Numéro d'anonymat :

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")` ?

Question 2

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

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.

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 ?

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.

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 $d, f \in \{0, \ldots, n-1\}$ 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.

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 ?

Question 8
Montrer que la fonction `plp(m,d,f)` se termine, par récurrence forte sur $k = f - d + 1$, pour tout $k \geq 0$.

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.

Question 10
En déduire que `plp(m,0,len(m)-1)` calcule un sous-mot 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. La programmation dynamique est étudiée dans l'UE d'Algorithmique de L3

$\{\{1,4\},\{1,5\},\{1,6\},\{2,4\},\{3,4\}\}$, $E_2 = \{\{1,2\},\{1,4\},\{2,3\},\{2,4\},\{3,4\},\{3,5\},\{4,6\},\{5,6\}\}$ 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.

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 listes 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 listes d'adjacences si G est un arbre ?

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.

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.

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 ?

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.

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.