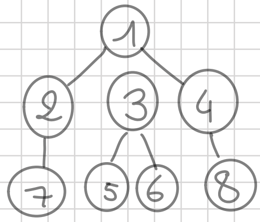
27.09.21

ACS TD2

Exercice 1

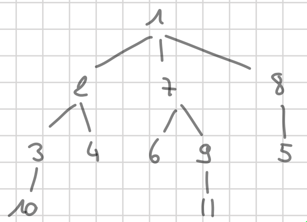
G1)

file Q	sommet u
[1]	1
[4, 3, 2]	2
[4, 3, 6, 5]	5
[4, 3, 6]	6
[4, 3, 7]	7
[4, 3]	3
[8]	4
[]	8



G2)

file Q	somme "s"
[1]	
[8, 7, 2]	1
[4, 3, 8, 7]	2
[9, 6, 4, 3, 8]	7
[5, 9, 6, 4, 3]	8
[10, 5, 9, 6, 4]	3
[10, 5, 9, 6]	4
[10, 5, 9]	6
[11, 10, 5]	9
[11, 10]	5
[11]	10
[ ]	11



Exercice 2

listes d'adjacence : $\mathcal{C} \in O(n+m)$

↳ on parcourt chaque liste d'adjacence 1 fois.

matrice d'adjacence : $\mathcal{C} \in O(n^2)$

↳ $\forall (u,v)$ = parcourir toute la ligne de u

Exercice 3

1) parcours en largeur et marquage de la distance au fur à mesure

créer (Q)
d (s) = 0
enfiler (Q, s)

tant que Q ≠ ∅
  u ← défiler Q

  ∀ (u, v) ∈ E ⟵ parcourir liste d'adjacence de u
    si d(v) = ∞
      d (v) = d (u) + 1
      enfiler (Q, v)

2) créer (Q)
d (s) = 0 , p (s) = " "
enfiler (Q, s)

tant que Q ≠ ∅
  u ← défiler Q

  ∀ (u, v) ∈ E
    si d(v) = ∞
      d (v) = d (u) + 1
      p (v) = p (u) + "u"
      enfiler (Q, v)

$O(n \times m)$ → p est un tableau de taille n
↳ au pire chaque case contient n lettres (si il faut passer par tous les autres sommets).

⇒ la fonction p contient une str d'un ⊕ court chemin.

→ même complexité que le parcours en largeur

3) À la ligne verte : p (v) = u $O(n+m)$
↳ on ne stocke que le père

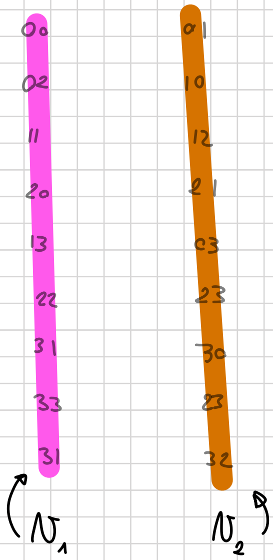
PCC (s, v)   au pire $O(n)$
  while v ≠ s :
    print v
    v = p (v)

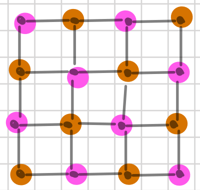
↳ on remonte le chemin de proche en proche jusqu'à retomber sur s.

Exercice 4

1) Non, on ne peut pas faire de partition. Les 3 sommets a, b, c forment un triangle, donc doivent appartenir à des parties distinctes de la partition V_1, V_2. $\Rightarrow$ IMPOSSIBLE

2) Oui





4) • parcours en largeur (BFS c̄ exo 3)

↳ à la ligne verte, on assigne une couleur (R ou O) au sommet, la couleur opposée à celle du père ~~c(x) = ! c(v)~~

• puis pour chaque sommet, on parcourt sa liste d'adjacence.

On teste si la couleur du sommet est bien différente de celle de ses voisins.

• Si la couleur est la même, on remonte le chemin des pères jusqu'à trouver un ancêtre commun pour trouver le cycle impair.

5) $$\mathcal{C}(n) \in \underbrace{O(n+m)}_{\text{assigner une couleur}} + \underbrace{O(n+m)}_{\text{comparer une couleur}} + \underbrace{2 \times O(n+m)}_{\text{trouver un cycle impair (voir ex 3 pour la justification)}} = \boxed{O(n+m)}$$

3) $\boxed{\Rightarrow}$ par l'absurde : $\exists$ cycle impair $\Rightarrow$ IMPOSSIBLE

↳ car un cycle impair n'est pas biparti.

$\boxed{\Leftarrow}$ idée : Mq si $\nexists$ cycle impair, alors algo pour [?] coloration

↳ faire avec parcours en largeur...