NSI Terminale - Structure de données

Les graphes - TD. Étude

Université de Lille

2020/04/30

Deux problèmes

Problème 1 *Mener l'ours en cage*

On dispose d'une variable `DICO` qui est une liste de mots d'un même nombre de lettres.

Voici exemple avec des mots de quatre lettres (cf dico.py) :

```
DICO = ["aime", "auge", "baie", "brie", "bris", "bure", "cage", "cale", "came", "cape",
        "cime", "cire", "cris", "cure", "dame", "dime", "dire", "ducs", "dues", "duos",
        "dure", "durs", "fart", "fors", "gage", "gaie", "gais", "gale", "gare", "gars",
        "gris", "haie", "hale", "hors", "hure", "iris", "juge", "jure", "kart", "laie",
        "lame", "lime", "lire", "loge", "luge", "mage", "maie", "male", "mare", "mari",
        "mars", "mere", "mers", "mime", "mire", "mors", "muet", "mure", "murs", "nage",
        "orge", "ours", "page", "paie", "pale", "pame", "pane", "pape", "pare", "pari",
        "part", "paru", "pere", "pers", "pipe", "pire", "pore", "prie", "pris", "pues",
        "purs", "rage", "raie", "rale", "rame", "rape", "rare", "rime", "rire", "sage",
        "saie", "sale", "sape", "sari", "scie", "sure", "taie", "tale", "tape", "tare",
        "tari", "tige", "toge", "tore", "tors", "tort", "trie", "tris", "troc", "truc"]
```

Le problème que l'on se pose est le suivant : on se donne deux mots *m1* et *m2* de `DICO` et on cherche à trouver, si elle existe, une suite de mots de `DICO` telle que :

- la suite commence par *m1* et se termine par *m2* ;
- deux mots consécutifs de la suite ne diffèrent que d'une lettre, sans tenir compte de l'ordre des lettres dans chacun des mots (on dira qu'ils sont *voisins*).

Il s'agit donc de trouver une méthode de résolution qui permette de trouver une telle suite des mots permettant d'aller d'un mot de `DICO` à un autre.

En Python, cela reviendrait à écrire une fonction `solve`, paramétrée par deux chaînes de caractères, dont le résultat est la liste de mots de la suite solution quand elle existe, et `None` dans le cas contraire.

```
>>> solve('ours', 'cage')
['ours', 'duos', 'ducs', 'dues', 'dure', 'bure', 'brie', 'baie', 'aime', 'came', 'cage']
>>> solve('ours', 'orme')
None
```

Problème 2 *le taquin*

Le jeu du taquin Le jeu du taquin est un jeu dans lequel 15 petits carreaux, numérotés de 1 à 15, glissent dans une grille de 16 emplacements répartis sur une grille de quatre lignes et quatre colonnes.

Lorsque le taquin contient les carreaux dans l'ordre avec la case en bas à droite, on dit qu'il est résolu (voir Figure).

Étant donné un taquin dans un état mélangé, le but est d'arriver au taquin résolu en faisant glisser les carreaux en Haut, en Bas, à Gauche ou à Droite :

Figure 1: exmple de taquin

```
+--+--+--+--+     +--+--+--+--+    +--+--+--+--+    +--+--+--+--+
| 1| 2| 3| 4|     | 1| 2| 3| 4|    | 1| 2| 3| 4|    | 1| 2| 3| 4|
+--+--+--+--+     +--+--+--+--+    +--+--+--+--+    +--+--+--+--+
| 5| 6| 8|11|     | 5| 6| 8|  |    | 5| 6|  | 8|    | 5| 6| 7| 8|
+--+--+--+--+ ->  +--+--+--+--+ -> +--+--+--+--+ -> +--+--+--+--+
| 9|10| 7|  |     | 9|10| 7|11|    | 9|10| 7|11|    | 9|10|  |11|
+--+--+--+--+     +--+--+--+--+    +--+--+--+--+    +--+--+--+--+
|13|14|15|12|     |13|14|15|12|    |13|14|15|12|    |13|14|15|12|
+--+--+--+--+     +--+--+--+--+    +--+--+--+--+    +--+--+--+--+

+--+--+--+--+     +--+--+--+--+
| 1| 2| 3| 4|     | 1| 2| 3| 4|
+--+--+--+--+     +--+--+--+--+
| 5| 6| 7| 8|     | 5| 6| 7| 8|
+--+--+--+--+ ->  +--+--+--+--+
| 9|10|11|  |     | 9|10|11|12|
+--+--+--+--+     +--+--+--+--+
|13|14|15|12|     |13|14|15|  |
+--+--+--+--+     +--+--+--+--+
```

Si cette opération est possible, alors on dit que le taquin est résoluble. Un taquin n'est pas toujours soluble.

Le problème est le suivant : étant donné un taquin mélangé que l'on suppose résoluble, peut-on trouver une suite de mouvements pour le résoudre ?

Étude

Étudiez ces deux problèmes et proposez un algorithme pour résoudre chacun, puis répondez aux questions suivantes :

- Parvenez-vous à dégager des points communs entre vos deux algorithmes ?
- Pouvez-vous citer un (ou plusieurs) autre exemple de problème qui pourrait se résoudre en utilisant un algorithme similaire ?
- Votre algorithme calcule-t-il la solution la plus courte (en nombre de pas de résolution) ? Sinon, est-il possible de le modifier pour que cela soit le cas ?