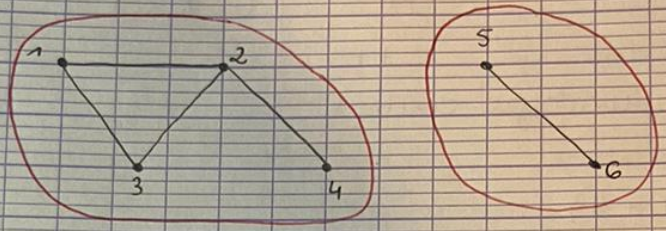
4 : 1

I Composantes connexes

→ Composante connexe dans un graphe non orienté $G=(V,E)$ sous-graphe connexe maximal de G (i.e. ce sous graphe connexe ne doit pas être inclus dans un sous-graphe connexe plus grand)



2 composantes connexes

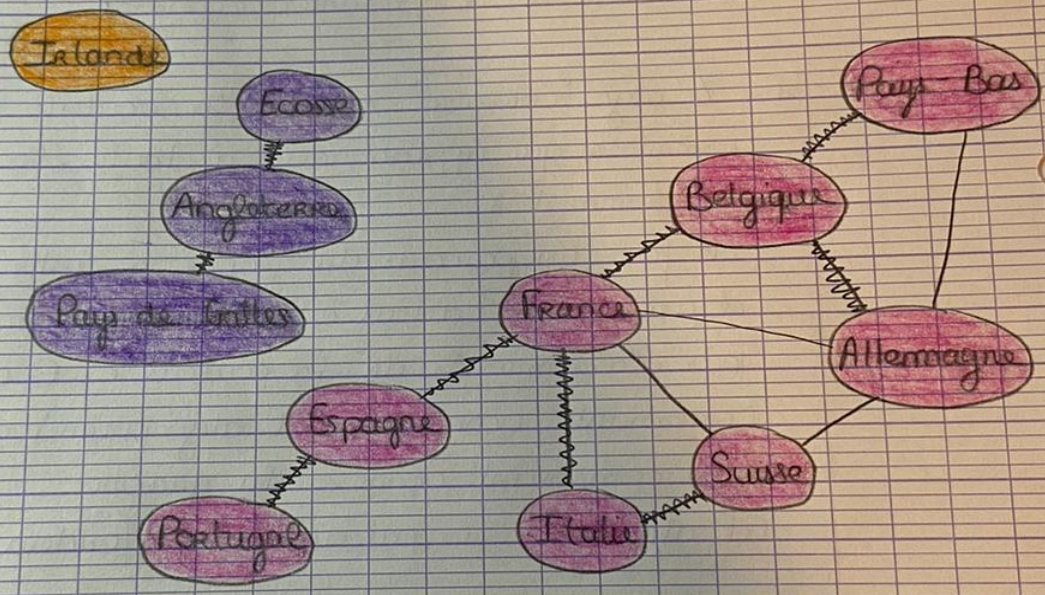
1. L'algo "colorier" associe la couleur c à tous les sommets qui appartiennent à toutes les chaines partant du sommet s.
Il fait un parcours du graphe à partir du sommet s et associe la couleur c à tous les sommets rencontrés.
Tous les sommets de la couleur c $\in$ à la même composante connexe. Cet algo permet donc de trouver une composante connexe du graphe
Cet algo fait un parcours en profondeur du graphe.

2. Comp_connexe (G : graphe non orienté) : tableau
Entrée : graphe $G=(V,E)$
Sortie : tableau couleur contenant les couleurs associées à chaque sommet du graphe
Variables locales : couleur : tableau de taille |V|,
c : entier

```
Pour chaque sommet s de V
    couleur[s] ← 0

c ← 1
Pour chaque sommet s de V
    si couleur[s] == 0
        Colorier(G, s, c)
        c++
renvoyer couleur
```

3) + 4)



c = 1 : Allemagne - Belgique - France - Espagne - Port

Pays-Bas

Italie - Suisse

c = 2 : Angleterre - Ecosse

Pays de Galles

c = 3 : Irlande

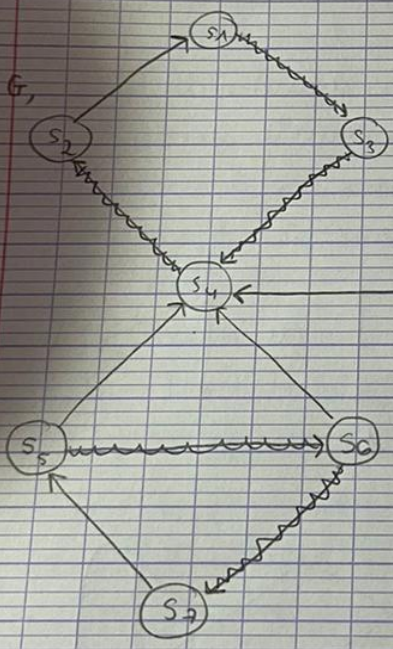
II Composantes fortement connexes

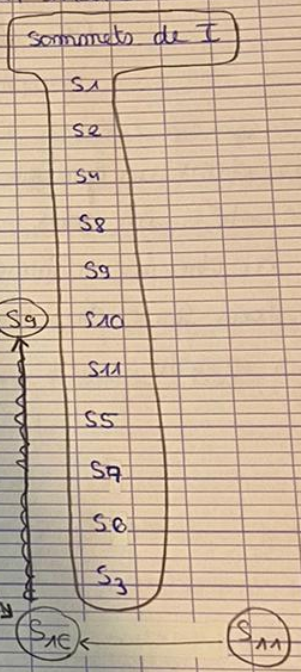
Sommet de G	début d	fin f	père
s_1	1	22 x	
s_2	2	21 x	s_1
s_3	4	5 x	s_4
s_4	3	20 x	s_2
s_5	6	11 x	s_4
s_6	8	9 x	s_7
s_7	7	10	s_5
s_8	12	19 x	s_4
s_9	13	18 x	s_8
s_{10}	14	17 x	s_9
s_{11}	15	16 x	s_{10}

d : date de début de traitement d'un sommet, première fois qu'on le rencontre

f : date de fin, dernière fois que l'on voit ce sommet, tous ses succ sont traités

Arcs inversé de G, I.





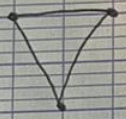
mm parcours en profondeur de T

Composantes fortement connexe : • S1 - S3 - S4 - S2
• S8 - S10 - S9
• S11
• S5 - S6 - S7

III k-connexité

Un graphe k-connexe est un graphe connexe qu'il est possible de déconnecter en supprimant k arêtes et tel que ce k soit minimal.
(k : le nombre minimal d'arêtes à supprimer dans le graphe pour le rendre non connexe)

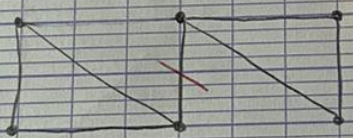
clique dans un graphe non orienté : un sous graphe où chaque sommet est relié à tous les autres sommets du sous graphe



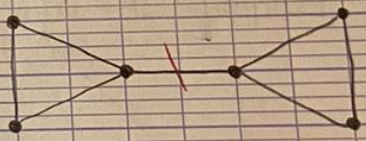
Graphe 2-connexe (si on supprime 1 arête toujours connexe)

→ donc 2 arêtes à supp pour le rendre non connexe

1.

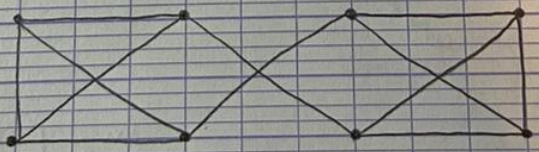


ou

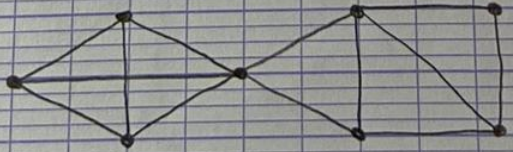


1-connexe

2.



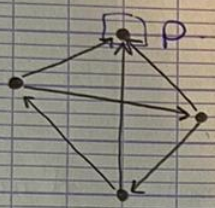
ou



2-connexe

3. Un graphe n'est pas k-connexe s'il existe au plus k+1 chaines dont les arrêtes sont toutes disjointes, entre 2 paires de sommets.
Car si on a k+1 chaines entre deux sommets, il faudra enlever k+1 arrêtes (1 de chaque chaine) pour déconnecter le graphe

IV Un exercice supplémentaire et optionnel



1. * Un graphe G contenant un puit parfait ne peut pas être fortement connexe, car il n'existe pas de chemins entre le sommet p et les autres sommets. (par def du puit parfait)
* Le Graphe G' associé sera connexe puisque pour tout couple de sommet il existe une chaine (sans prendre en compte le sens)

2. Le Graphe G ne peut pas contenir plusieurs puits parfaits. En effet, prenons un graphe G contenant 2 puits parfaits P_1 et P_2. Il doit y avoir un arc entre P_1 et P_2 (P_2 puits parfait) mais aussi de P_2 vers P_1 (P_1 puits parfait). Or ce sont deux puits parfaits donc il ne doivent pas avoir d'arcs sortants. → problème donc impossible.

3. nombre minimum d'arcs atteint s'il n'y a dans le graphe que les arcs des autres $n-1$ sommets vers le puits : $n-1$ arcs au minimum

. nombre max d'arcs atteint s'il y a un arc entre tout couple de sommets (u,v), $u \neq p$ et $v \neq p$:

$\underbrace{n-1}_{\text{nbr d'arcs des } n-1 \text{ sommets jusqu'au puit.}} + \underbrace{(n-1)(n-2)}_{\text{nbr d'arcs entre les } (n-1) \text{ sommets}} = (n-1)^2$

On recherche un sommet tel que il n'y ait que des 0 sur la ligne i et que des 1 sur la colonne J (sauf la case (i,i), pas de boucle)

4. Puits_parfait (MG : matrice d'adjacence d'un graphe orienté $G = (V, A)$) : booléen.
Sortie : vrai si G contient un puit parfait, faux sinon

Début :

```
J ← 0
Pour i de 0 à |V|-1
    Tant que J < |V|
        Si MG[i][J] = 0 et ((MG[J][i] = 1 ET
        i != J) OU (MG[J][i] = 0 ET i = J))
            J ← J+1
            Si J == |V|
                Renvoyer vrai
            Sinon
                J ← |V| ← ce sommet d'indice i n'est
                          pas un puit, on peut passer
                          au suivant
Renvoyer faux
```

Complexité : $\Theta(n^2)$ (au pire des cas) — n : nbr de sommet. 4 parcours de la matrice.