Prologin 2023

Concours national d'informatique

Épreuve écrite d'algorithmique

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 (∼15 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. **Ne dégradez votre copie sous aucun prétexte**.

— 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

L'apéro, c'est le moment de la journée où l'on se détend, où l'on partage des moments entre amis et où l'on s'amuse. Mais pour vous et votre meilleur ami Joseph Marchand, algorithmiciens passionnés, c'est bien plus que cela.

Vous avez invité Joseph Marchand à une soirée, mais au lieu de vous détendre, vous vous obstinez à trouver les solutions optimales à **tous** les jeux d'apéro. Des apérocubes aux pizzas en passant par la préparation des cocktails, vous ne laissez aucune question sans réponse.

— Dans la Section 3, vous vous affronterez sur une série de questions que vous trouverez en déballant des *Apérocubes*, un concept totalement révolutionnaire de cubes de fromages venant avec des énigmes algorithmiques. Cette section peut rapporter un total de **10 points**.

— Dans la Section 4, vous vous affronterez pour concocter les meilleurs cocktails possibles. Cette section peut rapporter un total de **10 points**.

— Dans la Section 5, après avoir érigé une tour de pizzas digne de la construction des lycéens du lycée Saint-Louis de Gonzague lors de la première demi-finale à Paris, vous entamez votre meilleure dégustation. Vous allez enfin pouvoir régler vos comptes une bonne fois pour toutes en vous assurant que Joseph Marchand se retrouve avec la dernière part de pizza (l'humiliation finale) ! Cette section peut rapporter un total de **20 points**.

Alors, prêts à relever le défi ? Joseph espère que vous avez apporté votre appétit et vos meilleures cartes algorithmiques pour ce duel acharné au sommet. Bon courage !

3 Apérocubes

Dans cette section, vous allez tour à tour piocher un *Apérocube*, lire le défi algorithmique proposé et le relever. Une fois sur deux, vous devrez résoudre le défi proposé. Vous obtiendrez 1 point pour avoir résolu le défi de manière efficace. Vous obtiendrez une fraction du point pour avoir résolu le défi de manière inefficace, ou si votre solution comporte une ou plusieurs erreurs.

Une fois sur deux, vous devrez vérifier la validité de la solution de Joseph Marchand. Vous obtiendrez un demi point pour avoir repéré la faille dans la solution de Joseph Marchand et apporté un contre-exemple à son algorithme, et un demi point pour avoir proposé une correction valide. Le correctif doit être une simple modification de l'algorithme de Joseph Marchand et non un tout nouvel algorithme.

Question 1 (1 point)

> Étant donné un entier relatif, écrivez un algorithme déterminant si cet entier est pair ou impair. Vous ne devez cependant pas utiliser ni de division, ni l'opérateur *quotient* ou *reste* (aussi appelé modulo).

Question 2 (1 point)

> Étant donné un entier relatif, écrivez un algorithme déterminant s'il existe un 4 dans la représentation décimale de ce nombre. Par exemple, il existe un 4 dans la représentation décimale de 42 et de -444, mais pas dans celle de 69 ni celle de -2898.
>
> Voici la solution proposée par Joseph Marchand :
>
> **Algorithme 1 :** Détection d'un 4 dans la représentation décimale d'un entier relatif.
>
> **Entrées :** Un entier relatif N.
> **Résultat :** Vrai s'il existe un 4 dans la représentation décimale de N. Faux sinon.
> **tant que** $N \neq 0$ *et* $N \not\equiv 4 \mod 10$ **faire**
> $N \longleftarrow \lfloor N/10 \rfloor$;
> **fin**
> **retourner** $N \neq 0$
>
> Prouvez que son algorithme est incorrect en apportant un contre-exemple, puis proposez un correctif.

Question 3 (1 point)

> Étant donné un entier relatif, écrivez une procédure qui appelle une unique fois la fonction `pika` si ce nombre est égal à 42, ou la fonction `kapi` si le nombre est différent de 42. Vous ne devez cependant pas utiliser la structure conditionnelle `si`. Uniquement les `tant que` et les assignations de variables sont autorisées.

Question 4 (1 point)

> Étant donné un tableau de N entiers relatifs non vide, écrivez une procédure en place qui effectue une rotation gauche d'ordre 3 sur le tableau. Par exemple, le tableau [3, 1, 4, 1, 5, 9, 2, 6, 5, 3], après appel de la procédure, doit devenir [1, 5, 9, 2, 6, 5, 3, 3, 1, 4].
>
> Voici la solution proposée par Joseph Marchand : [1]
>
> **Algorithme 2 :** Rotation gauche en place d'ordre 3.
>
> **Entrées :** Un tableau A non vide de N entiers relatifs.
> **Résultat :** Une rotation gauche d'ordre 3 est effectuée sur le tableau.
> $i \longleftarrow 0$;
> **répéter** $N-1$ **fois**
> $j \longleftarrow (i+3)\%N$;
> $x \longleftarrow A_i$;
> $A_i \longleftarrow A_j$;
> $A_j \longleftarrow x$;
> $i \longleftarrow j$;
> **fin**
>
> Prouvez que son algorithme est incorrect en apportant un contre-exemple, puis proposez un correctif.

Question 5 (1 point)

> Étant donné un entier naturel N, écrivez un algorithme qui retourne la liste de tous les nombres premiers inférieurs ou égaux à N. Cependant, le seul opérateur autorisé est l'addition. Il est donc interdit de soustraire, multiplier, diviser ou d'effectuer un modulo.

Question 6 (1 point)

> Étant donné un tableau de N entiers relatifs, déterminez si ce tableau contient une unique fois tous les entiers de 1 à N.
>
> Voici la solution proposée par Joseph Marchand :[2]
>
> Si vous le souhaitez, vous pouvez utiliser la fonction $H(x)$, retournant un entier, ayant la propriété qu'il soit très difficile de trouver un entier x tel que $H(x) = k$ ou $H(x) = H(k)$.
>
> **Algorithme 3 :** Détection de la présence des nombres de 1 à N dans un tableau
>
> **Entrées :** Un tableau A non vide de N entiers relatifs.
> **Résultat :** Vrai si le tableau contient une unique fois tous les entiers de 1 à N, Faux sinon.
> $S_1 \longleftarrow 0$;
> $S_2 \longleftarrow 0$;
> **pour** $i \leftarrow 1$ **à** N **faire**
> | $S_1 \longleftarrow S_1 \oplus i$;
> | $S_2 \longleftarrow S_2 \oplus A_i$;
> **fin**
> **retourner** $S_1 = S_2$;
>
> Prouvez que son algorithme est incorrect en apportant un contre-exemple, puis proposez un correctif.

Question 7 (1 point)

> La suite de Fibonacci est une suite où chaque terme est la somme des deux nombres précédents : $1, 1, 2, 3, 5, 8, 11, \ldots$
>
> Étant donné un entier naturel N, écrivez un algorithme qui retourne le N-ième nombre de la suite de Fibonacci. On considère que le premier nombre de Fibonacci est 1, donc la fonction doit par exemple retourner 2 pour $N = 3$. Cependant, votre algorithme ne doit pas utiliser plus de deux variables supplémentaires (qui doivent être des entiers). Il est autorisé de modifier N. Assigner à deux variables simultanément est interdit.

Question 8 (1 point)

> Étant donné un tableau de N entiers relatifs, écrivez une procédure qui inverse le tableau. Par exemple, après application de la procédure, le tableau $[3,\ 1,\ 4,\ 1,\ 5,\ 9]$ doit devenir $[9,\ 5,\ 1,\ 4,\ 1,\ 3]$. Cependant, la procédure ne doit pas utiliser plus d'une variable supplémentaire, qui doit être un entier.
>
> Voici la solution proposée par Joseph Marchand :[3]
>
> **Algorithme 4 :** Inversion d'un tableau en n'utilisant qu'une variable supplémentaire.
>
> **Entrées :** Un tableau A non vide de N entiers relatifs.
> **Résultat :** Le tableau est inversé.
> **pour** $i \leftarrow 0$ **à** $\left\lfloor \frac{N-1}{2} \right\rfloor$ **faire**
> | $A_i \longleftarrow A_i \oplus A_{N-i-1}$;
> | $A_{N-i-1} \longleftarrow A_{N-i-1} \oplus A_i$;
> | $A_i \longleftarrow A_i \oplus A_{N-i-1}$;
> **fin**
>
> Prouvez que son algorithme est incorrect en apportant un contre-exemple, puis proposez un correctif.

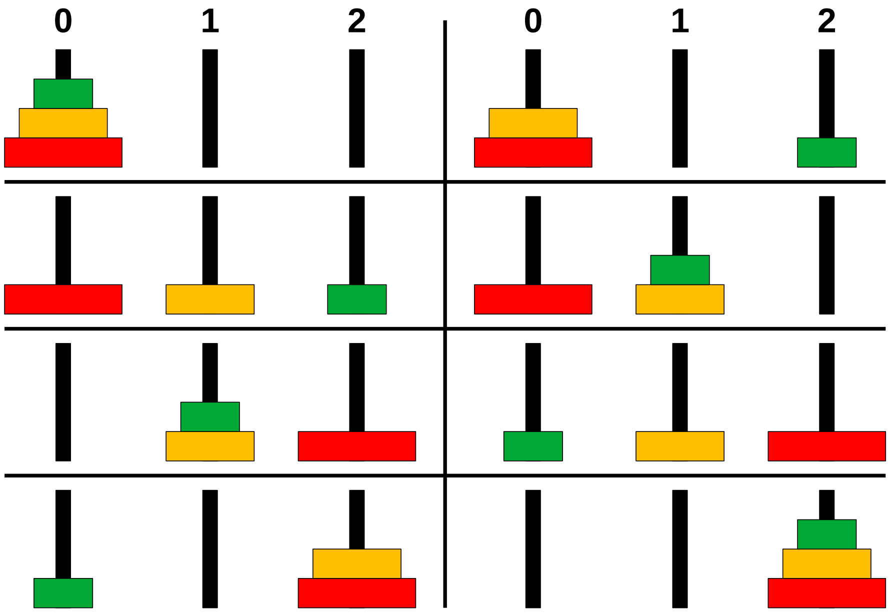
1. Ici, le tableau A est indicé à 0. Le premier élément est donc A_0 et le dernier A_{N-1}.
2. Ici, le tableau A est indicé à 1. Le premier élément est donc A_1 et le dernier A_N.
3. Ici, le tableau A est indicé à 0. Le premier élément est donc A_0 et le dernier A_{N-1}.

Question 9 (1 point)

Les *tours de Hanoï* sont un jeu consistant à déplacer des disques de différents diamètres d'une colonne à une autre, en respectant deux conditions :

— On ne peut déplacer qu'un disque à la fois,

— On ne doit jamais placer un disque sur un disque plus petit.



Cette image montre la liste des mouvements afin de déplacer trois disques de la colonne 0 à la colonne 2.

Écrivez une fonction `hanoï`, prenant en paramètre trois entiers N, A et B, qui déplace les N plus petits disques des tours de Hanoï de la colonne A à la colonne B. Vous pouvez appeler la fonction `déplace`, qui prend en paramètre deux entiers A et B, qui permet de déplacer un disque de la colonne A à la colonne B. Les colonnes sont numérotées de 0 à 2.

Cependant, vous ne devez pas utiliser de variable supplémentaire. Les valeurs de N, A et B sont également en lecture seule. Vous ne pouvez pas les modifier.

Question 10 (1 point)

Étant donné un tableau de N entiers naturels, écrivez une fonction qui détermine s'il est possible de partitionner le tableau en deux sous-tableaux de même somme.

Le tableau $[3,\ 1,\ 4,\ 1,\ 5]$ peut par exemple être divisé en $[3,\ 4]$ et $[1,\ 1,\ 5]$, de somme 7. Le tableau $[3,\ 1,\ 4,\ 1]$, quant à lui, ne peut pas être partitionné en deux sous-tableaux de même somme.

Voici la solution proposée par Joseph Marchand : [4]

Algorithme 5 : Détermination de l'existence d'une bipartition de même somme.

Entrées : Un tableau A non vide de N entiers naturels.
Résultat : Vrai s'il est possible de partitionner le tableau en deux sous-tableaux de même somme, Faux sinon.

`tri_rapide`(A) ;
$\Delta \longleftarrow 0$;
pour $i \leftarrow N$ **à** 1 **faire**
  si $\Delta \geq 0$ **alors**
    $\Delta \longleftarrow \Delta - A_i$;
  fin
  sinon
    $\Delta \longleftarrow \Delta + A_i$;
  fin
fin
retourner $\Delta = 0$

Prouvez que son algorithme est incorrect en apportant un contre-exemple. Il n'est ici exceptionnellement pas demandé d'apporter un correctif, car il n'existe aucun correctif simple.

4 Cocktails

Sans même voir le temps passer, vous avez déjà terminé toute la boîte d'Apérocubes. C'est maintenant l'heure de préparer vos meilleurs cocktails.

Un cocktail est un ensemble d'ingrédients. Deux cocktails sont distincts s'il existe un ingrédient présent dans l'un des deux cocktails mais pas dans l'autre.

4.1 Deux cocktails

Ici, votre objectif va être de concocter deux cocktails. Vous allez devoir répartir **tous** les ingrédients entre les deux cocktails. Chaque ingrédient doit être présent dans l'un des deux cocktails, mais pas dans les deux.

Cependant, certains mélanges ne sont pas désirables...

Question 11 (1 point)

> Pour réaliser vos cocktails, vous disposez ici de ces dix ingrédients :
> — Aperol
> — Cognac
> — Desperados
> — Gin
> — Kirsch
> — Pastis
> — Ricard
> — Tequila
> — Vodka
> — Whiskey
>
> Votre objectif est de concocter deux cocktails en utilisant tous les ingrédients, chaque ingrédient ne pouvant être utilisé que pour l'un des deux cocktails.
>
> Il est cependant interdit d'avoir dans un même cocktail :
> — Aperol et Kirsch
> — Aperol et Whiskey
> — Cognac et Desperados
> — Desperados et Gin
> — Pastis et Tequila
> — Tequila et Aperol
> — Tequila et Ricard
> — Vodka et Cognac
> — Vodka et Gin
>
> Indiquez les ingrédients de vos deux cocktails de telle sorte que tous les ingrédients soient utilisés dans précisément l'un des deux cocktails.

Question 12 (1 point)

> Indiquez précisément le nombre de solutions distinctes à la question précédente. On considère que deux solutions sont distinctes si un cocktail d'une solution n'est pas présente dans les cocktails de l'autre solution.

Question 13 (1 point)

> Représentez les contraintes sur les ingrédients à l'aide d'un graphe. Représentez également sur ce graphe votre solution à la question 11, à la manière de votre choix.

4. Ici, le tableau A est indicé à 1. Le premier élément est donc A_1 et le dernier A_N.

Question 14 (2 points)

Étant donné une liste d'ingrédients et la liste des incompatibilités entre les ingrédients, écrivez un algorithme qui retourne Vrai s'il est possible de répartir tous les ingrédients en deux cocktails, Faux sinon.

La liste des ingrédients est $[1,\ 2,\ 3,\ 4,\ \ldots,\ N]$. La liste des incompatibilités est donnée sous la forme d'une liste de paires $(A,\ B)$ avec A et B deux ingrédients (deux nombres entre 1 et N).

Vous pouvez, si vous le souhaitez, compléter cet algorithme :

Algorithme 6 : Détermination de la bipartition d'un graphe.

Entrées : Un entier N, une liste de paires d'entiers I.
Résultat : Vrai s'il est possible de répartir les ingrédients en deux cocktails, Faux sinon.

```
// Construction de la liste d'adjacence
Adj est un tableau de N listes, initialement vides;
pour chaque paire (A, B) dans I faire
    Ajouter A à Adj_B ;
    Ajouter B à Adj_A ;
fin

C est un tableau de N éléments, initialement 0;
fonction Coloration(i)



fin

pour i ← 1 à N faire



fin
```

Question 15 (1 point)

Étant donné une liste d'ingrédients et la liste des incompatibilités entre les ingrédients, écrivez un algorithme qui retourne le nombre de manières possibles de répartir tous les ingrédients en deux cocktails. Deux solutions sont considérées distinctes s'il existe un cocktail dans l'une des solutions qui n'est pas présent dans l'autre.

La liste des ingrédients est $[1,\ 2,\ 3,\ 4,\ \ldots,\ N]$. La liste des incompatibilités est donnée sous la forme d'une liste de paires $(A,\ B)$ avec A et B deux ingrédients (deux nombres entre 1 et N).

Vous pouvez, si vous le souhaitez, compléter cet algorithme :

Algorithme 7 : Détermination du nombre de bipartitions d'un graphe.

Entrées : Un entier N, une liste de paires d'entiers I.

Résultat : Vrai s'il est possible de répartir les ingrédients en deux cocktails, Faux sinon.

`// Construction de la liste d'adjacence`

`Adj` est un tableau de N listes, initialement vides ;

pour chaque *paire* $(A, B) \in I$ **faire**

  Ajouter A à $\mathtt{Adj}_B$;

  Ajouter B à $\mathtt{Adj}_A$;

fin

C est un tableau de N éléments, initialement 0 ;

fonction `Coloration`(i)

fin

$R \longleftarrow 1$;

pour $i \leftarrow 1$ **à** N **faire**

fin

retourner R ;

4.2 Un cocktail

Finalement, voyant que Joseph Marchand prépare lui aussi deux cocktails, vous décidez de changer de stratégie. L'objectif ici va être de réaliser un seul cocktail, en utilisant le plus d'ingrédients possibles. En effet, plus un cocktail contient d'ingrédients, meilleur il est (excepté lorsque ce cocktail contient une paire d'ingrédients incompatibles).

Question 16 (1 point)

> En utilisant les ingrédients et les contraintes sur les ingrédients décrites dans la question 11, combien d'ingrédients au maximum pouvez-vous mettre dans un cocktail ? Décrire un cocktail ayant le nombre maximum d'ingrédients.

Question 17 (1 point)

> Indiquez précisément le nombre de solutions distinctes à la question précédente. On considère que deux solutions sont distinctes si un ingrédient est présent dans le cocktail d'une solution mais n'est pas présent dans le cocktail de l'autre solution.

Question 18 (2 points)

> Le *Problème de la clique maximum* consiste à trouver une *clique* de cardinal maximal dans un graphe. Une *clique* d'un graphe est un sous-ensemble des sommets de ce graphe dont le sous-graphe induit est complet.
>
> Il est connu que le problème de la clique maximum est NP-difficile. Prouvez qu'il est impossible de trouver le meilleur cocktail en un temps polynomial. [5]

5. On se permet de supposer que $P \neq NP$

5 Tour de Pizzas

On sonne à la porte. C'est Dominologin's Pizza qui est arrivé avec la commande de pizzas ! Vous ouvrez toutes les boîtes (il y en a beaucoup), et construisez ensemble une énorme tour de pizzas avec l'intégralité des parts. Cette tour de pizzas prend la forme d'un arbre : le sommet est la racine de l'arbre. Une propriété intéressante de cet arbre est que toutes les parts de pizzas reposent toujours sur 0, 1, ou 3 autres parts de pizzas.

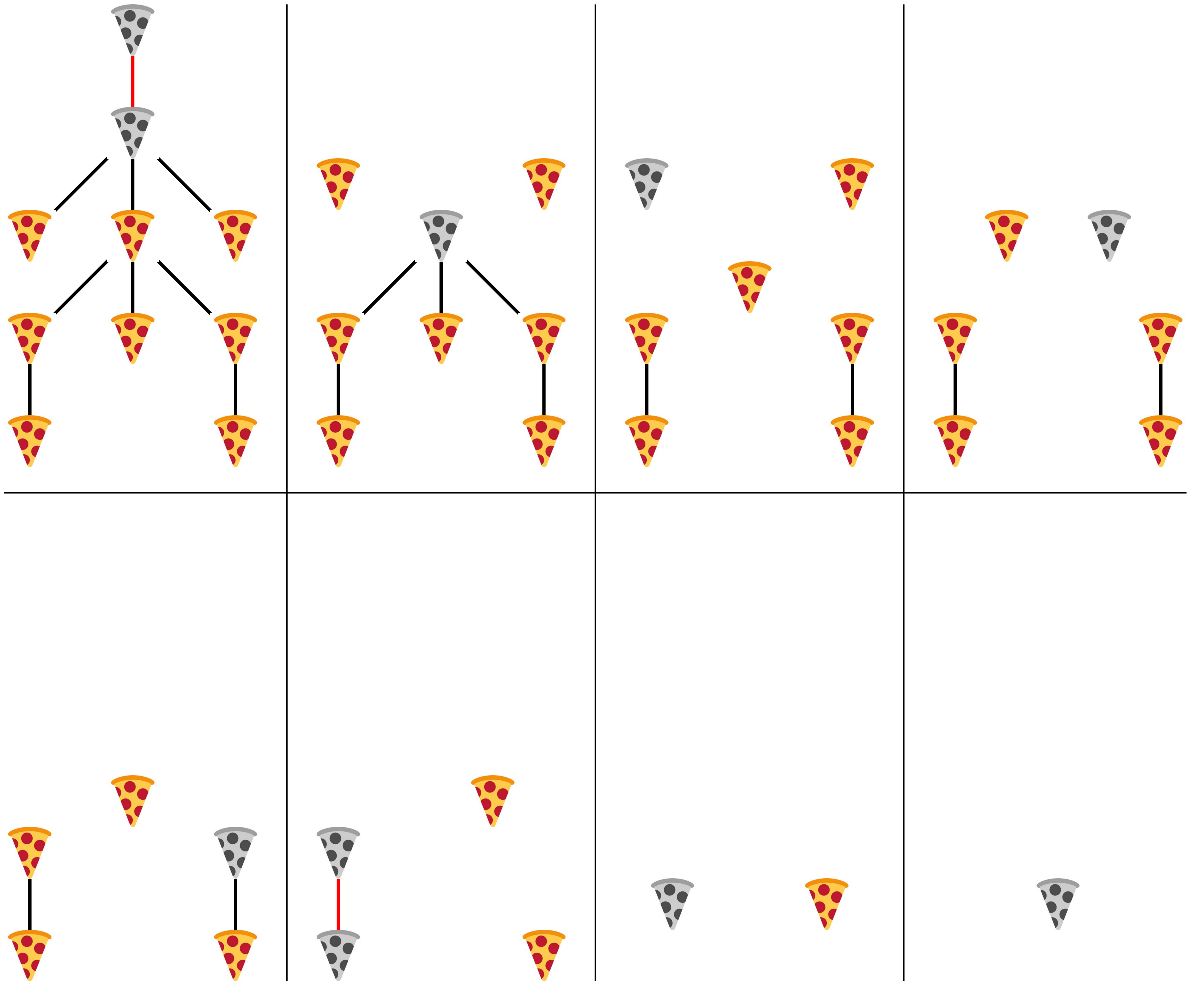
Tour à tour, vous commencez à manger les parts au sommet de la tour de pizzas d'une manière bien précise :

- Soit vous ne mangez **que** la part de pizza au sommet de la tour,
- Soit vous mangez la part de pizza au sommet de la tour **et** toutes les parts de pizzas sur lesquelles elle repose, c'est à dire ses 1 ou 3 « fils ».

À noter qu'après quelques tours, il est possible que l'on se retrouve avec plusieurs tours de pizzas séparées (une forêt). Dans ce cas, vous effectuerez l'une de ces deux actions sur **une** des tours, au choix.

Au fur et à mesure que la dégustation avance, vous commencez à penser au redoutable événement : Qui va manger la dernière part ? Depuis des générations, manger la dernière part de pizza est reconnu comme un crime international, prohibé même par la convention de Genève !

Votre objectif va donc être de faire tout votre possible pour forcer Joseph à consommer la dernière part de pizza. Attention cependant, il en fera de même !



Cette image montre un exemple de déroulé de la consommation de la tour de pizza. Supposons que vous commencez la dégustation.

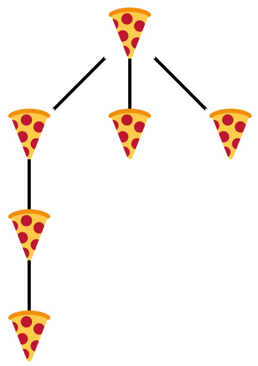
- Dans la première case, vous décidez de consommer la pizza au sommet de la tour ainsi que la part de pizza sur laquelle elle repose.
- Dans la deuxième case, Joseph Marchand se retrouve devant trois tours de pizzas. Il décide alors de consommer uniquement la pizza au sommet de la tour centrale.
- Dans la troisième case, vous avez le choix parmi cinq différentes tours de pizzas. Vous décidez de consommer une pizza isolée.

— Dans la quatrième case, Joseph Marchand se retrouve devant quatre différentes tours de pizzas. Il décide à son tour de consommer une pizza isolée.
— Dans la cinquième case, vous vous retrouvez devant deux tours de pizzas composées de deux pizzas chacunes, et d'une tour de pizza composée d'une unique pizza. Vous décidez de consommer uniquement la pizza au sommet d'une tour de pizzas composée de deux pizzas.
— Joseph Marchand se retrouve alors dans la sixième case, devant une tour de pizzas composée de deux pizzas et deux pizzas isolées. Il décide de consommer la pizza au sommet de la grande tour restante, ainsi que son unique fils sur lequel il repose.
— Il ne reste plus que deux pizzas isolées dans la septième case, et c'est à vous de vous servir. Vous n'avez pas d'autre choix que de consommer une part de pizza isolée.
— Joseph Marchand se retrouve alors avec la dernière part de pizza. Quelle honte ! Vous remportez la victoire.

5.1 Mieux tomber...

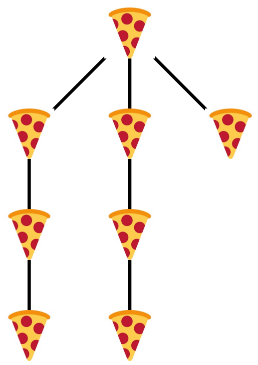
Dans cette première partie, nous allons au contraire étudier comment s'assurer de prendre la dernière part. Ceci est évidemment à titre éducatif uniquement, merci d'éviter de commettre une telle inhumanité dans la vraie vie, même sous la torture.

Question 19 (1 point)



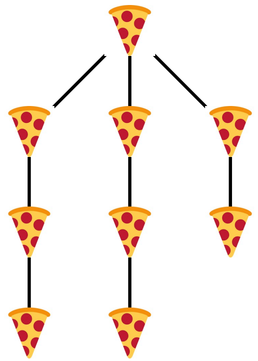
En supposant que la configuration initiale était celle-ci, afin d'être certain de consommer la dernière part de pizza, est-il préférable de commencer ou de laisser Joseph entamer la dégustation ? Détailler votre stratégie.

Question 20 (1 point)



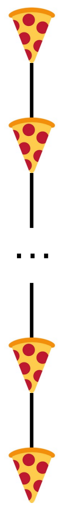
En supposant que la configuration initiale était celle-ci, afin d'être certain de consommer la dernière part de pizza, est-il préférable de commencer ou de laisser Joseph entamer la dégustation ? Détailler votre stratégie.

Question 21 (1 point)



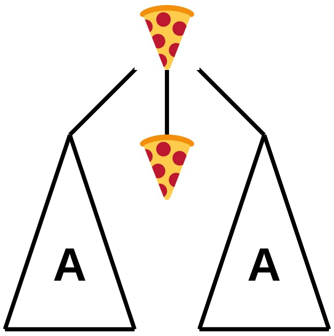
En supposant que la configuration initiale était celle-ci, afin d'être certain de consommer la dernière part de pizza, est-il préférable de commencer ou de laisser Joseph entamer la dégustation ? Détailler votre stratégie.

Question 22 (2 points)



En supposant que la configuration initiale est un arbre filiforme composé de N parts de pizzas. Indiquer en fonction de N quand faut-il commencer ou quand faut-il laisser Joseph Marchand commencer, et détailler votre stratégie pour réussir à vous emparer de la dernière part de pizza.

Question 23 (2 points)



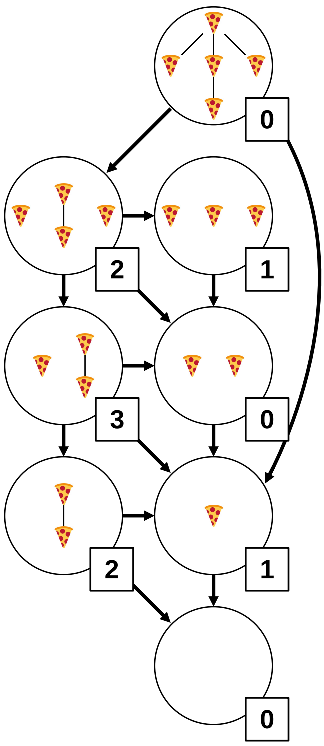
En supposant maintenant que la configuration devant laquelle vous vous trouvez suive le motif de cette image, c'est-à-dire une pizza reposant sur une pizza unique et deux tours de pizzas strictement similaires, indiquer une stratégie pour vous assurer la dernière part de pizza.

5.2 Nombres de Grundy

Les *nombres de Grundy* sont des nombres très utiles pour analyser ce genre de jeux. Ils sont définis de la sorte :

— Le nombre de Grundy d'une situation finale (dans notre cas la situation sans aucune pizza restante) est toujours 0.

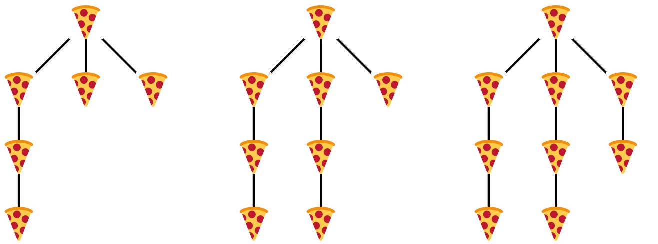
— Le nombre de Grundy de toute autre situation est **le plus petit entier positif ou nul qui n'est pas présent dans l'ensemble des nombres de Grundy des situations qui la succèdent immédiatement**.



Cette image représente les nombres de Grundy de 8 différentes situations. Les flèches représentent les possibilités de successions des situations. La situation vide, en bas de l'image, est l'unique situation finale. Son nombre de Grundy est donc de 0. Pour toutes les autres situations, leur nombre de Grundy est le plus petit entier positif ou nul qui n'apparaît pas dans la liste des nombres de Grundy de leurs successeurs immédiats. Pour la première situation, en haut de l'image, les nombres de Grundy de ses successeurs sont 2 et 1. Son nombre de Grundy est donc 0.

Question 24 (2 points)

Déterminer, sans justifier, le nombre de Grundy de ces trois situations :



Question 25 (2 points)

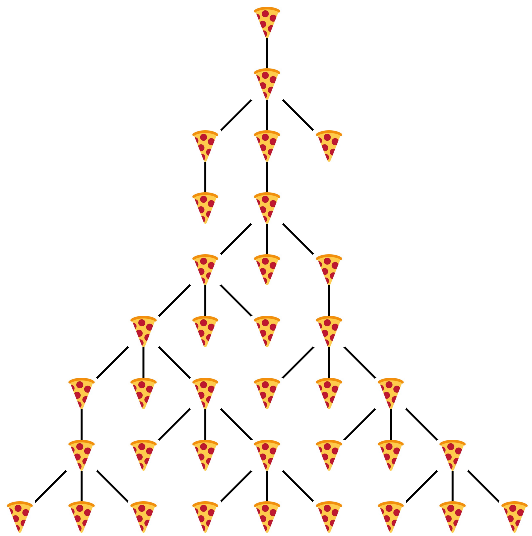
Par ailleurs, on définit une situation comme « gagnante » pour la personne qui s'apprête à choisir ses pizzas s'il existe une stratégie permettant de s'assurer l'obtention de la dernière part de pizza depuis cette situation. [6]

Prouvez qu'une situation est gagnante si et seulement si son nombre de Grundy est différent de 0. Aussi, justifier le fait que le nombre de Grundy de la situation initiale (avec une seule tour) ne peut pas être supérieur à 2.

Question 26 (2 points)

Enfin, le théorème de Sprague-Grundy affirme que le nombre de Grundy d'une somme de plusieurs situations (dans notre cas plusieurs tours) est simplement le OU Exclusif des nombres de Grundy de ces situations.

En utilisant ce fait (ou pas, mais bon courage), déterminez le nombre de Grundy de cette situation, déterminez s'il est avantageux de commencer, et indiquez le coup optimal :



6. Cependant, on peut considérer la situation actuelle comme perdante, puisque vous avez perdu au Jeu.

Question 27 (2 points)

Écrivez un algorithme déterminant le nombre de Grundy d'une tour de pizzas. Une tour de pizzas est donnée sous la forme d'une structure `Tour` : une liste de `Tour`s décrivant les tours de pizzas sous la pizza supérieure.

Une pizza isolée est donc représentée par une liste vide. Une tour de deux pizzas l'une sur l'autre est donc représentée par une liste contenant une unique `Tour` : une liste vide. La tour de pizzas représentée dans la question 19 ressemblerait donc à `[[[[ ]]], [ ], [ ]]`.

Vous pouvez vous aider de ce début d'algorithme si nécessaire. De plus, vous pouvez supposer qu'un cache existe sur la fonction `Grundy`, ainsi si la fonction est appelée deux fois sur la même tour, alors le résultat sera immédiatement retourné sans refaire de calcul. Il n'est donc pas nécessaire d'effectuer de la programmation dynamique.

Algorithme 8 : Détermination du nombre de Grundy d'une tour

Entrées : Une `Tour`, T

Résultat : Le nombre de Grundy de la situation contenant uniquement cette tour.

```
fonction Mex(S)
    // Retourne le plus petit entier positif ou nul n'apparaîssant pas dans S.
    i ⟵ 0;
    tant que i ∈ S faire
        i ⟵ i + 1;
    fin
    retourner i
fin
```

5.3 ...pour mieux se relever !

Question 28 (5 points)

Détaillez une stratégie afin de s'assurer, si possible, que Joseph Marchand prenne la dernière part, et non vous.

Il est très fortement conseillé d'utiliser des concepts vus dans la section précédente.

6 Bonus

Question bonus 29 (1 point)

Comptez le nombre de dessins de pizzas présents sur le sujet. Si ce nombre est pair, percez un trou de précisément 5 millimètres de diamètre dans le coin inférieur droit de la première page de votre copie. Si ce nombre est un multiple de trois, effectuez une légère déchirure dans le coin supérieur droit de la première page de votre copie. Sinon, si ce nombre n'est ni pair ni un multiple de trois, déchirez un triangle dans le coin supérieur droit de la dernière page de votre copie. Enfin, repérez l'unique dessin de pizza différent des autres pizzas. Décrivez sa position en réponse à cette question, sans utiliser de nombres.

Question bonus 30 (2 points)

Comptez le nombre de fois que le terme « pizza » ou « pizzas » est présent sur le sujet. Ne tentez surtout pas le bonus 1. Il ne rapporte pas de point s'il est effectué. Aucune pizza n'est différente des autres, elles sont toutes identiques. Rendez votre copie en bon état. Ne dégradez surtout pas votre copie, cela applique un malus (c'est indiqué dans le préambule). Dessinez simplement votre meilleure tour de pizzas en réponse à cette question. Ne faites rien de particulier avec le nombre de fois que le terme « pizza » ou « pizzas » est présent sur le sujet. Ce n'est pas la peine d'écrire ce nombre sur votre copie, le correcteur n'a pas pris la peine de compter.

7 Annexe

Vous trouverez ici les définitions de quelques notions employées dans le sujet.

Ensembles usuels En mathématiques, on peut classifier les nombres en divers ensembles. Voici les deux ensembles de nombres utilisés dans ce sujet.

— L'ensemble $\mathbb{N}$ des nombres dits *naturels* comprend tous les entiers positifs ou nuls. 0, 1, et 14 sont des entiers naturels. -1 et 3.14 ne sont pas des entiers naturels.

— L'ensemble $\mathbb{Z}$ des nombres dits *relatifs* comprend tous les entiers, positifs, nuls ou négatifs. $0, 1, 14, -1$ et -25 sont des entiers relatifs. 0.609 et -6.18 ne sont pas des entiers relatifs.

Opérations bit à bit En informatique, il est courant de représenter un nombre sous sa forme *binaire* plutôt que décimale. La forme binaire d'un nombre est une manière de le représenter avec seulement deux chiffres, 0 et 1, plutôt que les dix chiffres 0-9 utilisés en décimal.

En décimal, le *poids* du dernier chiffre d'un nombre est 1, l'avant dernier 10, ensuite 100, etc. Le nombre 4321 en décimal vaut donc $4 \times 1000 + 3 \times 100 + 2 \times 10 + 1 \times 1$.

En binaire, le *poids* du dernier chiffre d'un nombre est 1, l'avant dernier 2, ensuite 4, ensuite 8, etc. Le nombre 101010_2 en binaire vaut donc $1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$, soit 42 en décimal. Le 2 en indice indique qu'un nombre est représenté sous sa forme binaire. Les chiffres d'un nombre sous sa forme binaire sont souvent appelés *bit*s.

Plusieurs opérateurs se basent sur la représentation binaire d'un nombre. On en citera notemment trois qui peuvent être utilisés tout au long du sujet :

— L'opérateur *ET*, de symbole $\cdot$ ou &, où chaque bit du résultat vaut 1 uniquement si les deux bits de même poids des deux opérandes valent également 1. Par exemple, $21\&57 = 17$ car $10101_2\&111001_2 = 1001_2$. Dans ce sujet, nous préférerons utiliser le symbole & pour éviter toute confusion avec la multiplication.

— L'opérateur *OU*, de symbole + ou |, où chaque bit du résultat vaut 1 si au moins l'un des deux bits de même poids des deux opérandes vaut 1. Par exemple, $21|57 = 61$ car $10101_2|111001_2 = 111101_2$. Dans ce sujet, nous préférerons utiliser le symbole | pour éviter toute confusion avec l'addition.

— L'opérateur *OU Exclusif*, de symbole $\oplus$ ou ^, où chaque bit du résultat vaut 1 uniquement si précisément l'un des deux bits de même poids des deux opérandes vaut 1. Par exemple, $21 \oplus 57 = 44$ car $10101_2 \oplus 111001_2 = 101100_2$. Dans ce sujet, nous préférerons utiliser le symbole $\oplus$ pour éviter toute confusion avec l'exponentiation.

Arrondi Un arrondi mathématique permet de convertir n'importe quel nombre décimal en un entier proche. Dans ce sujet, trois fonctions peuvent être utilisées pour réaliser un arrondi :

— L'arrondi inférieur, $\lfloor x \rfloor$, retournera le plus grand entier relatif inférieur ou égal à x. $\lfloor 3.94 \rfloor = 3$, $\lfloor -3.14 \rfloor = -4$, et $\lfloor 5 \rfloor = 5$.

— L'arrondi supérieur, $\lceil x \rceil$, retournera le plus petit entier relatif supérieur ou égal à x. $\lceil 3.14 \rceil = 4$, $\lceil -3.94 \rceil = -3$, et $\lceil 5 \rceil = 5$.

— La fonction *partie entière*, $E(x)$, retournera la partie entière d'un nombre. $E(3.94) = 3$, $E(-3.94) = -3$, et $E(5) = 5$.

Arithmétique modulaire Lorsque l'on effectue une *division Euclidienne* entre deux entiers, nous pouvons en extraire du résultat le *quotient* et le *reste*. Par exemple, lorsque l'on effectue la division Euclidienne de 11 par 4, on trouve un quotient de 2 et un reste de 3, car $11 = 4 \times 2 + 3$.

On peut donc se permettre d'utiliser ces opérateurs :

— La fonction quotient, de symbole //, retourne le quotient mathématique de la division Euclidienne entre deux nombres. Par exemple, $57//10 = 5$ (car $57 = 10 \times 5 + 7$), et $(-57)//10 = -6$ (car $-57 = 10 \times (-6) + 3$).

— La fonction reste, de symbole %, retourne le reste mathématique de la division Euclidienne entre deux nombres. Par exemple, $57\%10 = 7$ (car $57 = 10 \times 5 + 7$), et $(-57)\%10 = 3$ (car $-57 = 10 \times (-6) + 3$). Notez que $a\%b$ sera toujours compris entre 0 et $b - 1$, et que l'on a toujours $a = b \times (a//b) + (a\%b)$.

— Enfin, la congruence modulaire permet d'évaluer si deux nombres ont le même reste. $a \equiv b \mod n$ indique que $a\%n = b\%n$. On a donc $22 \equiv 42 \mod 10$ car $22\%10 = 42\%10 = 2$. Au contraire, $a \not\equiv b \mod n$ indique que $a\%n \neq b\%n$. On a donc $22 \not\equiv 42 \mod 11$ car $22\%11 = 0$ et $42\%11 = 9$.

Théorie des nombres

— On dit qu'un entier naturel est *premier* s'il ne possède que deux diviseurs distincts : 1 et lui-même. 1 n'est pas considéré comme premier. Les premiers nombres premiers sont donc $2, 3, 5, 7, 11, 13, 17, \ldots$