Commentaires - TP Défi - Lacs en Finlande en OCaml

Remarques générales :

- réviser régulièrement le mémo OCaml (ici : manipulation des tableaux, références),

- éviter les erreurs classiques :
    vouloir modifier une variable (-> utiliser une référence),
    vouloir renvoyer une information depuis l'intérieur d'une boucle (-> utiliser une référence)
    ne pas se tromper pour les indices valides dans un tableau OCaml

- compiler et exécuter au fur et à mesure. trop de codes fournis ne compilent pas

- planifier :
    ne pas s'embêter avec les bords (ne gérer que les cases intérieures ?)
    ne pas redéfinir des fonctions existantes (min, max)
    obtenir la carte des profondeurs est un bon objectif : cela permet de visualise l'exécution de son algorithme en affichant la carte, et d'obtenir facilement le résultat (recherche du maximum)

Algorithmes pour résoudre le problème :

- plusieurs algorithmes sont envisageables pour trouver la profondeur d'une case EAU (i0,j0) :

    - utiliser la définition : c'est le minimum des distances à toutes les cases TERRE. Il suffit de réaliser un parcours complet du tableau (-> correction version 1)

    - partir de (i0,j0) vers le haut, vers le bas, vers la gauche, vers la droite jusqu'à atteindre la TERRE, et conserver la plus courte distance ... malheureusement, cet algorithme est incorrect : voir la carte 1

    - il faudrait tenir compte de tous les chemins possibles, ce qui revient à explorer des zones de plus en plus larges autour de (i0, j0) jusqu'à atteindre la TERRE.

    - récursivement ? le profondeur est le minimum de la profondeur des 4 voisins +1 ... malheureusement, difficile de faire en sorte que les appels récursifs ne bouclent pas : le calcul de la profondeur de (i0, j0) nécessite le calcul de la profondeur de (i0+1, j0) qui lui-même nécessite la profondeur de (i0, j0) ...

    - autre approche : calculer les profondeurs par couches "successives" : on marque -1 (profondeur inconnue) toutes les cases EAU ; puis progressivement les cases avec leur profondeur : 1 (les cases EAU voisines de 0); 2 (les cases EAU voisines de 1); ... ainsi de suite (-> correction version 2). On peut optimiser en conservant la liste des cases qu'il reste à traiter.