Séance d'exercice

CPUMons

08 novembre 2022

Avant de commencer... Ce document comporte plusieurs exercices dont la difficulté est renseignée par un nombre et est *globalement* croissante. Il est évidemment fortement recommandé de réfléchir au préalable sur papier avant de se lancer dans la partie implémentation.
Répartition de la difficulté :

- Niveau 1 : problème facile, ne nécessite pas de code complexe pour être mis en œuvre ;
- Niveau 2 : problème moyen, demande une réflexion plus poussée sur la présentation du code ;
- Niveau 3 : problème délicat, demande une réflexion poussée sur le problème en soi. (Niveau maximal que le test de novembre de progra 1 peut atteindre)
- Niveau 4 : problème difficile.

1 Exercice 1 (Niveau 1.5)

Oh non ! Le Compte Prétendu urbaniste Mais Obligatoirement Non Sceptique a mis du désordre dans sa chambre et il ne retrouve plus ses affaires !

Heureusement chaque objet de sa chambre est étiquète, par exemple tous ses livres sont étiquète "livres" et tous ses jouets "jouets".

Le Compte fait appel a vos service pour l'aider !

Sa chambre est divisée en case et chaque case contient un objet avec une étiquette. Nous représenterons la chambre sous forme d'une matrice carrée. Votre but est d'écrire un algorithme /find/ qui en prenant la matrice "chambre" et une étiquette "étiquette" en entrée va donner la position de tous les objets de ce type s'il y en a.

```
input : [
          ["A", "A", "B", "D"],
          ["C", "P", "U", "A"],
          ["M", "O", "N", "S"],
          ["A", "B", "D", "A"]
        ]
```

```
+---+---+---+---+
| A | A | B | D |
+---+---+---+---+
| C + P + U + A |
+---+---+---+---+
| M + O + N + S |
+---+---+---+---+
| A + B + D + A |
+---+---+---+---+
```

Si "étiquette" vaut "A" alors votre algorithme donnera comme output : [(0, 0), (0, 1), (1, 3), (3, 0), (3, 3)]
Si "étiquette" vaut "B" : [(0, 2), (3, 1)]
Si "étiquette" vaut "Chocolat" : []

2 Critique de l'exploitation animale dans les champs de salsifis. (Niveau 2)

Votre aide est requise par la Corporation des Producteurs et Utilisateurs Mexicains d'Ornithorynques pour Nourrir des Salsifis. En effet, cette noble entreprise étudie l'efficacité des ornithorynques dans l'entretient des champs de salsifis au Mexique. La capacité de cet animal pour entretenir les champs de salsifis est, il faut l'avouer, fortement sous-estimée. De nombreuses études ont montré que des ornithorynques, bien positionnées dans un champ de salsifis, entretenaient parfaitement tous les salsifis proches d'eux.

Vous avez été embauché par la C.P.U.M.O.N.S. pour leur fournir un algorithme leur permettant d'évaluer le nombre de plans de salsifis dont s'occupe un ensemble donné d'ornithorynques.

Le champ est représenté par une grille carrée dont on vous donne la longueur du coté, N. Sur cette grille sont placés M ornithorynques ($0 \leqslant M < N$). Sachant qu'un ornithorynque s'occupe de tous les plans de salsifis sur les 8 cases adjacentes à sa position, vous devez calculer le nombre de plans de salsifis qui peuvent être entretenus au maximum. Il est à noter que, le salsifis étant une plante très fragile, si plus d'un ornithorynque s'occupe d'un plan donné, celui-ci sera trop nourri et mourra malheureusement. (A noter, bien entendu, qu'une case contenant un ornithorynque ne peut pas contenir de plan de salsifis.)

Entrée :

— Sur la première ligne, se trouvent les deux entiers N et M ;

— Les M lignes suivantes contiennent chacune deux entiers représentant la position des M ornithorynques (on suppose toutes les positions distinctes).

Sortie :

— Le nombre maximal de plans de salsifis qui peuvent être entretenus par les ornithorynques.

Remarque : Afin de rendre le problème un peu plus court, on suppose que les positions des ornithorynques sont toutes telles que les 8 cases adjacentes existent (donc qu'il n'y a pas d'ornithorynque en bord de champ).

3 Une histoire déjà vue... (Niveau 2)

Dans une petite ville, comme il en existe tant d'autres, s'était implantée une gigantesque chocolaterie qui faisait vivre toute la région par son ampleur et son développement. En effet, la conjoncture économique courante avait été mise à mal par la pénultième crise financière survenue deux ans auparavant, mais qui n'avait été que doux euphémisme par rapport à celle de l'année suivante qui, bien qu'antithétique par son origine, avait fait de la ville le palliatif apanage de la misère ; tel un abject homuncule [1] devant le méprisable, mais inexorable zéphyr de la vie... Voici donc le contexte, somme toute commun mais malheureusement compréhensible, dans lequel s'était implantée cette grande chocolaterie : la Chocolaterie Préférée de l'Union des Médiocres Ouvriers Nouvellement Sous-employés.

Un jour, le PDG annonça qu'un grand concours était lancé afin qu'un enfant puisse visiter la C.P.U.M.O.N.S. [2]. Pour remporter le concours, il fallait trouver un ticket doré caché dans une des barres chocolatées de la société. Cependant, le PDG ajouta un indice : "Le ticket doré a été caché par mes soins dans une barre de chocolat, mais pas n'importe quelle barre ! La somme récursive des chiffres de son numéro de série est la réponse à la vie, l'univers et tout le reste ! Bonne chance pour le trouver !"

Intrigué par le terme de "somme récursive", vous apprenez que c'est un calcul assez simple à faire : après avoir fait la somme de tous les chiffres du numéro de série, si le nombre obtenu a strictement plus de 2 chiffres, vous faites la somme récursive du nombre obtenu. Jusqu'à obtenir un nombre avec, au plus, deux chiffres.

Vous décidez donc d'écrire un petit programme qui, étant donné le numéro de série d'une barre de chocolat, vous informe si cette barre a des chances d'être gagnante ou non...

Challenge (niveau 3) : Obtenir une complexité dans le pire des cas en $O(n)$ où n est le nombre de chiffres du numéro de série.

1. Ce mot étant la seule raison pour laquelle ce texte n'est pas en vers.
2. Aucun lien de parenté avec votre kot-associatif préféré.

4 Exercice 4 (Niveau 1)

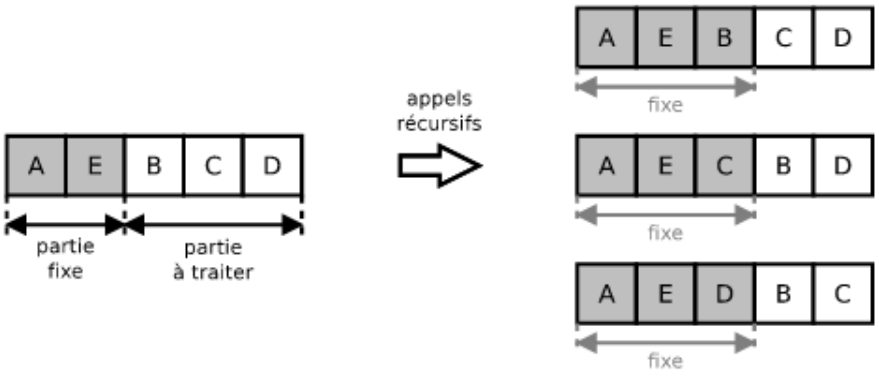
Vous gérez un site internet où des gens doivent s'inscrire avec leur date de naissance qu'ils tapent eux même.
Certaines personnes tentent de mettre de fausse dates de naissance tel que le 32/13/1234.
Vous décidez donc de créer une fonction qui vérifie si une date existe.
Remarque : on ne prend pas en compte les exceptions dans les années bissextiles.

5 Exercice 5 (Niveau 3)

L'objectif de cette question est la conception et l'implémentation en python d'un algorithme permettant de générer toutes les permutations d'une séquence de N caractères différents. Prenons, par exemple, la séquence de 3 caractères ABC. Il existe 6 permutations différentes de cette séquence : ABC, ACB, BAC, BCA, CAB et CBA.

Voici comment un tel algorithme pourrait procéder récursivement. L'algorithme considère que la séquence de caractères à traiter est composée de deux parties : la partie de gauche qui ne peut plus être changée (déjà traitée) et la partie de droite qui doit encore être permutée. Cette situation est illustrée ci-dessous. La chaîne initiale est ABCDE. Lors de l'appel actuel de l'algorithme, les deux premiers caractères de la séquence sont fixés. Il reste donc à générer les permutations qui commencent par AE et qui se terminent par toutes les permutations possibles de BCD.

L'algorithme doit récursivement générer toutes les permutations de la partie de droite. Pour cela, il prend un caractère restant dans la partie de droite et il ajoute ce caractère à la partie fixe de gauche. Il s'appelle ensuite récursivement pour la partie de droite diminuée du caratère fixé. L'exemple de la figure illustre le fait que 3 appels récursifs sont effectués. Par exemple, le premier de ces appels va générer les chaînes commençant par AEB et se terminant par les permutations de CD.



Exercice : Créer une fonction qui retourne une liste contenant toutes les permutations possibles pour une chaine de caractères. La fonction prend une chaine de caractères en argument ainsi qu'un index (l'indice à laquelle la partie fixe s'arrête).
Petit conseil : Créer une fonction qui retourne le nombre de combinaisons possibles pour une liste de taille N pour vérifier que vous avez fait toutes les combinaisons.

6 Exercice 6 (Niveau 1)

Le but de cet exercice est d'implémenter une fonction permettant de parcourir une chaîne de caractères et d'afficher tous les éléments se trouvant entre des délimiteurs successifs. Par exemple, prenons la chaîne de caractères suivante « -cochon-panda-unau-merle-orque-narval-singe- » et le caractère « - » comme délimiteur ; une telle fonction affichera :
cochon
panda
unau
merle
orque
narval
singe

Question : Donnez le code python d'une fonction *tokenizer(chaine,delimiteur)*. Le paramètre chaine représente la chaîne de caractères à parcourir. Le paramètre delimiteur représente quant à lui le délimiteur à utiliser. Cette fonction vous affichera les éléments compris entre deux délimiteurs successifs. L'usage de la fonction split() est interdit.
Veuillez envisager les cas particuliers suivants :

- La fonction de doit pas afficher de lignes vides si un ou plusieurs délimiteurs se suivent.
- Pensez au cas où la chaine de caractères ne se termine pas ou ne commence pas par un délimiteur. Que devrait retourner cette fonction sur les chaines « -chat-chien » ou « chat-chien- » avec « - » comme délimiteur ?

7 Exercice 7 (Niveau 3)

Pleins de petits tas de sables sont disposé sur une table quadrillée, allant de 0 à 3 grains de sable. Lorsqu'on rajoute un grain de sable à un tas de 3, il s'écroule dans les 4 tas d'à côté. Évidemment si le grain tombe dans un tas de 3 alors cela se répète.
Si un tas s'écroule au bord de la table alors les grains tombent en dehors de la table.
Veuillez créer un algorithme *add_sand()* qui prends en paramètre la position d'ajout et qui modifie la table comme demander jusqu'à ce qu'il n'y ait plus de tas de 4 grains sur la table.

Exemple : Exemple 1 :

```
mat = [[3, 3, 2],
       [2, 1, 0],
       [0, 1, 2]]

add_sand(1, 0)

# passera par:
# [[4, 0, 3],
#  [2, 2, 0],
#  [0, 1, 2]]

>>>[[0, 1, 3],
    [3, 2, 0],
    [0, 1, 2]]
```

Exemple 2 :

```
mat = [[0, 0, 3, 0, 0],
       [0, 3, 3, 3, 0],
       [0, 0, 3, 0, 0]]

add_and(2, 1)

Passera par :
[[0, 0, 4, 0, 0],
 [0, 4, 0, 4, 0],
 [0, 0, 4, 0, 0]]
Ensuite :
[[0, 2, 0, 2, 0],
 [1, 0, 4, 0, 1],
 [0, 2, 0, 2, 0]]

>>[[0, 2, 1, 2, 0],
   [1, 1, 0, 1, 1],
   [0, 2, 1, 2, 0]]
```

8 Concours

Pour ceux qui veulent s'entraîner pour participer aux concours, voici quelques liens pour vous permettre de travailler sur ce qui vous intéresse à votre propre rythme :

1. FranceIOI : nécessite de débloquer les premiers niveaux, mais recouvre une très grande variété d'algorithmes et de problèmes différents rangés par thématique ;
2. Isograd : sur ce site, vous retrouverez de nombreux anciens concours dont, en particulier, les BattleDev précédentes qui constituent un *excellent* point de départ dans le monde des concours ;
3. Google Code Jam : les énoncés des années précédentes y sont disponibles. Le niveau requis est, bien évidemment, progressif et les problèmes sont en général assez intéressants ;
4. Kattis : site reprenant de *très* nombreux problèmes. Il vous est conseillé de tester un ou deux problèmes "triviaux" afin de bien vérifier si vous n'avez pas de problèmes avec les entrées et sorties ; puis, de passer aux faciles et, rapidement, aux moyens (les difficiles portant très bien leur nom).