Grundy de la feuille (0), c'est à dire 1.

— Enfin, le fichier vierge possède un nombre de Grundy de 2, car on peut directement atteindre depuis cette situation trois autres situations ayant comme nombre de Grundy 1, 0 et 1. Le plus petit entier (positif ou nul) n'appartenant pas à ces nombres est bien 2.

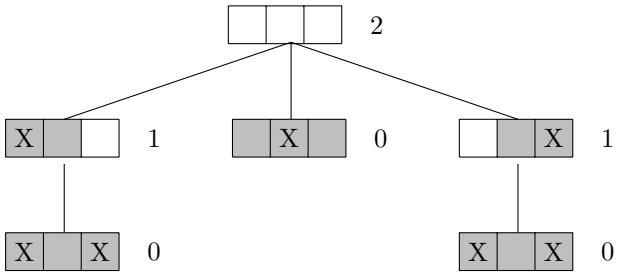


FIGURE 5 – Les nombres de Grundy de toutes les situations pour $(N, M) = (1, 3)$

Question 18 (2 points)

> Indiquer le nombre de Grundy de toutes les situations possibles pour un fichier de largeur $M = 4$. On décrira une situation sous la forme `X__X`, où `X` désigne une case contenant un caractère et `_` désigne une case libre.

Situation	Nombre de Grundy
`____`	
`___X`	
`__X_`	
`_X__`	
`_X_X`	
`X___`	
`X__X`	
`X_X_`	

Question 19 (1 point)

> Justifier le fait qu'une situation est gagnante pour la personne qui doit écrire son caractère si et seulement si le nombre de Grundy de la situation est différent de 0.

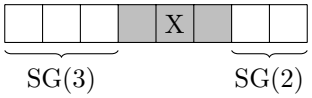


FIGURE 6 – Division d'une situation en *somme* de deux situations plus petites

On appelle SG la fonction qui donne pour chaque entier M le nombre de Grundy de la situation initiale d'un fichier de taille $1 \times M$, sans prendre en compte la propriété cyclique du fichier. Le *Théorème de Sprague-Grundy* indique que si une situation peut être décrite comme la somme de deux situations plus petites, alors le nombre de Grundy de cette situation est le OU EXCLUSIF [4] ($\oplus$) des nombres de Grundy des deux situations plus petites.

Par exemple, la Figure 6 montre une situation pouvant être considérée comme la somme de deux situations initiales, avec $M = 3$ et $M = 2$. En effet, jouer sur ce plateau de 7 cases avec un caractère déjà posé est strictement similaire à jouer sur deux plateaux vides de 3 cases et de 2 cases. Alors, le théorème de *Sprague-Grundy* indique que le nombre de Grundy de ce plateau est le OU EXCLUSIF des nombres de Grundy des deux plateaux vides de 3 cases et de 2 cases. Autrement dit, $G = SG(3) \oplus SG(2)$.

Question 20 (2 points)

> Indiquer sans justifier les valeurs de $SG(M)$, M allant de 1 à 10. Vérifier que cela coïncide avec les gagnants trouvés précédemment.

Question 21 (2 points)

> Étant donné un entier N, écrire un algorithme calculant la valeur de $SG(M)$. Vous obtiendrez la totalité des points si la complexité temporelle de votre algorithme est $O(M^2)$ ou moins. [5]

5.3 Maintenant, pour de vrai.

On suppose maintenant un fichier de taille carrée, donc de taille $N \times N$. On prend désormais en considération la propriété cyclique du fichier. Il n'est donc pas possible d'avoir un caractère au début et à la fin d'une même ligne en même temps.

Question 22 (1 point)

> Déterminer, sans justifier, s'il est avantageux d'écrire le premier caractère pour $N = 1$ jusqu'à $N = 4$.

Question 23 (3 points)

> Décrire une stratégie gagnante si N est pair.

Question 24 (2 points)

> On définit cette fois-ci $G(N)$ comme étant le nombre de Grundy de la situation initiale d'un fichier de taille $N \times N$, en prenant en considération la propriété cyclique de la grille. Justifier que l'on a alors :
>
> $$G(N) = \begin{cases} 0 & \text{si } N \text{ est pair} \\ 1 & \text{si } N = 1 \\ 0 & \text{si } N \text{ est impair, } N \geq 3 \text{ et } SG(N-3) \neq 0 \\ 1 & \text{si } N \text{ est impair, } N \geq 3 \text{ et } SG(N-3) = 0 \end{cases}$$
>
> En déduire pour tout $N \leq 10$ s'il est avantageux d'écrire le premier caractère.

4. Le OU EXCLUSIF de deux nombres A et B est le nombre ayant ses bits à 1 là où A ou B a son bit à 1, mais pas les deux. Par exemple, $12 \oplus 10 = 1100_2 \oplus 1010_2 = 0110_2 = 6$.

5. On rajoute un point bonus si vous trouvez une solution en $O(1)$. C'est possible.

6 Bonus

Question bonus 25 (1 point)

> Faites le strict opposé de ce qui suit, puis cochez la case 2.
> *Vous ne devez en aucune circonstance ne pas éviter de ne pas cocher uniquement les cases non paires*
>
> ☐ 1
> ☐ 2
> ☐ 3
> ☐ 4

Question bonus 26 (1 point)

> Laissez un carreau de plus que la marge pour la totalité des questions, sauf pour la question 19.