# Algorithmique et Structures de données

L2 2021-2022

## Travaux Pratiques 10

---

`gcc mon_fichier.c -std=c11 -Wall -Wextra -o mon_programme`

Les exercices marqués d'un @ sont optionnels et à faire dans un second temps.

---

Bien qu'elle soit très gourmande en espace, on utilisera la définition de type suivante pour représenter des graphes. Nos graphes sont étiquetés par des nombres entiers positifs ou nuls.

Si `i` et `j` sont deux sommets du graphe la case `graphe[i][j]` doit contenir ...

— ...$-1$ si il n'y a pas d'arc allant du sommet `i` vers le sommet `j`.

— ...$a$ avec $a$ positif si il y a un arc allant du sommet `i` vers le sommet `j` étiqueté par $a$.

```
# define MAX_NOEUD 40  // Le nombre maximum de sommets d'un graphe
typedef struct gra
{
   int graphe[MAX_NOEUD][MAX_NOEUD];
   int nbNoeud; // le nombre de sommet du graphe
} Graphe;
```

**Exercice 1.** *Création de graphe*

Créer la fonction `void creerGrapheE(Graphe* g,FILE* grDesc)` qui construit un graphe à partir du fichier `grDesc`.

Exemple de contenu du fichier `grDesc`.

```
7
1->7:3
4->4:2
7->3:1
2->1:3
3->1:64
```

le graphe a 7 sommets 1 2 3 4 5 6 7 (il n'y a pas de sommet 0)

Pour la lecture on peut utiliser `fscanf(grDesc,"%d->%d:%d",&d,&a,&e)`.

**Exercice 2.** *Plus court chemin*

Créer la fonction `void cheminOpti(Graphe* g,int s,int a)` qui affiche le plus court chemin pour aller de `s` à `a` s'il existe.

**Exercice 3.** *Tri topologique*

Créer la fonction `void triTopo(Graphe* g,int* tab)` qui remplit `tab` par un tri topologique du graphe `g` supposé sans circuit.

**Exercice 4.** *@Composante connexe*

Créer la fonction `void memeCompos(Graphe* g,int s,int *tabl)` remplit le tableau `*tabl` de la manière suivante. La case `i` doit contenir 1 si et seulement si il existe un chemin de `i` à `s` et un chemin de `s` à `i`.

**Exercice 5.** *@Affiche circuit*

Créer la fonction `void circuit(Graphe* g)` qui affiche tous les circuits élémentaires du graphe `g`.