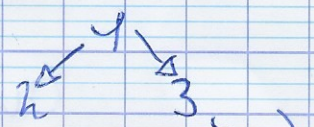
TD02 IN403

1) a) Le 1er sommet vu dans la boucle pour est le sommet 1

A traiter	1			...
	↑ i	↑ j		

i < j j = 1 i = 2
1 2 On ajoute les 2 successeurs de 1 (2 et 3) qui sont "non vu"



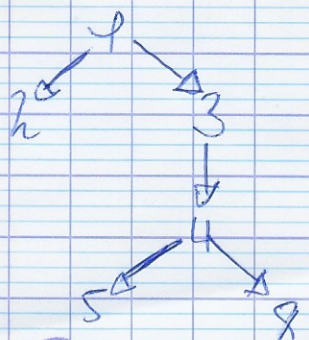
(à travers Pile)

A traiter	1	2	3	...
		↑ i		↑ j

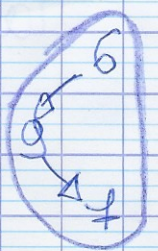
i < j j = 2 i = 3
2 4 Le seul successeur de 2 est déjà "vu" je ne le passe rien
i < j j = 3 i = 4
3 4

A traiter	1	2	3	4	...
				↑ i	↑ j

i < j j = 4 i = 5
4 5



A traiter	1	2	3	4	5	6	7	8	
					↑ i				↑ j



Parmi les 4 successeurs de ④ ⑦ et ⑧ sont déjà vus

⑤ n'a que ② comme successeur : déjà vu
② n'a pas de successeur

2) + 3) C'est une file qui induit un parcours en largeur

Dans l'algo, l'indice i indique la case du tableau où l'on va insérer un nouveau sommet (qui arrive dans la file)

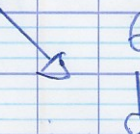
Et l'indice i la case de l'élément "à traiter" = celui qui va être servi (développé)

On crée au fur et à mesure une arborescence de parcours à travers la structure Père.

4) D'où l'intérêt de la boucle :
quand on n'arrive pas à atteindre tous les sommets et qu'il faut repartir d'une autre racine en créant un autre arbre

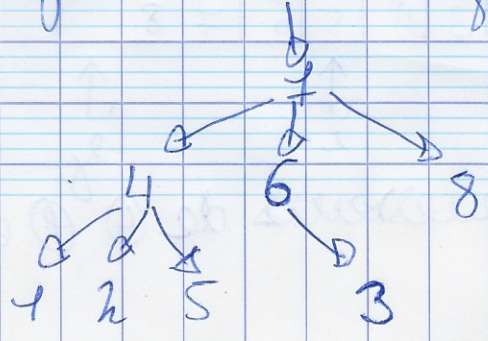
A traiter	1	2	3	4	5	8	6		

i j

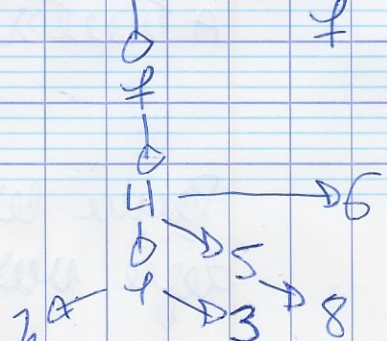


5) A partir du sommet 9 :

largeur :



profondeur :



2) 4)

```
Début
  Pour chaque sommet x faire
    couleur[x] = 0
  Fin Pour
  i = 1  j = 1  k = 0
  Pour chaque sommet x faire
    Si (couleur[x] = 0) alors
      atraiter[j] = x  couleur[x] = (i % 2) + 1
      j++
      Tant que i < j faire
        y = atraiter[i]  i++  k = couleur[y]
        Pour chaque voisin z de y
          Si (couleur[z] = 0) alors
            couleur[z] = (k % 2) + 1
            atraiter[j] = z  j++
          Sinon
            si (couleur[z] = (k % 2) + 1
              retourner faux
            Fin Si
        Fin Pour
      Fin Tant que
    Fin Si
  Fin Pour
  retourner vrai
Fin
```

2) couleur(1) = 2 couleur(6) = 1
couleur(3) = 1 couleur(2) = 2
couleur(4) = 2 couleur(5) = 2
couleur(5) = couleur(2) Faux

⟶ Non Biparti