DM 2

Informatique de tronc commun, classe de PC

Julien Reichert

L'esprit de ce devoir est d'utiliser des dictionnaires en Python, sauf éventuellement dans les cas où une liste / liste de listes fonctionne. Inutile de matérialiser la structure de table de hachage sous-jacente.

L'utilisation de la programmation dynamique sera pertinente pour certains exercices.

Exercice 1 : Soient deux listes d'objets quelconques (mais hachables) dont on admet qu'elles sont de même taille (inutile de le vérifier). Écrire une fonction d'évaluation similaire à celle du Mastermind et qui retourne, avec ces deux listes en argument, le nombre d'éléments placés au même endroit dans les deux listes et le nombre d'éléments de la deuxième liste placés à un endroit différent dans la première, en tant que couple.

Attention, il ne faut pas compter en trop d'occurrences les éléments placés à un endroit différent, par exemple pour les listes `[1, 4, 2, 1, 3, 3, 3]` et `[3, 4, 4, 4, 1, 1, 1]`, on ne compte que trois éléments placés à un endroit différent (en plus du `4` bien placé) : deux `1` et un `3`.

Exercice 2 : Soient deux chaînes de caractères dont on admet qu'elles sont de même taille et qu'elles ne contiennent que des lettres en majuscule (inutile de vérifier ces deux propriétés). Écrire une fonction d'évaluation similaire à celle du Wordle (ou Motus) et qui retourne, avec ces deux chaînes en argument, une liste de la même taille dont l'élément d'indice `i` vaut `2` si la lettre d'indice `i` est la même dans les deux chaînes, `1` si la lettre d'indice `i` de la deuxième chaîne apparaît dans la première à un indice différent « non encore considéré » et `0` sinon, et ce pour tout `i`.

Dans le même esprit que pour l'exercice précédent, si on considère les chaînes `"ELLES"` et `"CREEE"`, la réponse sera `[0, 0, 1, 2, 0]` car on commence par mettre le `2` à l'indice `3` pour les `'E'` alignés, puis le `1` à l'indice `2` pour l'apparition d'un `'E'` dans le premier mot, mais pas à l'indice `4` car il n'y a plus de `'E'` ailleurs dans le premier mot.

Exercice 3 : Soit une liste de listes dont les éléments peuvent être quelconques (mais hachables), on considère que la distance entre deux éléments est l'écart absolu des lignes plus l'écart absolu des colonnes. Écrire une fonction qui prend en argument une telle liste de listes et qui retourne la distance minimale et la distance maximale entre deux éléments identiques à des positions différentes, en tant que couple. Si les éléments sont tous distincts deux à deux, on retournera le couple (-1, -1).

Exercice 4 : Même exercice mais cette fois-ci on suppose qu'il y a au total `n` éléments et qu'il s'agit de tous les entiers entre `0` et `n-1`. Cette fois-ci on recherche la distance entre deux entiers consécutifs, de nouveau minimale et maximale.

Exercice 5 : Soit une matrice rectangulaire d'entiers, donnée en tant que liste de listes. Un chemin dans cette matrice est une suite de positions dans cette matrice telle que chaque position s'obtienne en augmentant soit la ligne soit la colonne d'un (« on fait un pas à droite ou en bas »). Le poids du chemin est la somme des entiers rencontrés. Écrire une fonction qui prend une telle matrice en argument et qui détermine le poids minimal d'un chemin. Calculer aussi la complexité.

On peut aussi se poser la question de la manière d'obtenir un chemin minimisant ainsi le poids. En programmation dynamique, un supplément de mémoire est utile dans cette optique, et cette pratique s'est sans doute retrouvée en première année lorsque l'algorithme de Dijkstra a été abordé.