

TÉLÉCOM PARIS X

TÉLÉCODE

SÉANCE 7 | TÉLÉCODE COMPOSANTES FORTEMENT CONNEXES

Présenté et rédigé par Léopold Bernard

PLAN

01

COMPOSANTES CONNEXES

02

COMPOSANTES FORTEMENT CONNEXES

03

ALGORITHME

04

IMPLÉMENTATION

05

PROBLÈMES

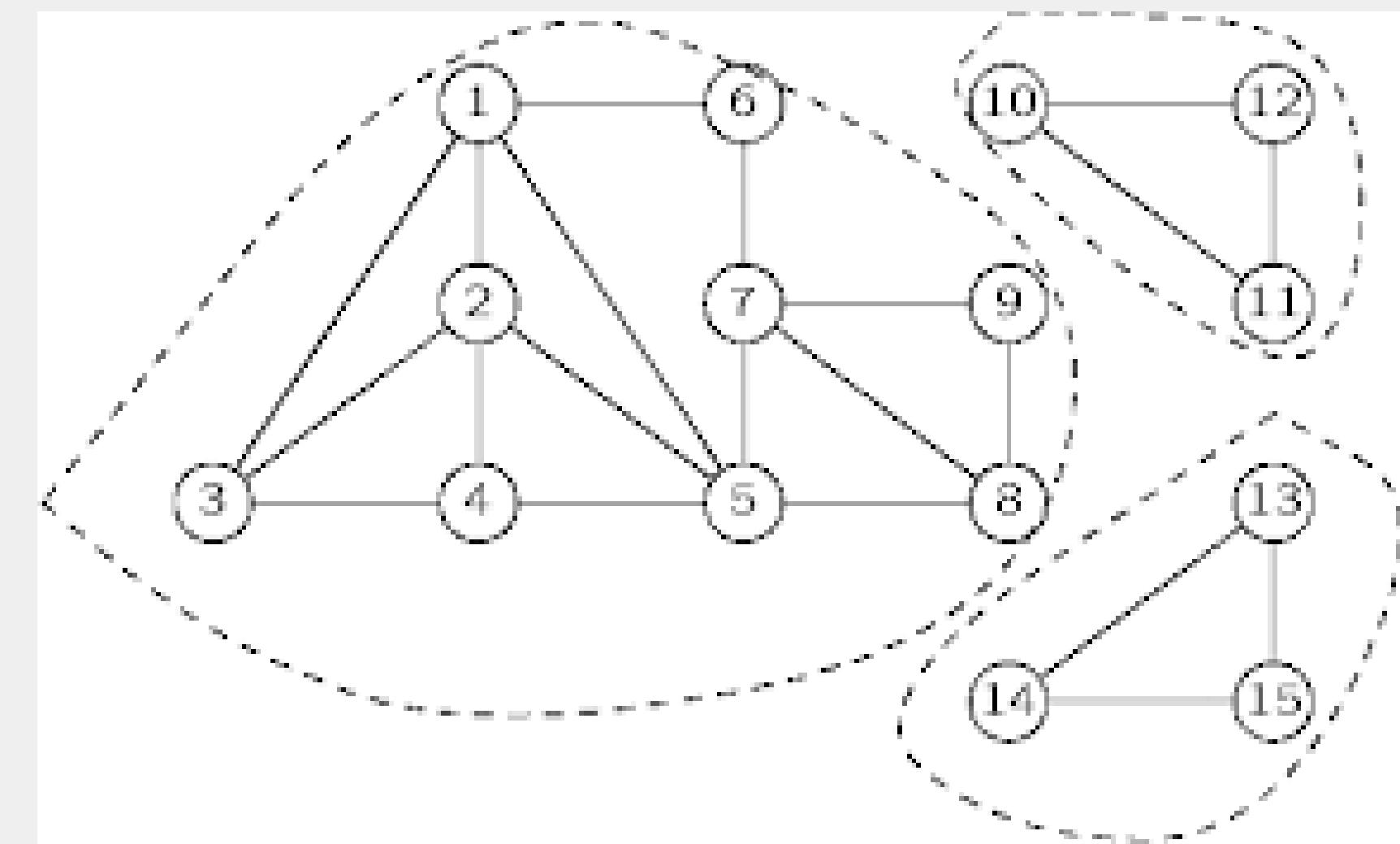
1. COMPOSANTES CONNEXES

- $G = (V, E)$ non orienté
- Un graphe est connexe si à partir de tout sommet u on peut atteindre tout autre sommet v
- Un graphe peut être décomposé en composantes connexes
- => Algorithme : BFS ou DFS en $O(|V| + |E|)$



1. COMPOSANTES CONNEXES

Les composantes connexes du graphe sont entourées en pointillés



2. COMPOSANTES FORTEMENT CONNEXES

- $G = (V, E)$ orienté
- Comment peut-on définir une **relation d'équivalence** intéressante entre deux sommets d'un graphe orienté ?
- $u \sim v \Leftrightarrow$ Il existe un chemin de u à v et il existe un chemin de v à u
- Les composantes fortement connexes sont les **classes d'équivalences** !



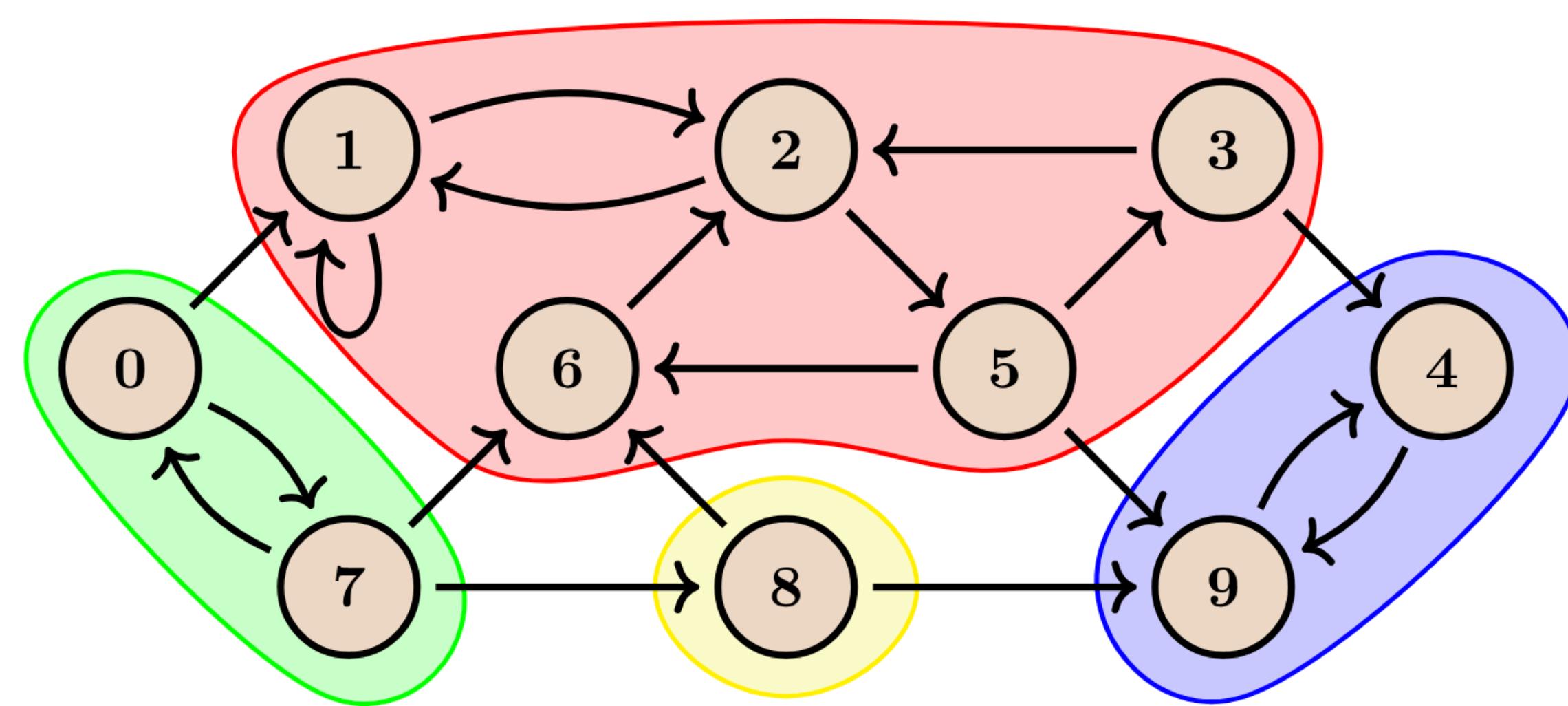
2. COMPOSANTES FORTEMENT CONNEXES

- Remarque : on peut donc réduire notre graphe en le quotientant par la relation d'équivalence
=> Graphe des composantes fortement connexes (**condensation graph**, noté GC)
- Noeuds(GC) = {SCC de G}
- Arêtes(GC) : SCC_i est reliée à SCC_j dans GC s'il existe un noeud de SCC_i relié à un noeud de SCC_j



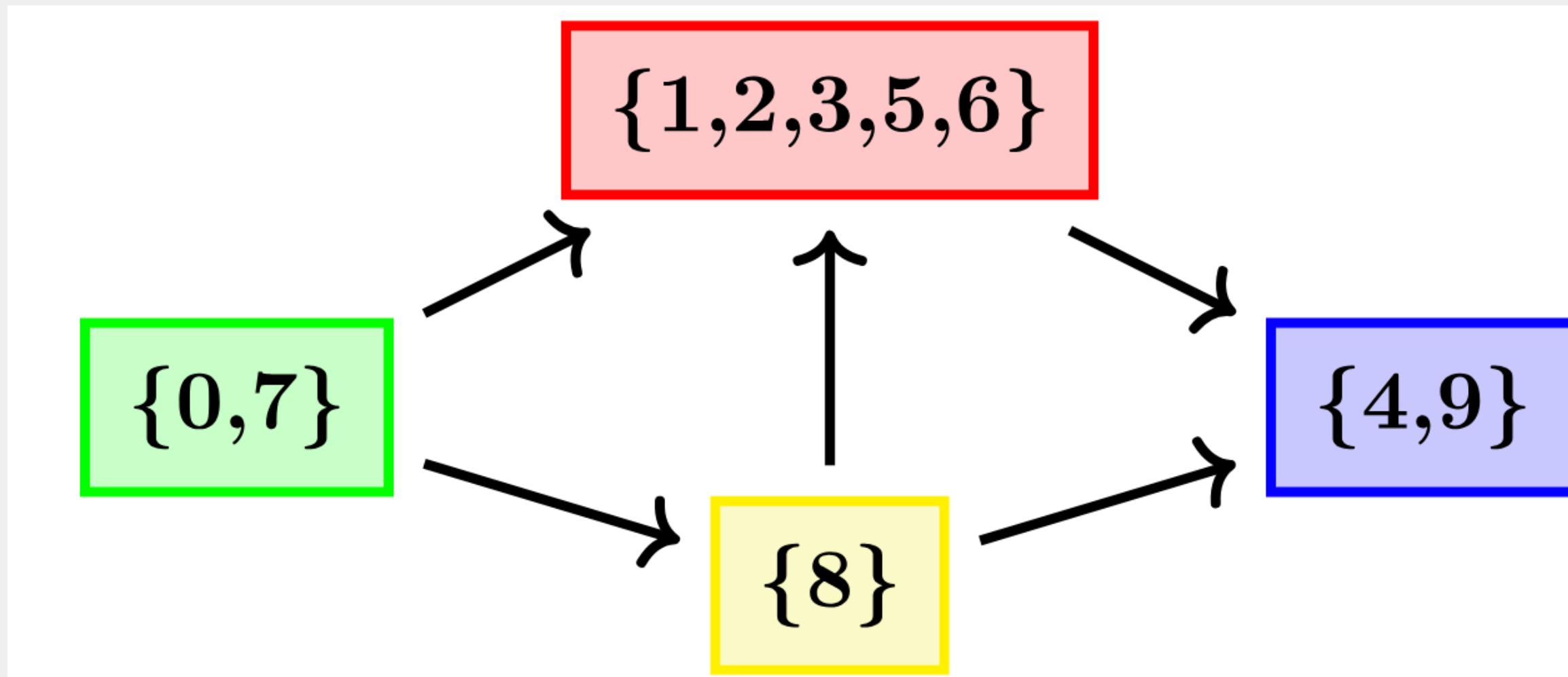
2. COMPOSANTES FORTEMENT CONNEXES

Graphe orienté G et ses composantes fortement connexes



2. COMPOSANTES FORTEMENT CONNEXES

Graphe condensé GC du graphe G précédent



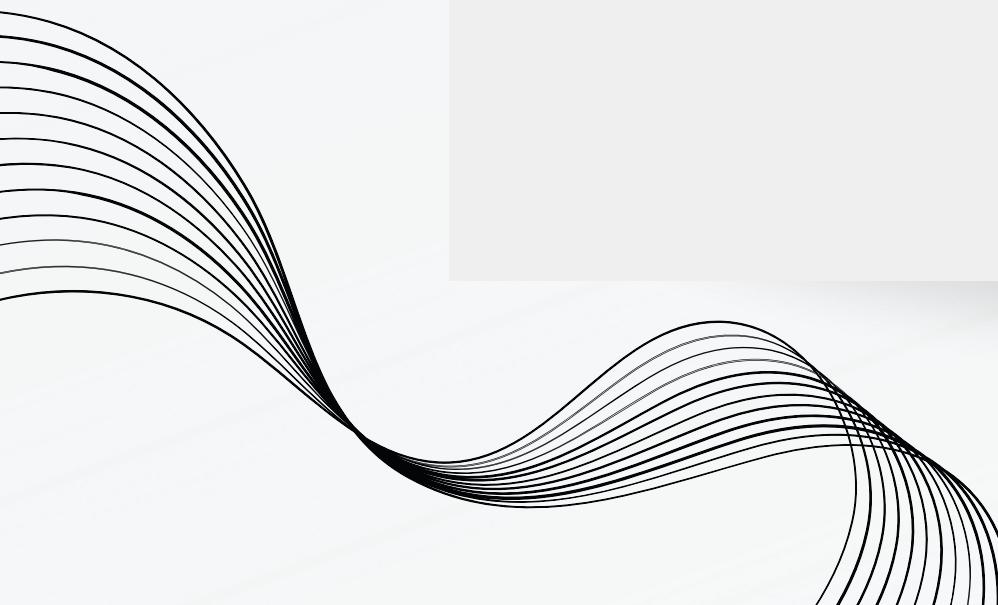
3. ALGORITHME

- Kosaraju (~1980)
- Idée principale : 2 DFS
- Complexité linéaire i.e. $O(|V| + |E|)$



3. ALGORITHME

- DFS1 :
 - on construit un classement des nœuds par ordre décroissant de temps de fin de visite (**ordre suffixe** du parcours)



3. ALGORITHME

- DFS2 :
 - on le fait sur le **graphe transposé** G^T (on inverse toutes les arêtes)
 - pour un sommet u , lorsque l'on commence DFS2 avec u on est certains que les nœuds de sa SCC sont après lui dans l'ordre donné par DFS1 et donc n'ont pas encore été visités
 - => Difficile à comprendre, mais c'est la clé



4. IMPLÉMENTATION

Fonction DFS1

```
def dfs1(self, u) :  
    self.visited[u] = True  
    for v in self.g[u] :  
        if not self.visited[v] :  
            self.dfs1(v)  
    self.post_ordre.append(u)
```

4. IMPLÉMENTATION

Fonction DFS2

```
def dfs2(self, u) :  
    self.visited[u] = True  
    self.curr_composante.append(u)  
    for v in self.gt[u] :  
        if not self.visited[v] :  
            self.dfs2(v)
```

4. IMPLÉMENTATION

Fonction **Kosaraju** pour trouver les SCC (CFC en français)

```
def kosaraju(self) :
    for i in range (self.n) :
        if not self.visited[i] :
            self.dfs1(i)
    self.reset_visited()
    assert len(self.post_ordre) == self.n, "Post-ordre n'a pas la taille convenable"
    for u in self.post_ordre[::-1] :
        if not self.visited[u] :
            self.curr_composante = []
            self.dfs2(u)
            self.cfc.append(self.curr_composante.copy())
    return self.cfc
```

4. IMPLÉMENTATION

Fonction pour construire le graphe des SCC (**graphe condensé**)

```
def grapheCFC(self) :
    cfc = self.kosaraju()
    nb_cfc = len(cfc)
    for i in range(nb_cfc) :
        for u in cfc[i] :
            self.composante[u] = i
    l = [set() for _ in range(nb_cfc)]
    for u in range(self.n) :
        for v in g[u] :
            if self.composante[u] != self.composante[v] :
                l[self.composante[u]].add(self.composante[v])
    res = [list(L) for L in l]
    return res
```

5. PROBLÈMES

France-IOI (ch. 6) :

Composantes fortement connexes

Codeforces :

427/C (about SCC)

Kattis :

[open.kattis.com/problems/johndillermand
problems/downfall](https://open.kattis.com/problems/johndillermand/problems/downfall)