[Boggle] Algorithme de recherche d'un mot sur une grille

Les algorithmes qui suivent font l'hypothèse que la grille et les lettres qui la composent sont accessibles. De plus, les algorithmes nécessitent de pouvoir marquer les lettres de la grille comme étant visitées (ou non visitées). Cela revient à stocker un booléen par position dans la grille.

Algorithme 1 : Recherche principale

```
1 Fonction Recherche(mot : String)
2     marquer toutes les lettres de la grille comme étant non visitées
3     pour toute coordonnée coord de la grille faire
4         si SousRecherche(mot, 0, coord) alors
5             retourner vrai
6     retourner faux
```

L'algorithme principal ci-dessus cherche à placer dans la grille le mot à partir de chaque coordonnée possible. Il repose sur la fonction récursive SousRecherche. Cette fonction vise à déterminer si le mot commençant à la position *pos* peut être placé à partir de la coordonnée *coord*. Ceci n'est possible que s'il n'y a plus de lettre à placer (testé ligne 2) ou si :

1. *coord* désigne une case de la grille (testé ligne 4) et,
2. la lettre en coordonnée *coord* correspond à la lettre en position *pos* du mot (testé ligne 6) et,
3. la lettre en coordonnée *coord* n'a pas déjà été employée pour une lettre précédente du mot (testé ligne 8) et,
4. la suite du mot peut être placée à partir d'une case adjacente de la grille (testé ligne 12).

Une première version de la fonction SousRecherche est donnée ci-dessous sous forme récursive.

Algorithme 2 : Recherche récursive

```
1  Fonction SousRecherche(mot : String, pos : int, coord : Coord)
2      si pos ≥ mot.length() alors
3          retourner vrai
4      si coord est hors-limite alors
5          retourner faux
6      si la lettre de la grille en coord ne correspond pas à la lettre à la position pos de la chaine mot alors
7          retourner faux
8      si la lettre de la grille en coord est marquée comme étant visitée alors
9          retourner faux
10     marquer la lettre de la grille en coord comme étant visitée
11     pour toute coordonnée coord′ adjacente à coord faire
12         si SousRecherche(mot, pos + 1, coord′) alors
13             retourner vrai
14     marquer la lettre de la grille en coord comme étant non visitée
15     retourner faux
```

La fonction SousRecherche peut aussi être écrite sous forme itérative même si cela impose l'emploi explicite d'une pile.

L'ensemble DIRECTIONS réuni toutes les 8 directions possibles et nous faisons l'hypothèse que nous savons calculer la coordonnée atteinte lorsque nous nous déplaçons d'une case dans une direction donnée à partir d'une coordonnée donnée.

On peut facilement remarquer que le paramètre *pos* de cette version de la fonction SousRecherche pourrait être une simple variable locale initialisée à zéro. Le paramètre a été laissé uniquement pour correspondre au prototype de la fonction SousRecherche de l'algorithme récursif.

Les éléments contenus dans la pile sont des couples formés d'une coordonnée et d'un ensemble de directions. Il représente les cases en cours d'analyse et les directions qui n'ont pas été explorées.

Les deux premiers tests consistent respectivement à vérifier que le mot à placer n'est pas vide et que la première lettre de ce mot est bien présente sur la grille au coordonnée de départ. Si ces deux conditions sont vérifiées alors une recherche pour placer les lettres suivantes du mot est entamée. A cette fin, la coordonnée de départ et les

directions non encore explorées sont empilées. Comme aucune direction n'a encore été analysée, l'ensemble complet DIRECTIONS est mis sur la pile.

A chaque étape, l'algorithme essaye de trouver une nouvelle direction à suivre pour compléter son analyse. S'il n'en trouve une valide, cette direction est retirée des directions non encore explorées (et ceci est mémorisé sur la pile) et l'analyse est poursuivie dans la direction choisie. S'il n'en existe aucune, la pile est dépilée pour revenir en arrière dans le mot et continuer la recherche dans une nouvelle direction parmi les restantes.

Algorithme 3 : Recherche itérative

```
1  Fonction SousRecherche(mot : String, pos : int, coord : Coord)
2      si pos ≥ mot.length() alors
3          retourner vrai
4      si la lettre de la grille en coord ne correspond pas à la lettre à la position pos de la chaine mot alors
5          retourner faux
6      empiler le couple (coord, DIRECTIONS) dans la pile
7      marquer la lettre de la grille en coord comme étant visitée
8      pos = pos + 1
9      tant que la pile n'est pas vide faire
10         mettre le couple étant au sommet de la pile dans (coord, DIR)
11         dépiler la pile
12         si il existe une direction dir ∈ DIR vérifiant estValide(mot, pos, coord, dir) alors
13             si pos + 1 ≥ mot.length() alors
14                 retourner vrai
15             réempiler le couple (coord, DIR \ {dir}) dans la pile
16             calculer la coordonnée coord′ atteinte à partir de coord et dir
17             empiler le couple (coord′, DIRECTIONS) dans la pile
18             marquer la lettre de la grille en coord′ comme étant visitée
19             pos = pos + 1
20         sinon
21             marquer la lettre de la grille en coord comme étant non visitée
22             pos = pos − 1

23 Fonction estValide(mot : String, pos : int, coord : Coord, dir : DIRECTIONS)
24     calculer la coordonnée coord′ atteinte à partir de coord et dir
25     si coord′ est hors-limite alors
26         retourner faux
27     si la lettre de la grille en coord′ ne correspond pas à la lettre à la position pos de la chaine mot alors
28         retourner faux
29     si la lettre de la grille en coord′ est marquée comme étant visitée alors
30         retourner faux
31     retourner vrai
```