Devoir d'informatique : Gendarmes et Voleurs

Consignes

- Le devoir se fera sur copie double uniquement.
- Le numéro de chaque exercice et de chaque question devront être indiqués sur votre copie.
- Les indentations devront correctement figurer sur votre copie. Vous pourrez par exemple tracer une barre verticale.
- Pour chaque fonction vous donnerez au plus une ligne de commentaire permettant de spécifier votre fonction.

1 Introduction

Le jeu des gendarmes et des voleurs est bien connu dans les cours de récréation. Les gendarmes doivent attraper les voleurs. La partie se termine quand il n'y a plus de voleurs ou que les gendarmes sont épuisés.

Nous allons travailler avec une règle revisitée de ce jeu. Nous disposons d'une liste `GV` de caractères, chaque élément de la liste est soit `'G'` pour gendarme soit `'V'` pour voleur. Chaque gendarme peut attraper au plus un voleur qui doit être à une distance maximale notée `k` (entier).
Exemples :

```
GV1=['G', 'V', 'V', 'G', 'V']
GV2=['V', 'V', 'G', 'G', 'V', 'G']
GV3=['G', 'V', 'G', 'V', 'V', 'G']
```

2 Force brute

Une première approche est de lister toutes les solutions possibles et d'évaluer la solution de taille maximale. Pour GV1 et k=1, les 2 solutions sont, sous forme de couple d'indices, [[(0,1),(2,3)],[(0,1),(3,4)]].

Question 1 *Pour la liste GV2 et k=2, proposer les différentes solutions possibles sous forme de liste de couple d'indices.*

Question 2 *Écrire une fonction* `associe(tab:list,i:int,j:int)->bool:` *qui teste si l'élément à l'indice* `i` *de tab est un gendarme* `'G'` *et si l'élément à l'indice* `j` *est un voleur* `'V'`*, si c'est le cas les éléments aux indices* `i` *et* `j` *sont passés à* `None` *et la fonction renvoie* `True`*. Sinon, rien n'est modifié et la fonction renvoie* `False`*. les indices* `i` *et* `j` *sont pris tel que i>j ou j>i.*

Question 3 *Supposons que la liste* `GV` *de taille n possède* `m` *'G' avec m<n. Pour une distance donnée k>0, quel serait dans le pire des cas le nombre d'appel à la fonction* `associe(tab:list,i:int,j:int)` *si on veut toutes les solutions possibles.*

3 Algorithme glouton

Une solution plus efficace est d'utiliser un algorithme glouton. Pour cela, nous devons définir une propriété de choix local.

On note que la distance entre deux éléments de la liste est la valeur absolue de la différence des indices.

3.1 Le plus proche voleur

Règle : pour chaque gendarme, en commençant le traitement de la liste par la gauche, lui associer le voleur le plus proche possible.

On donne la fonction suivante :

```
def lePlusProche(tab:list,k:int)->int: # ligne 1
    res=0
    for i in range(len(tab)):
        rep=False
        if tab[i]=='G':
            j=1
            while j<k+1 and rep==False:
                if (i-j>=0 and associe(tab,i,i-j)) or (i+j<len(tab) and associe(tab,i,i+j)):
                    rep=True
                    res+=1
                j=j+1
    return res # ligne 12
```

Question 4 *Commenter les différentes lignes de 1 à 12 de la fonction* `lePlusProche(tab:list,k:int)->int:` *qui prend en argument la liste* `tab` *d'éléments* `'G'` *et* `'V'` *et* `k` *la distance maximale entre 'G' et 'V' (k>0). Est-ce que* `tab` *est modifiée ?*

Question 5 *Rédiger la signature de la fonction.*

Question 6 *Que renvoient* `lePlusProche(GV1,1)` *et* `lePlusProche(GV2,2)` *? Conclure. Qu'affiche print (GV1) après l'appel à* `lePlusProche(GV1,1)` *?*

Question 7 *Donner la définition du variant de boucle. Quel est le variant de la boucle* `while` *de la fonction* `lePlusProche(tab,k)` *? Justifier votre réponse.*

3.2 Le plus éloigné voleur possible

Règle : pour chaque gendarme, en commençant le traitement de la liste par la gauche, lui associer le voleur le plus éloigné possible.

Question 8 *A partir de la fonction* `lePlusProche(tab,k)`*, écrire une fonction* `lePlusEloigne(tab:list,k:int)->int:` *qui prend en argument la liste* `tab` *d'éléments* `'G'` *et* `'V'` *et* `k` *la distance maximale entre 'G' et 'V' (k>0). Cette fonction renvoie le nombre de voleurs attrapés. Mettre en évidence les lignes de code différentes entre les deux fonctions.*

Question 9 *Que renvoient* `lePlusEloigne(GV2,2)` *et* `lePlusEloigne(GV3,3)` *? Conclure.*

4 Approches itérative et récursive

Afin de résoudre le problème posé, nous allons séparer les gendarmes 'G' et les voleurs 'V' dans deux listes que nous pourrons ensuite comparer avec la distance k.

Question 10 *Écrire une fonction* `indicesGV(tab:list)->tuple`*, qui prend en argument la liste des 'G' et des 'V' et renvoie deux listes,* `indices_G` *la liste des indices des éléments 'G' et* `indices_V` *la liste des indices des éléments 'V'. Par exemple,* `indicesGV(['G', 'V', 'V', 'G', 'V'])` *renvoie* `([0, 3], [1, 2, 4])`.

On peut parcourir les deux listes `indices_G` et `indices_V` de gauche à droite. Si le premier indice de `indices_G` soit `indices_G [0]` est assez "proche" du premier indice de `indices_V` soit `indices_V [0]`, alors le voleur est attrapé, sinon soit le gendarme est trop loin (à droite) et on passe au voleur suivant soit le voleur est trop loin (à droite) et on passe au gendarme suivant.

Exemple : On reprend les deux listes créées `indices_G=[0, 3]` et `indices_V=[1, 2, 4]` et k=1 ;
- `indices_G [0]`=0 et `indices_V [0]`=1, ils sont assez proches, le gendarme peut attraper le voleur ;
- On passe aux indices suivants et couplesGV=1 ;

- `indices_G [1]`=3 et `indices_V [1]`=2, ils sont assez proches, le gendarme peut attraper le voleur et couplesGV=2 ;
- Toute la liste `indices_G` a été lue, la fonction renvoie le résultat : 2

Question 11 *Écrire une fonction* `gendarmeVoleurIte(ind_G :list ,ind_V :list ,k:int)->int`, *qui prend en argument la liste des indices de 'G' et la liste des indices de 'V' obtenues avec la fonction* `indicesGV(tab:list)` *ainsi que k la distance maximale et qui renvoie le nombre de voleurs attrapés.*

Question 12 *Écrire une fonction* `gendarmeVoleurRec(ind_G :list ,ind_V :list ,k:int)->int`, *qui prend en argument la liste des indices de 'G' et la liste des indices de 'V' obtenues avec la fonction* `indicesGV(tab:list)` *ainsi que k la distance maximale et qui renvoie le nombre de voleurs attrapés par une méthode récursive. Préciser les arguments qui varient dans la récursion.*