# TP 3 : Bord et composantes connexes

## Exercice 1

On considère le format de maillage introduit au TP précédent, et on suppose donné un nuage de points consistant en un tableau `vtx` de paires de `float`. Selon ce format, dans un maillage triangulaire, chaque triangle est représenté par un triplet de `int`. De la même manière, on peut modéliser le maillage d'un contour 1D comme collection d'arêtes qui, en `Python`, sera représentée par un tableau de paires de `int`.

Si l'on reprend l'exemple du `maillage1.msh` en annexe de la feuille de TP2, le bord du maillage triangulaire est alors un contour représenté par le tableau suivant.

```
eltb =
[[0 ,  1  ]
 [1 ,  2  ]
 [2 ,  3  ]
 [3 ,  7  ]
 [7 ,  11 ]
 [10 , 11 ]
 [9 ,  10 ]
 [8 ,  9  ]
 [4 ,  8  ]
 [0 ,  4  ]]
```

**Question 1.1** Écrire une fonction `Boundary` prenant en argument un tableau `elt` de `int` de taille `nbr_elt`$\times 3$ représentant un maillage triangulaire, et qui renvoie en sortie un tableau `eltb` de `int` de taille `nbr_eltb`$\times 2$ représentant le bord du maillage. On pourra s'aider des types `set` et `dict` de la bibliothèque standard de `Python`.

**Question 1.2** Modifiez la fonction `PlotMesh` mise au point dans le TP2 afin que l'on puisse également lui passer un tableau de taille `N`$\times 2$ en entrée, auquel cas `PlotMesh` réalisera l'affichage graphique du contour. Vous testerez votre travail en réalisant l'affichage du bord pour le domaine du fichier `maillage3.msh`.

**Question 1.3** Modifiez la fonction `Boundary` afin qu'elle renvoie un tuple `(eltb,be2e)` dans lequel `eltb` désigne le même tableau qu'à la question 1.1, et `be2e` est tableau de `int` de taille `nbr_eltb` repésentant une table de correspondance entre les triangles de `elt` et

les arêtes de `eltb`. Cette table de correspondance adoptera le format suivant :

```
be2e[j] = 3*p+k
pour  j = 0,..., nbr_eltb-1
      p = 0,..., nbr_elt-1
      k = 0,1,2
```

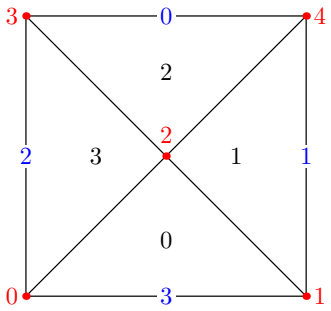
signifie que la `j`-eme arête du bord est la `k`-eme arête du `p`-eme triangle de `elt`. On prendra comme convention que, dans un triangle, la `k`-eme arête est celle qui est opposée au `k`-eme sommet. Ci-dessous on donne un exemple d'une telle table de correspondance pour un maillage très simple à 4 triangles.

```
vtx =
[[0.,  0. ]
 [1.,  0. ]
 [0.5, 0.5]
 [0.,  1. ]
 [1.,  1. ]]
```

```
elt =
[[0,  1,  2 ]
 [1,  2,  4 ]
 [2,  3,  4 ]
 [0,  2,  3 ]]
```

```
eltb =
[[3,  4 ]
 [1,  4 ]
 [0,  3 ]
 [0,  1 ]]

be2e =
[6,  4,  10,  2]
```




# Exercice 2

**Question 2.1** Écrire une fonction `CCmpt` prenant en argument d'entrée un tableau `elt` représentant un maillage triangulaire de taille `nbr_elt`×3, et renvoyant en sortie un tableau `cc` de `int` de taille `nbr_elt` tel que `cc[j] = k` si le triangle no.`j` du maillage appartient à la composante connexe no. `k` du maillage. Vous adopterez la numérotation des composantes connexes que vous souhaitez. Vous pourrez vous aider de la fonction `scipy.sparse.csgraph.connected_components`.

**Question 2.2** En utilisant la fonction de la question précédente, réalisez l'affichage graphique du domaine de calcul associé au fichier `maillage4.msh` dans lequel vous ferez apparaitre chacune des composantes connexes du domaine dans une couleur différente.

**Question 2.3** Adaptez `CCmpt` pour qu'elle puisse également prendre en argument d'entrée un tableau représentant le maillage d'un contour (l'argument d'entrée `elt` sera alors de taille `nbr_elt`×2) et qui renverra en sortie un tableau `cc` indiquant sa décomposition en composantes connexes.

**Question 2.4** En utilisant la fonction de la question précédente, réalisez l'affichage graphique du bord du domaine de calcul associé au fichier `maillage5.msh` dans lequel vous ferez apparaitre chacune des composantes connexes de ce contour dans une couleur différente.

# Exercice 3

Écrire une fonction `Refine` prenant en argument d'entrée des tableaux `vtx` et `elt` représentant un maillage, et qui renvoie en sortie deux autres tableaux `refined_vtx` et `refined_elt` qui correspondent à ce même maillage soumis à un raffinement barycentrique. Un raffinement barycentrique consiste à sub-diviser chaque triangle en 4 sous-triangles selon le schéma ci-dessous dans lequel on a introduit les milieux de chaque arête. Vous vérifirez bien entendu le résultat de votre travail au moyen d'un affichage graphique.