INFO-F103 – Algorithmique I
Projet 1 - Trouvez les tous

Youcef Bouharaoua
youcef.bouharaoua@ulb.be

Année académique 2021-2022

Trouvez les tous : LITE

Vous avez à votre disposition:

- **Une phrase valide (en français) sans espaces**, par exemple :"*Ceciestunephrasevalide*".
- **Une liste de mots (dictionnaire)**, par exemple "*Ceci*","*une*","*phrase*","*est*","*valide*".

À partir de ces deux éléments, déterminez toutes les façons possibles de décomposer la phrase en mots simples appartenant à la liste de mots.

Nous vous demandons donc d'écrire un petit programme Python qui utilise une chaîne de caractères et un dictionnaire de mots stocké dans un ensemble. Dans ce programme vous devez ajouter des espaces dans la chaîne de caractères pour construire des phrases où chaque mot appartient au set. Chaque mot de peut être utilisé plus d'une fois. Affichez (sur le terminal) toutes les phrases possibles. C'est à dire les phrases qu'il est possible de former en conservant l'ordre des mots dans la phrase initiale (sans espaces).

Pour mener à bien cette tâche, nous vous demandons d'implémenter deux fonctions Python:

- *diviser_phrase(string,n,resultat)* Cette fonction récursive permettra de construire et d'imprimer (afficher sur le terminal) toutes les phrases qu'il est possible de former à partir de la chaîne de caractères initiale. Nous commençons à scanner la phrase en partant de la gauche. Lorsqu'un mot valide (appartenant au set) est trouvé, vous devez vérifier si le reste de la phrase peut contenir des mots valides ou non. Voici à quoi ressemblera le premier appel à votre fonction *diviser_phrase(phrase_initiale_sans_espaces,len(phrase_initiale_sans_espaces),"")*. C'est donc dans le paramètre résultat qu'on construira chacune des phrases solutions.
- *toutes_les_phrases(phrase)* qui est une fonction d'une ligne, cette ligne consiste en l'appel de la fonction *diviser_phrase(string,n,resultat)* avec les

paramètres adéquats. La fonction *toutes_les_phrases(phrase)* prend en paramètres la phrase sans espaces.

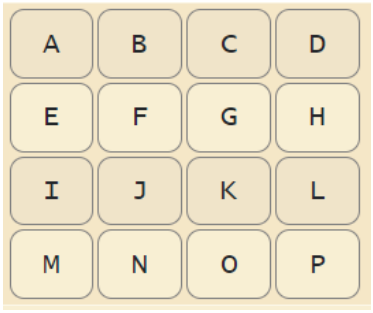
Ces deux fonctions seront implémentées dans le fichier *trouvez_les_tous_lite.py* qui contient la fonction *toutes_les_phrases(phrase)* d'une ligne et qui fait appel à votre fonction *diviser_phrase(string,n,resultat)*.

Voici quelques exemples d'exécutions:

```
Phrase: "magrandmerepossedeuncoffrefortdanssabassecours"
Dictionnaire de mots : {"grand", "mere", "grandmere", "possede", "un",
                        "coffre", "fort", "coffrefort", "dans", "sa",
                        "bassecours", "basse", "cours","ma"}

Resultat:
ma grand mere possede un coffre fort dans sa basse cours
ma grand mere possede un coffre fort dans sa bassecours
ma grand mere possede un coffrefort dans sa basse cours
ma grand mere possede un coffrefort dans sa bassecours
ma grandmere possede un coffre fort dans sa basse cours
ma grandmere possede un coffre fort dans sa bassecours
ma grandmere possede un coffrefort dans sa basse cours
ma grandmere possede un coffrefort dans sa bassecours

Phrase: "pineapplepenapple",
Dictionnaire de mots : {"apple","pen","applepen","pine","pineapple"}

Resultat:
pine apple pen apple
pineapple pen apple
pine applepen apple


Phrase: "unbonhommeestforcementungentilhomme",
Dictionnaire de mots : {"Un","bonhomme","force","ment","forcement",
                        "est","homme","bon","gentil","un","homme","gentilhomme"}

Resultat:
un bon homme est force ment un gentil homme
un bon homme est force ment un gentilhomme
un bon homme est forcement un gentil homme
un bon homme est forcement un gentilhomme
un bonhomme est force ment un gentil homme
un bonhomme est force ment un gentilhomme
un bonhomme est forcement un gentil homme
un bonhomme est forcement un gentilhomme
```

Trouvez les tous : PRO

Le jeu **Trouvez les tous** se joue par le biais d'une grille carrée $N \times N$ contenant un ensemble de lettres comme le montre la **Figure 1a**, et d'une liste de mots connue au préalable comme illustré dans **Figure 1b**



(a) Grille (4x4) du jeu "Trouvez les tous"

(b) liste contenant une partie des mots possible de former à partir de la grille

Figure 1: éléments du jeu "Trouvez les tous"

Le but du jeu est de retrouver tous les mots de la liste donnée à partir d'un enchaînement des lettres présentes dans la grille, où un mot peut être formé en utilisant des lettres adjacentes (haut, bas, gauche, droite et diagonale). Aucune lettre ne peut être utilisée plus d'une fois.

Il faut donc tracer un chemin à travers les lettres adjacentes. Deux lettres sont considérées comme adjacentes si elles se trouvent l'une à côté de l'autre horizontalement, verticalement ou en diagonale. Chaque lettre de la grille a donc au maximum huit lettres adjacentes, comme le montre la **Figure 2**.

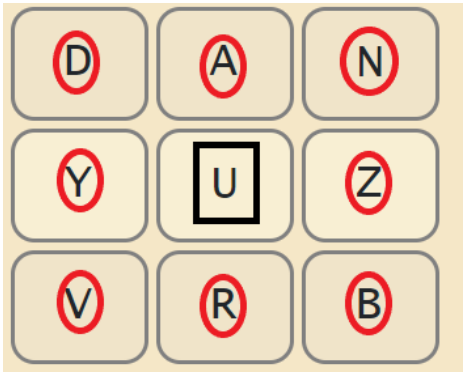


Figure 2: Lettres adjacentes à la lettre U

Vous pouvez donc commencer par n'importe quelle lettre, et toutes les lettres qui l'entourent sont acceptées. Une fois que vous passez à la lettre suivante, toutes les lettres qui l'entourent sont acceptées, à l'exception des lettres précédemment utilisées, et ainsi de suite.

A titre d'exemple, dans la grille de la **Figure 3**, en suivant les étapes illustrées, on réussit à trouver le mot *Algorithme*.

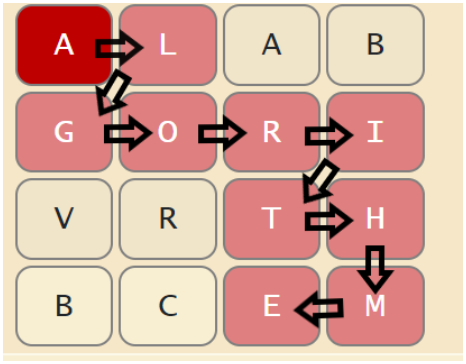


Figure 3: Exemple

Votre mission est d'écrire un programme qui va simuler ce petit jeu. Votre programme devra d'abord construire et stocker la grille du jeu, il devra ensuite parcourir cette grille afin de trouver tous les mots présents dans la liste de mots valides correspondants à cette grille.

Ce travail vous donnera un peu plus de pratique en programmation mais l'accent est mis sur la conception et l'implémentation d'algorithmes récursifs et d'une notion vue en cours, le backtracking.

Implémentation

Dans le fichier *trouvez_les_tous_pro.py* Nous vous demandons d'implémenter une classe TLT qui contiendra la grille du jeu qui sera sous la forme d'une matrice $N \times N$. La classe contiendra aussi la liste des mots possibles de former à partir de la grille. Pour cette classe, il est demandé d'implémenter les méthodes suivantes:

- Deux méthodes $recuperer\_grille(nom\_du\_fichier)$
  et $recuperer\_mots\_possibles(nom\_du\_fichier)$ qui vont lire dans un fichier *txt* les informations nécessaires à la construction d'une grille de jeu, ainsi que la liste des mots qu'il est possible de former à partir de cette grille. Cette liste vous permettra de vérifier que chaque mot trouvé dans la grille figure bien parmi les solutions possibles. S'il n'y figure pas, alors ce mot ne doit pas être pris en compte. Afin de tester votre programme, plusieurs fichiers *txt* seront fournis avec l'énoncé. Le fichier sera structuré comme suit:

```
3

C,A,T
R,R,E
T,O,N

CARTON
CARNET
NOTRE
RATER
CARTE
CARRE
TRONE
ARRET
ENORA
CARON
TAROT
TERRA
TONER
TORT
CAEN
REAC
RARE
RATE
TARE
```

Figure 4: Exemple du fichier text

- ⋆ La première ligne du fichier contiendra la taille (N) de la grille.
- ⋆ N lignes représentant la grille du jeu, chaque caractère étant séparé par une virgule.
- ⋆ Un nombre M de mots représentant la liste de mots valides que votre programme devra trouver.

Les méthodes $recuperer\_grille(nom\_du\_fichier)$
et $recuperer\_mots\_possibles(nom\_du\_fichier)$ prennent en paramètre un nom de fichier (par exemple :"$grille\_1.txt$"), placez le fichier correspondant dans le même répertoire que votre fichier *trouvez_les_tous_pro.py*.

- une méthode $prefixes(liste\_de\_mots)$ qui prend en paramètre la liste de mots valides (présents dans le fichier *txt* et qui retourne tous les préfixes. Pour des raisons de simplification, les préfixes d'un mot dans notre cas représente l'ensemble des sous-chaînes de caractères dont la taille est égale à deux minimum et à $latailledumot - 1$ au maximum. Par exemple, les préfixes de *Python* sont $[Py, Pyt, Pyth, Pytho]$. Cette méthode vous permettra de faire une optimisation qui vous évitera de faire de longues pauses café pendant que l'ordinateur cherchera futilement des mots comme zxgub,

zxaep, etc... alors que dans la langue française, il n'y a pas de mots commençant par ZX.

- une méthode *trouver_mots_rec* qui permet de construire tous les mots qu'il est possible de créer en partant d'un caractère de la grille. Vous utiliserez le concept de backtracking : dès que vous réalisez que vous ne pouvez pas former le mot commençant à une position, vous passez à la position suivante. Veillez à utiliser la méthode $prefixes$ pour abandonner les recherches qui n'ont pas d'issues.

- une méthode *trouver_tous_les_mots* qui utilise *trouver_mots_rec* et qui va parcourir toute la grille afin d'y trouver tous les mots qui peuvent être trouvés sur le plateau.

Il est évidemment fortement encouragé d'implémenter d'autres fonctions/méthodes afin de vous simplifier la vie.

Test de Trouvez les tous PRO

Pour savoir si votre programme fonctionne correctement il faut que:

- Le nombre de mots que votre programme retrouve à partir de la grille soit égal au nombre de mots de la liste donnée. Nous vous encourageons très fortement à implémenter une méthode permettant de faire cette vérification.

- le temps d'exécution soit raisonnable.

Consignes Générales

Il vous est demandé d'implémenter les solutions des deux versions du jeu **Trouvez les tous**,

Vous devez donc remettre deux fichiers : *trouvez_les_tous_lite.py* et *trouvez_les_tous_pro.py*. Veuillez respecter le nom, la casse (pas de majuscule), le format des fonctions et le noms des fichiers.

Seront évalués la qualité de votre code, la mise en pratique de la matière vue en cours et les optimisations mises en place. Veillez à ce que votre code soit commenté de manière concise pour mettre en valeur ses fonctionnalités et les optimisations appliquées. Dans le cas où l'exécution de votre code en ligne de commande produit une erreur, une note nulle sera reportée pour la partie exécution, veillez donc à tester votre code à l'aide des fichiers de test fourni avant la remise. Pour remettre votre projet sur l'UV, nous vous demandons de:

- Créer localement sur votre machine un répertoire de la forme NOM_Prenom (exemple :DUPONT_Jean) dans lequel vous mettez les fichiers *trouvez_les_tous_lite.py* et *trouvez_les_tous_pro.py* à soumettre (sans y inclure de fichier de données).

- Compresser ce répertoire via un utilitaire d'archivage produisant un *.zip* (aucun autre format de compression n'est accepté).
- Soumettre le fichier archive *.zip*, et uniquement ce fichier, sur l'UV.

Le projet est à remettre pour le 13 mars 2022 à 23h59 sur l'UV. Tout manquement aux consignes ou retard sera sanctionné directement d'un **0/10**.