TP9-TP10
Recherche de chemin dans un réseau routier

On s'intéresse au réseau routier apparaissant sur une photographie aérienne. Cette photographie est en noir et blanc. On supposera dans la suite que les teintes les plus foncées (noir) correspondent aux routes, et que les divers autres éléments de la photo (champs, agglomérations, forêts) sont de divers tons de gris. On veut développer un programme C permettant de rechercher s'il existe un chemin entre deux points apparaissant sur la photo, et de trouver le chemin le plus court. Pour cela, on propose de représenter la photo en C par un tableau $N \times N$ que l'on appellera "carte". On quadrille la photo avec des carrés de petite dimension et chaque case du tableau est mise en correspondance avec l'un de ces carrés. La case du tableau reçoit alors un nombre représentant la densité de la couleur du carré. Plus cette couleur est foncée, plus le nombre est grand. La suite du traitement se fait comme suit :

- On binarise la carte pour extraire le réseau routier du "fond", c'est à dire qu'on remplace les nombres situés dans chaque case du tableau par un 0 ou un 1. La binarisation se fait en calculant la moyenne des éléments du tableau et en remplaçant les éléments supérieurs ou égaux à cette moyenne par des 1 et ceux qui sont inférieurs par des 0.
- On cherche s'il existe un chemin entre deux points (voir définitions ci-dessous).
- On détermine le chemin le plus court (voir définition ci-dessous).

On utilisera les définitions suivantes :

Définition 1 : Un élément du tableau identifié par ses numéros de ligne et de colonne est un *point* s'il est égal à la valeur 1.

Définition 2 : Les *points voisins* d'un point donné sont les points du tableau se trouvant à droite, à gauche, au dessus et au dessous de ce point (les points situés sur les diagonales par rapport au point donné ne sont pas considérés comme des voisins). Ainsi, les points au bord de la carte ou dans les angles ont au maximum deux ou trois voisins.

Définition 3 : Un *chemin* entre deux points D (départ) et A (arrivée) est une suite non vide de points p_1, $p_2 \ldots p_n$, telle que $\forall i$ appartenant à $[1...n-1]$, p_i est voisin de p_{i+1}, avec $D = p_1$ et $A = p_n$.

Définition 4 : La *longueur d'un chemin* est égale au nombre de points qui le composent moins un.

Questions :

(pensez à faire des tests les plus exhaustifs possible à chaque étape !)

1. Donner en C la spécification de la carte, son type, et les procédures de manipulation de cette structure de données.
2. Écrire en C la fonction de binarisation.
3. Définir en C une fonction récursive qui recherche s'il existe un chemin entre deux points de la carte.
4. Transformer la fonction récursive précédente en une fonction récursive qui donne tous les chemins existant entre deux points de la carte.

5. Soit l'algorithme itératif suivant donnant la longueur du plus court chemin entre deux points :

Algorithme 1 : fonction LONGUEUR (maCarte : carte ; départ,arrivée : point)

```
% Variables locales :
  % Àvoir : ensemble
  % Vu : ensemble
  % distance : tableau d'entiers de dimension nbPoints
  % S, V : point
début
    Àvoir = {départ}
    Vu = {}
    distance[départ] = 0
    tant que nonVide(Àvoir) et arrivée n'appartient pas à Vu faire
        CHOISIR S dans Àvoir
        ajouter S à Vu
        pour tout V voisin de S dans maCarte faire
            si V n'appartient pas à Vu alors
                ajouter V à Àvoir
                distance[V] = distance[S] + 1
            sinon
                si distance[V] > distance[S] + 1 alors
                    ajouter V à Àvoir
                    retirer V de Vu
                    distance[V] = distance[S] + 1
    si arrivée appartient à Vu alors
        retourner distance[arrivée]
    sinon
        retourner pasDeChemin
fin
```

(a) Définir en C le type ensemble (on peut s'appuyer sur des types déjà connus).

(b) Écrire en C les fonctions : INIT_ENS, ENS_EST_VIDE, APPARTIENT_ENS, RETIRER_ENS et AJOUTER_ENS agissant sur des objets de type ensemble. Puis, écrire les fonctions CHOISIR et LONGUEUR.