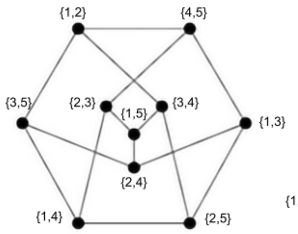
commentaires enseignant
global : bon rapport sf certains point logiques
exercice 1 : ok
exercice 2 : algos ok mais conclusions a revoir
exercice 3 : ok
exercice 4 : ok
exercice 5 : preuves très incomplètes

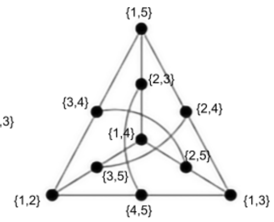
RAPPORT pour Denis LAPOIRE

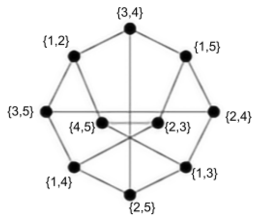
Séance numéro : 2
Nom du groupe : G4 I1
Noms des auteurs : Mouad BOUMOUR
Nolan BIZON
Lucas GAVÉRIAUX

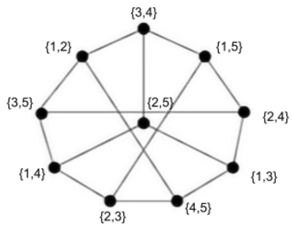
+++
Exercice 1
Auteur principal : Lucas GAVÉRIAUX
Contributeur(s) : Nolan BIZON, Mouad BOUMOUR
+++

Chacun des dessins fournis convient. Pour preuve ci-dessous des positionnements possibles des sommets :









ok

+++
Exercice 2
Question 1
Auteur principal : Mouad BOUMOUR
Contributeur(s) : Nolan BIZON, Lucas GAVÉRIAUX
+++

1.

```
SONT_ISOMORPHES(G1:graphe, G2:graphe): booléen
   Retourner AdmetDessinSansNom( G1,effaceNoms( graphe2gessin(G2)
))
```

Donc si le problème `admetDessinSansNom` admet une solution en temps polynomial alors `SONT_ISOMORPHES` aussi.
ok

2. Tant que les sous-fonctions intermédiaires utilisées ayant une complexité linéaire et le fait que la fct AdmetDessinSansNom résout le problème de la fonction SONT_ISOMORPHES, alors la fonction AdmetDessinSansNom a une complexitée inférieure ou égale à celle de la fonction SONT_ISOMORPHES.non c'est le contraire

3.
```
AdmetDessinSansNom(G:graphe, D:dessinSansNom): booléen
    Retourner SONT_ISOMORPHES( G1, dessin2graphe( Nommage(D) ) )
```
ok

4. De même or la fonction SONT_ISOMORPHES résout le problème de la fonction ADmetDessinSansNom donc la complexité optimale en temps de AdmetDessinSansNom peut être inférieure ou égale à celle de SONT_ISOMORPHES.

   On conclut que les deux problèmes algorithmiques ~~fonctions~~ SONT_ISOMORPHES et ADmetDessinSansNom ayant même complexité ~~optimale~~ en temps car chacune résout l'autre.
   ok

++

Exercice 3

Auteur principal : Nolan BIZON

Contributeur(s) : Lucas GAVÉRIAUX, Mouad BOUMOUR

++

1. Nous pouvons représenter sur machine très simplement toute fonction f : $V_G \rightarrow [1,3]$ sous forme d'un tableau d'entier. ex : [1, 2 , 1 , 3 , … , 2, 1, 3]

ok

2. Il existe 3^n fonctions f : $V_G \rightarrow [1,3]$ dans un graphe G d'ensemble de sommets V_G = [1,n].

ok

3. On peut représenter chaque fonction f : $V_G \rightarrow [1,3]$ par un chiffre entre 0 et 3^n-1 en utilisant un code d'écriture simple :

   111…1 → 0
   111…2 → 1
   111…3 → 2

111…21 → 3
111…22 → 4
111…23 → 5
….
333…32 → 3^n-2
333…33 → 3^n-1

ok

4. Solution algorithmique :

```
fonction est3colorable( G : graphe )
début
        Pour i de 0 à 3ⁿ-1 :  Faire
                Si est3coloration( G, i )  == Vrai :
                        retourner Vrai
        retourner Faux
Fin
```

ok

5. La complexité en temps de notre algorithme est $O(3^n)$ soit exponentielle.

ok

+++

Exercice 4

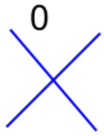
Question 1

Auteur principal : Lucas GAVÉRIAUX

Contributeur(s) : Nolan BIZON, Mouad BOUMOUR

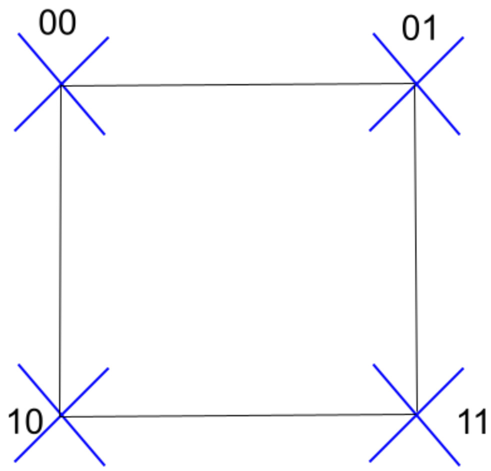
+++

H0 : sommets dans {0} :

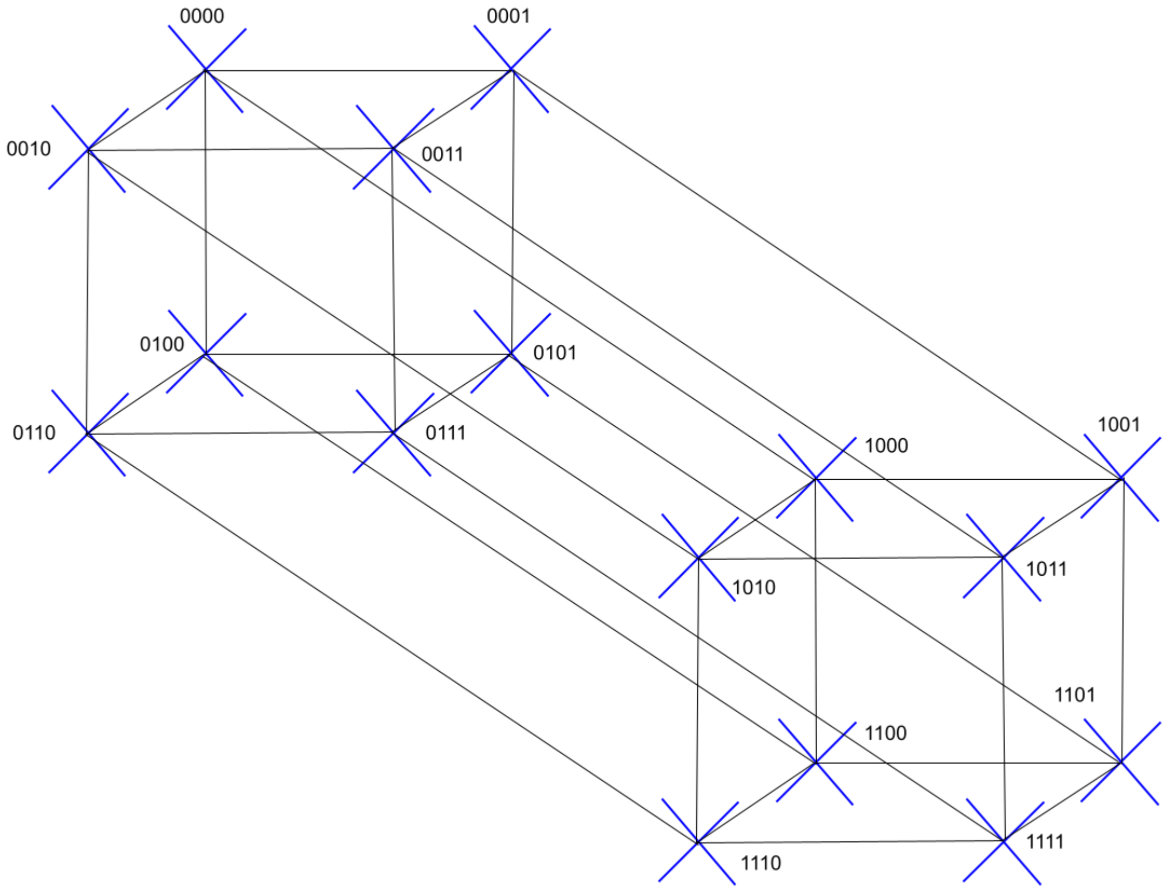


H1 : sommets dans [0,1] :

H2 : sommets dans [0,3] :



H4 : sommets dans [0,16] :



ok

++
Exercice 4
Question 2
Auteur principal : Lucas GAVÉRIAUX
Contributeur(s) : Nolan BIZON, Mouad BOUMOUR
++

La distance entre deux sommets correspond au nombre de bits différents pour ces sommets. Par exemple, pour les sommets suivants :

0101000111
0110100010

On compte une différence sur 5 bits. La distance entre ces deux sommets vaut donc 5.

ok

++

Exercice 4

Question 3

Auteur principal : Lucas GAVÉRIAUX

Contributeur(s) : Nolan BIZON, Mouad BOUMOUR

++

Pour le même exemple qu'à la question 2, on peut fournir le chemin de longueur 5 suivant :

0101000111
|
0111000111
|
0110000111
|
0110100111
|
0110100011
|
0110100010

ok

++

Exercice 4

Question 4

Auteur principal : Lucas GAVÉRIAUX

Contributeur(s) : Nolan BIZON, Mouad BOUMOUR

++

Nous proposons un algorithme qui compte le nombre de bits différents entre deux entiers passés en paramètre. Pour ce faire, on peut calculer le OU EXCLUSIF (noté ^) entre ces deux entiers, puis compter le nombre de bits à 1.

  0101000111
^ 0110100010
= 1100011010

Fonction nb_bits_1(x : entier) : entier

Début

compteur←0

Tant que (x >= 0) *Alors*

Si (x modulo 2 == 1) *Alors*

compteur←compteur + 1

x←x-1

x←x/2

Retourner compteur

Fin

Fonction dist_2_sommets(x,y : entiers) : entier

Début

Retourner nb_bits_1(x^y)

Fin

ok

++

Exercice 5

Question 1

Auteur principal : Lucas GAVÉRIAUX

Contributeur(s) : Mouad BOUMOUR, Nolan BIZON

++

Démonstration de l'implication 1⇒2 :

Soit G = (E_g, V_g, f_g) un arbre. Alors, par définition d'un arbre, **G est connexe.**

Ainsi, pour tout couple de sommet (s,t) dans V_g, il existe AU MOINS un chemin allant de s à t.

Cependant, G est également **acyclique**. Ainsi, il existe AU PLUS un chemin allant de s à t.
a prouver !

Autrement dit, pour tout couple de sommet (s,t), il existe EXACTEMENT 1 chemin allant de s à t. **Il suffit donc de supprimer une arête sur ce chemin pour que G ne soit plus connexe**.
a prouver !

Ceci étant vrai pour tout couple de sommets (s,t), on a démontré que si G est un arbre, alors G est **connexe** et est **minimal à vérifier cette propriété.**

++

Exercice 5

Question 2

Auteur principal : Lucas GAVÉRIAUX

Contributeur(s) : Mouad BOUMOUR, Nolan BIZON

++

Démonstration de l'implication 6⇒1 :

Soit G = (E_g, V_g, f_g) un graphe tel que tout couple de sommets (s,t) admet un unique chemin simple allant de s à t.

Autrement dit, tout couple (s,t) admet un chemin allant de s à t. Donc, par définition, **G est connexe**.

De plus, ce chemin est unique, donc pour tout couple (s,t), il n’existe pas de cycle simple d’origine s passant par t. Donc **G est acyclique**.

a prouver !

On a alors démontré que si, pour tout couple de sommets (s,t) dans G, il existe un unique chemin allant de s à t, alors **G est un arbre**.