Examen 2I003

Vendredi 19 Janvier 2018, 2 heures
aucun document autorisé

Exercice 1 – Un exemple à ne pas suivre

Dans cet exercice, les seuls caractères considérés sont des lettres en minuscule. On dit qu'une chaîne de caractères est un *palindrome* si elle reste la même, qu'on la lise de gauche à droite ou de droite à gauche.
Par exemple :
– le mot vide est un palindrome, tout mot réduit à une seule lettre est un palindrome,
– "noyon", "elle", "esoperesteicietserepose", "eibohpphobie" sont des palindromes,
– "maman", "papa", "marmelade" n'en sont pas.
Soit $m = m_0 \dots m_{n-1}$ un mot de longueur n. On appelle *sous-mot* du mot m tout mot de la forme $m_{i_1} \dots m_{i_k}$ avec $0 \leq i_1 < i_2 < \dots < i_k \leq n-1$.
Par exemple : "noel " est un sous-mot de "nouvelan" ($i_1 = 0, i_2 = 1, i_3 = 4, i_4 = 5$), mais "babar" n'est pas un sous-mot de "barbapapa".

Dans cet exercice nous voulons trouver, pour un mot m donné, un sous-mot de m qui soit le plus long palindrome possible.
Par exemple, pour "noyon", la réponse est "noyon", mais pour "marmelade", il y a plusieurs réponses possibles : "mam", "ara", "mrm", etc. Dans ce cas, notre algorithme renverra une seule des possibilités.

Préliminaires

Rappels :
— la numérotation des positions dans une chaîne de caractères commence à 0 ;
— la fonction `len` renvoie la longueur d'un mot (`len(m)=n`, par exemple : `len("babar")=5`).

On considère les deux fonctions :

```
def est_palindrome_rec(m, d, f):
    if f <= d:
        return True
    if m[d] != m[f]:
        return False
    return est_palindrome_rec(m,d + 1, f - 1)
```

et

```
def est_palindrome(m):
    return est_palindrome_rec(m, 0, len(m) - 1)
```

où m est une chaîne de caractères, d et f des entiers naturels.

Question 1

1. Quelle est la valeur de `est_palindrome_rec("camarade", 1, 3)` ?
   de `est_palindrome_rec("barbapapa", 0, 4)` ?
2. Quelle est la valeur de `est_palindrome("noyon")` ?

Solution:

1. `est_palindrome_rec("camarade", 1, 3) : True`
   `est_palindrome_rec("barbapapa", 0, 4) : False`
2. `est_palindrome("noyon") : True`

Question 2

1. Montrer que, pour $d, f \in \{0, \dots n-1\}$ tels que $f \geq d-1$, `est_palindrome_rec(m, d, f)` se termine et renvoie la valeur `True` si $m_d \dots m_f$ est un palindrome, et la valeur `False` sinon.
   Indication. Faire un raisonnement par récurrence forte sur $k = f - d + 1$ (longueur de $m_d \dots m_f$), en prenant comme cas de base $k \leq 1$.
2. En déduire que `est_palindrome(m)` renvoie la valeur `True` si m est un palindrome, et la valeur `False` sinon.

Solution:

1. **Base** Si $k = f - d + 1 \leq 1$ alors $f \leq d$ donc `est_palindrome_rec(m, d, f)` se termine et renvoie `True`, ce qui est correct car $m_d \dots m_f$ est soit le mot vide, soit un mot réduit à une lettre.
   Induction Soit $k > 1$, supposons que la propriété soit vraie pour tout $j < k$.
   Si $m_d \neq m_f$ alors `est_palindrome_rec(m,d,f)` se termine et renvoie `False`, ce qui est correct car $m_d \dots m_f$ n'est pas un palindrome.
   Si $m_d = m_f$ alors `est_palindrome_rec(m,d,f)` fait un appel récursif à `est_palindrome_rec(m,d+1,f-1)`, qui se termine. Donc `est_palindrome_rec(m,d,f)` se termine. De plus, `est_palindrome_rec(m,d,f)` renvoie le résultat de `est_palindrome_rec(m,d+1,f-1)`. Comme $(f-1) - (d+1) = k - 2 < k$, on utilise l'hypothèse de récurrence :
   – si `est_palindrome_rec(m,d+1,f-1)` renvoie `True` alors $m_{d+1} \dots m_{f-1}$ est un palindrome et $m_d \dots m_f$ est aussi un palindrome puisque $m_d = m_f$;
   – si `est_palindrome_rec(m,d+1,f-1)` renvoie `False` alors $m_{d+1} \dots m_{f-1}$ n'est pas un palindrome et $m_d \dots m_f$ n'est pas un palindrome lui non plus.
   Dans les deux cas, `est_palindrome_rec(m,d,f)` renvoie le résultat correct.
   Conclusion La propriété est vraie pour tout k.
2. `est_palindrome(m)` renvoie la même valeur que `est_palindrome_rec(m, 0, len(m) - 1)`, c'est-à-dire la valeur `True` si $m = m_0 \dots m_{len(m)-1}$ est un palindrome, et la valeur `False` sinon.

Question 3

Calculer la complexité de `est_palindrome` dans le meilleur cas et dans le pire cas, en précisant quel est le meilleur cas et quel est le pire cas pour un mot de longueur n.

Solution:

La complexité de `est_palindrome` est en $\Omega(1)$ dans le meilleur cas et en $O(n)$ dans le pire cas.
Meilleur cas : la première lettre du mot est différente de la dernière.
Pire cas : le mot est un palindrome.

Un premier algorithme, très lent

Voici un premier algorithme permettant, étant donné un mot m, de calculer un plus long sous-mot de m qui soit un palindrome. Celui-ci consiste à calculer tous les sous-mots de m, à tester pour chacun d'eux s'il est un palindrome (en utilisant la fonction `est_palindrome`) et à retenir le plus long (ou l'un des plus longs, s'il y en a plusieurs).

Question 4

Combien y-a-t-il de sous-mots pour un mot de longueur n ?

Solution:

Il y en a 2^n.

On admet que la complexité du calcul d'un sous-mot d'un mot m de longueur n est en $\Theta(n)$.

Question 5

En déduire la complexité de l'algorithme proposé, pour un mot de longueur n.

Solution:

Pour un mot de longueur n, la complexité du calcul d'un sous-mot est en $\Theta(n)$, tester si ce sous-mot est un palindrome a une complexité en $O(n)$ et il y a 2^n sous-mots à calculer, on a donc une complexité totale en $\Theta(n2^n)$.

Un deuxième algorithme, un peu moins lent

On considère la fonction :

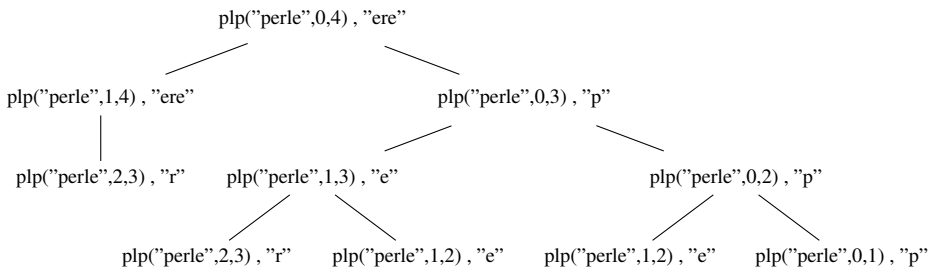
```
def plp(m, d, f):
    if f < d:
        return ""
    if f == d:
        return m[d]
    if f == d + 1:
        if m[d] == m[f]:
            return m[d] + m[f]
        return m[d]
    if m[d] == m[f]:
        return m[d] + plp(m, d + 1, f - 1) + m[f]
    s1 = plp(m, d + 1, f)
    s2 = plp(m, d, f - 1)
    if len(s1) > len (s2):
        return s1
    return s2
```

où m est une chaîne de caractères, d et f des entiers naturels tels que $d, f \in \{0, \dots n-1$ et $f \geq d-1$.

Question 6

Dessiner un arbre des appels pour `plp("perle",0,4)` et préciser les sous-mots retournés par chaque appel récursif.

Solution:



Question 7

On note $c(k)$ le nombre de concaténations effectuées par `plp(m,d,f)`, avec $k = f - d + 1$.

1. Cas de base : calculer $c(0)$, $c(1)$ et montrer que $c(2) \leq 1$.
2. Montrer que $c(k)$ vaut $2 + c(k-2)$ ou bien $2c(k-1)$.
3. Montrer, par récurrence sur k, que $c(k) \leq 2^k$.

4. En déduire la complexité de `plp(m,0,len(m)-1)` dans le pire cas.
5. Si le mot m est un palindrome, quelle est la valeur de $c(k)$ dans tous les appels récursifs ? Quelle est la complexité de `plp(m,0,len(m)-1)` dans le meilleur cas ?

Solution:

1. $c(0) = 0$, $c(1) = 0$.
   $c(2) = 1$ si $m[d] = m[d+1]$ et $c(2) = 0$ sinon, donc $c(2) \leq 1$.
2. $c(k)$ vaut $c(k-2)+2$ si $m[d] = m[f]$ et $c(k) = 2c(k-1)$ sinon.
3. **Base** C'est vrai pour $k = 0$, $k = 1$, $k = 2$.
   Induction Soit $k > 2$, supposons que $c(j) \leq 2^{j-1}$ pour tout $j < k$.
   Si $m[d] \neq m[f]$ alors $c(k) = 2c(k-1) \leq 2 * 2^{k-1}$ donc $c(k) \leq 2^k$.
   Si $m[d] = m[f]$ alors $c(k) = c(k-2)+2 \leq 2^{k-2}+2 \leq 2^{k-2}+2^{k-2}$ (en effet, pour $k > 2$, on a $k-2 \geq 1$ donc $2^{k-2} \geq 2$), donc $c(k) \leq 2^{k-1} < 2^k$.
   Conclusion La propriété est vraie pour tout $k \geq 0$.
4. La complexité de `plp(m,0,len(m)-1)` est égale à $c(n)$, elle est donc en $O(2^n)$ dans le pire cas.
5. Si le mot m est un palindrome alors $c(k)$ vaut $c(k-2)+2$ dans tous les appels récursifs et il y a $\lfloor \frac{n}{2} \rfloor$ appels récursifs. C'est le meilleur cas. La complexité dans le meilleur cas est donc en $\Omega(n)$.

Question 8

Montrer que la fonction `plp(m,d,f)` se termine, par récurrence forte sur $k = f - d + 1$, pour tout $k \geq 0$.

Solution:

Si $k \leq 2$, la fonction `plp(m, d, f)` se termine. Si $k > 2$, la fonction `plp(m, d, f)` fait un appel à `plp(m, d+1, f-1)`, ou bien à `plp(m, d+1, f)` et `plp(m, d, f-1)`, qui se terminent donc `plp(m, d, f)` se termine.

Question 9 – Bonus

Soit la propriété :

$\mathcal{P}(k)$: `plp(m, d, f)` calcule un sous-mot de $m_d \dots m_f$ qui est le plus long palindrome possible.

On veut montrer, par récurrence sur $k = f - d + 1$, que la propriété $\mathcal{P}(k)$ est vraie pour tout $k \leq n$.

1. Cas de base : montrer que $\mathcal{P}(k)$ est vraie pour $k \leq 2$.
2. Induction : soit $k > 2$, on suppose que $\mathcal{P}(j)$ est vraie pour tout $j < k$, montrer que $\mathcal{P}(k)$ est vraie.
   Indication. On distinguera deux cas : le cas où m_d est égal à m_f et le cas où m_d est différent de m_f.
3. Conclure.

Solution:

1. Si $k \leq 0$ alors le mot $m_d \dots m_f$ est vide, son plus long sous-mot palindrome est le mot vide, c'est bien le résultat de `plp(m, d, f)`. Si $k = 1$ alors le mot $m_d \dots m_f$ a une seule lettre m_d, son plus long sous-mot palindrome est m_d, c'est bien le résultat de `plp(m, d, f)`. Si $k = 2$ alors le mot $m_d \dots m_f$ a deux lettres ab. Si $a = b$ alors son plus long sous-mot palindrome est aa, c'est bien le résultat de `plp(m, d, f)`. Sinon, le plus long sous-mot palindrome n'a qu'une seule lettre, par exemple a, c'est bien le résultat de `plp(m, d, f)`.
2. Soit $k > 2$, on suppose que $\mathcal{P}(j)$ est vraie pour tout $j < k$.
   — cas où m_d est égal à m_f. Notons a cette lettre. Notons p le mot `m[d]+plp(m,d+1,f-1)+m[f]` et p' le mot `plp(m,d+1,f-1)`, alors $p = ap'a$ et p' est égal au plus long sous-mot palindrome de $m_{d+1} \dots m_{f-1}$, par hypothèse de récurrence.
   Soit q un sous-mot palindrome de $m_d \dots m_f$. Si q ne commence pas par a alors q ne se termine pas par a, donc q est un sous-mot palindrome de $m_{d+1} \dots m_{f-1}$, sa longueur est inférieure ou égale à celle de p', et donc à celle de p. Si q commence par a alors q se termine par a, il s'écrit donc $aq'a$ où q' est un sous-mot palindrome de $m_{d+1} \dots m_{f-1}$. La longueur de q' est donc inférieure ou égale à celle de p'. Par conséquent la longueur de $q = aq'a$ est inférieure ou égale à celle de $p = ap'a$. p est donc le plus long sous-mot palindrome de $m_d \dots m_f$.

— cas où m_d est différent de m_f, alors on ne peut avoir simultanément m_d et m_f dans un plus long sous-mot palindrome de $m_d \dots m_f$. Notons p le plus long sous-mot palindrome de $m_d \dots m_f$ alors p est soit un sous-mot de $m_d \dots m_{f-1}$, soit un sous-mot de $m_{d+1} \dots m_f$. C'est donc le plus long entre le plus long sous-mot palindrome de $m_d \dots m_{f-1}$ (c'est-à-dire `plp(m,d,f-1)` par hypothèse de récurrence), et le plus long sous-mot palindrome de $m_{d+1} \dots m_f$ (c'est-à-dire `plp(m,d+1,f)` par hypothèse de récurrence). p est bien le résultat qui est calculé par `plp(m,d,f)`.

3. On a montré par récurrence forte que `plp(m,d,f)` calcule un sous-mot de $m_d \dots m_f$ qui est le plus long palindrome possible.

Question 10

En déduire que `plp(m,0,len(m)-1)` calcule un sous-mot de m qui est le plus long palindrome possible.

Solution:

`plp(m,0,len(m)-1)` calcule un sous-mot de $m_0 \dots m_{len(m)-1}$, c'est-à-dire de m, qui est le plus long palindrome possible.

Épilogue

Ces deux algorithmes sont à prohiber car leurs complexités sont beaucoup trop élevées. On peut faire beaucoup mieux en utilisant la *programmation dynamique* [1], qui permet d'écrire un algorithme en $\Theta(n^2)$ (où n est la longueur du mot).

Exercice 2 – Graphes bipartis

Dans cet exercice, $G = (V, E)$ désigne un graphe **non orienté connexe** possédant au moins deux sommets. n désigne le nombre de sommets, et m le nombre d'arêtes.

Si H est un ensemble, on rappelle qu'une partition est une suite de sous-ensembles $H_0, \cdots, H_k$ non vides de H telle que $H = \cup_{i=0}^{k} H_i$ et $\forall (i,j) \in \{1, \cdots, k\}^2$ tels que $i \neq j$, $H_i \cap H_j = \emptyset$.

Un graphe $G = (V, E)$ est biparti si il existe une partition des sommets en deux ensembles V_0 et V_1 tels que toute arête $e = \{x, y\} \in E$ possède un sommet dans V_0 et un sommet dans V_1. On dira alors que V_0 et V_1 forment une bipartition du graphe.

Un cycle est impair (pair) si il contient un nombre impair (pair) d'arêtes. On considère qu'un sommet ne forme pas à lui seul un cycle.

Soient également les graphes $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ et $G_3 = (V_3, E_3)$ définis par les ensembles de sommets $V_1 = V_2 = V_3 = \{1, 2, 3, 4, 5, 6\}$ et les ensembles d'arêtes $E_1 = \{\{1,4\}, \{1,5\}, \{1,6\}, \{2,4\}, \{3,4\}\}$, $E_2 = \{\{1,2\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}, \{3,5\}, \{4,6\}, \{5,6\}\}$ et $E_3 = \{\{1,2\}, \{1,5\}, \{2,3\}, \{2,4\}, \{3,5\}, \{4,6\}\}$.

Question 1

1. Donnez la définition d'un graphe non orienté connexe. Donnez la définition d'un arbre.
2. Est-ce-que les graphes G_1, G_2, G_3 sont connexes ? Est-ce-que ce sont des arbres ? Justifiez votre réponse.
3. Est-ce-que les graphes G_1, G_2 et G_3 sont bipartis ? Donnez le cas échéant une bipartition.

Solution:

1. Un graphe est connexe si pour tout couple de sommets, il existe une chaîne qui les relie. Un arbre est un graphe connexe sans cycle.
2. Les 3 graphes sont connexes. G_1 est sans cycle, c'est donc un arbre. G_2 et G_3 possèdent des cycles, ce ne sont donc pas des arbres.

1. La programmation dynamique est étudiée dans l'UE d'Algorithmique de L3

3. G_1 est biparti : $V_0 = \{1, 2, 3\}$ et $V_1 = \{4, 5, 6\}$ forme une bipartition. G_2 n'est pas biparti. On ne peut pas former une bipartirion a cause du cycle $2, 3, 4, 2$. G_3 est biparti : $V_0 = \{1, 3, 4\}$ et $V_1 = \{2, 5, 6\}$ forme une bipartition.

Question 2

1. Donnez la matrice sommet-sommet du graphe G_1. Donnez la représentation de G_1 sous forme de listes d'adjacences.
2. Quel est l'ordre de grandeur de la taille de la représentation sommet-sommet et par liste d'adjacences d'un graphe G ? Quelle est la relation entre le nombre de sommets et le nombres d'arêtes d'un arbre ? Que deviennent alors les ordres de grandeur de la taille de la représentation sommet-sommet et par liste d'adjacences si G est un arbre ?

Solution:

1. La matrice sommet-sommet est :
$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$
La représentation par listes d'adjacences est $L[1] = [4, 5, 6]$, $L[2] = [4]$, $L[3] = [4]$, $L[4] = [1, 2, 3]$, $L[5] = [1]$ et $L[6] = [1]$.
2. La taille d'une matrice sommet-sommet est en $\Theta(|V|^2)$. La taille par liste d'adjacence est en $\Theta(\max(|V|, |E|))$. Si G est un arbre, $|E| = |V| - 1$. La taille d'une matrice sommet-sommet reste en $\Theta(|V|^2)$. La taille par liste d'adjacence est en $\Theta(|V|)$.

Question 3

Montrez par récurrence faible sur le nombre de sommets que tout arbre possèdant au moins deux sommets est biparti. On admet que tout arbre possède au moins un sommet de degré 1 qui correspond à une feuille.

Solution:

$\Pi(n)$, $n \geq 2$: tout arbre $G = (V, E)$ de n sommets est biparti.

On démontre $\Pi(n)$ par récurrence faible.

Base : Si $n = 2$, le graphe possède une seule arête $e = \{x, y\}$. En posant $V_0 = \{x\}$ et $V_1 = \{y\}$, on obtient une bipartition de G. $\Pi(2)$ est donc vérifiée.

Induction : Soit $n - 1 \geq 2$ telle que $\Pi(n - 1)$ soit vérifiée. Soit alors un arbre G de n sommets et soit f une feuille de G. Si on supprime f, on obtient un arbre G' de $n - 1$ sommets. Par hypothèse de récurrence, $G' = (V - \{f\}, E)$ est bi-parti. Soit alors x le sommet de G' telle que l'arête $\{x, f\} \in E$ et soit V_0' et V_1' la partition des sommets correspondant.

Si $x \in V_0'$, on pose alors $V_0 = V_0'$ et $V_1 = V_1' \cup \{f\}$. De même, si $x \in V_1'$, on pose alors $V_1 = V_1'$ et $V_0 = V_0' \cup \{f\}$. Dans les deux cas, on obtient une bi-partition de V. On en déduit que G est bi-parti, et donc $\Pi(n)$ est vérifié.

Conclusion : La propriété est vérifiée par récurrence faible sur n.

Question 4

Soit G un graphe connexe. Démontrez que si G est biparti, alors tous les cycles de G sont pairs. Indication : faire un raisonnement direct à partir d'une bipartition du graphe.

Solution:

Si G est biparti, alors il existe une bipartition du graphe V_0 et V_1. Tout circuit c est composé de manière alternative de sommets dans V_0 et dans V_1. c est donc forcément pair.

Question 5

Soit G un graphe connexe et soit x un sommet de $G = (V, E)$. Pour tout sommet $y \in V$, on note $\delta(y)$ la valeur minimale en nombre d'arêtes d'une chaîne de x à y. Pour toute valeur $k \in \{0, \cdots, n-1\}$, on note également D_k l'ensemble des sommets y de G tels que $\delta(y) = k$.

Dans cette question, on considère le graphe $G_3 = (V_3, E_3)$.

1. Calculez les valeur $\delta(y)$, $y \in V_3$ pour le sommet $x = 1$. En déduire les ensemble D_k, $k \in \{1, \cdots 5\}$.
2. Calculez $V_0 = \cup_{k=0,kpair}^{n} D_k$ et $V_1 = \cup_{k=0,kimpair}^{n} D_k$. Qu'observez vous ?

Solution:

1. On obtient $\delta(1) = 0$, $\delta(2) = 1$, $\delta(3) = 2$, $\delta(4) = 2$, $\delta(5) = 1$ et $\delta(6) = 3$. On en déduit que $D_0 = \{1\}$, $D_1 = \{2, 5\}$, $D_2 = \{3, 4\}$, $D_3 = \{6\}$ et $D_4 = D_5 = \emptyset$.
2. $V_0 = D_0 \cup D_2 \cup D_4 = \{1, 3, 4\}$ et $V_1 = D_1 \cup D_3 \cup D_5 = \{2, 5, 6\}$. C'est une bipartition de G.

Question 6

Soit G un graphe connexe.

1. Montrez que les ensembles $D_k \neq \emptyset$, $k \in \{1, \cdots, n-1\}$ forment une partition de V
2. On suppose que G ne possède pas de cycle impair. Montrez par l'absurde que les ensembles $V_0 = \cup_{k=0,kpair}^{n} D_k$ et $V_1 = \cup_{k=0,kimpair}^{n} D_k$ forment une bipartition de G.
3. En déduire que G est biparti si et seulement si tous ses cycles sont pairs.

Solution:

1. Pour tout sommet $y \in V$, $\delta(y) \in \{0, \cdots n-1\}$ et ainsi, $y \in D_{\delta(y)}$. Ainsi, pour toutes valeurs $k \neq k'$, $D_k \cap D_{k'} = \emptyset$. De plus, tout sommet $y \in V$ possède une valeur $\delta(y) \in \{0, \cdots, n-1\}$, donc $V = \cup_{k=0}^{n} D_k$.
2. Supposons que les ensembles V_0 et V_1 ne forment pas une bipartition. Il existe une arête $e = \{y, z\}$ avec les deux sommets y et z dans S_0 ou dans S_1.

   Supposons que y et z sont dans V_0. Alors, il y a une chaîne dans G de longueur paire de x à y et de x à z. En rajoutant l'arête $\{y, z\}$, on obtient un cycle impair, ce qui est contradictoire.

   Supposons que y et z sont dans V_1. Alors, il y a une chaîne dans G de longueur impaire de x à y et de x à z. En rajoutant l'arête $\{y, z\}$, on obtient un cycle impair, ce qui est encore contradictoire.

   Donc, on en déduit que les ensembles V_0 et V_1 forment une bi-partition.
3. Ainsi, on a montré que, tout graphe G sans cycle impair est biparti. Or, on a montré dans la question 3 que si G est biparti, tous ses cycles sont pairs. On a donc montré la propriété.

Question 7

Soit x un sommet fixé de G. On suppose que, pour tout $y \in V$, on a calculé $\delta(y)$.

1. Décrire un algorithme qui calcule V_0 et V_1.
2. En déduire un algorithme qui détermine si un graphe est biparti.

Solution:

1. On parcourt les valeurs $\delta(y)$. Si $\delta(y)$ est pair, on rajoute y à V_0 sinon on rajoute y à V_1.
2. Il suffit de vérifier si les ensembles V_0 et V_1 forment une bipartition : on vérifie que toutes les arêtes de E ont une extrémité dans V_1 et une extrémité dans V_0. Si c'est le cas, le graphe est biparti. Sinon, il ne l'est pas.