CI 2 : Algorithmique & Programmation

Algorithmes d'Informatique

1 Recherches dans une liste

1.1 Recherche d'un nombre dans une liste

```python
def is_number_in_list(nb,tab):
    """Renvoie True si le nombre nb est dans la liste de nombres tab

    Keyword arguments:
    nb -- nombre entier
    tab -- liste de nombre entiers

    """
    i=0
    while i<len(tab) and tab[i]!=nb:
        i+=1
    return i<len(tab)
```

1.2 Recherche du maximum dans une liste de nombre

```python
def what_is_max(tab):
    """ Renvoie le plus grand nombre d'une liste

    Keyword arguments:
    tab -- liste de nombres

    """
    i=1
    maxi=tab[0]
    while i<len(tab):
        if tab[i]>maxi:
            maxi=tab[i]
        i+=1
    return maxi
```

1.3 Recherche par dichotomie dans un tableau trié

```python
def is_number_in_list_dicho(nb,tab):
    """ Renvoie l'index si le nombre nb est dans la liste de nombres tab.
        Renvoie None sinon.

    Keyword arguments:
    nb -- nombre entier
    tab -- liste de nombres entiers triés

    """
    g, d = 0, len(tab)-1
    while g <= d:
        m = (g + d) // 2
        if tab[m] == nb:
```

```python
            return m
        if tab[m] < nb:
            g = m+1
        else:
            d = m-1
    return None
```

2 Gestion d'une liste de nombres

2.1 Calcul de la moyenne

```python
def calcul_moyenne(tab):
    """ Renvoie la moyenne des éléments d'un tableau.

    Keyword arguments:
    tab -- liste de nombres entiers triés

    """
    res = 0
    for i in range(len(tab)):
        res = res+tab[i]
    return res/(len(tab))
```

2.2 Calcul de la variance

2.3 Calcul de la médiane

3 Chaînes de caractères

3.1 Recherche d'un mot dans une chaîne de caractères

```python
def index_of_word_in_text(mot, texte):
    """ Recherche si le mot est dans le texte.
    Renvoie l'index si le mot est présent, None sinon.

    Keyword arguments:
    mot -- mot recherché
    texte -- texte

    """
    for i in range(1 + len(texte) - len(mot)):
        j = 0
        while j < len(mot) and mot[j] == texte[i + j]:
            j += 1
        if j == len(mot):
            return i
    return None
```

Estimation de la complexité

4 Calcul numérique

4.1 Recherche du zéro d'une fonction continue monotone par la méthode de dichotomie

Pseudo Code

```
Début Fonction
    Données : f, a, b, ε
    g ← a
    d ← b
    f_g ← f(g)
    f_d ← f(d)
    tant que (d − g) > 2ε faire
        m ← (g + d)/2
        f_m ← f(m)
        si f_g · f_m ≤ 0 alors
            d ← m
            f_d ← f_m
        sinon
            g ← m
            f_d ← f_m
        fin
    fin
    retourner (g + d)/2
Fin
```

python

```python
def rechercheDichotomique(f,a,b,eps):
    g = a
    d = b
    f_g = f(g)
    f_d = f(d)
    while (d-g) > 2*eps:
        m = (g+d)/2
        f_m =f(m)
        if f_g * f_m <= 0 :
            d = m
            f_d = f_m
        else :
            g  = m
            f_d = f_m
    return (g+d)/2
```

Précision du calcul

Rapidité

Comparaison à zéro

4.2 Recherche du zéro d'une fonction continue monotone par la méthode de Newton

Pseudo Code

Début Fonction
  Données : f, f', a, ε
  $g \leftarrow a$
  $c \leftarrow g - \dfrac{f(g)}{f'(g)}$
  tant que $|c - g| > \varepsilon$ **faire**
    $g \leftarrow c$
    $c \leftarrow c - \dfrac{f(c)}{f'(c)}$
  fin
  retourner c
Fin

Précision du calcul

Rapidité

4.3 Méthode des rectangles pour le calcul approché d'une intégrale sur un segment

4.4 Méthode des trapèzes pour le calcul approché d'une intégrale sur un segment

4.5 Méthode d'Euler pour la résolution d'une équation différentielle

Complexité algorithmique

5 Algorithmes de tris

5.1 Tri par insertion

5.2 Tri rapide «Quicksort»

5.3 Tri fusion

6 Algorithmes classiques

6.1 Division euclidienne

Pseudo Code

Data : $a, b \in \mathbb{N}^*$
reste $\leftarrow$ a
quotient $\leftarrow 0$
tant que *reste* $\geq b$ **faire**
  reste $\leftarrow$ reste $- b$
  quotient $\leftarrow$ quotient $+ 1$
fin
Retourner quotient,reste

6.2 Algorithme d'Euclide

Cet algorithme permet de calculer le PGCD de deux nombres entiers. Il se base sur le fait que si a et b sont deux entiers naturels non nuls, $pgcd(a,b) = pgcd(b, a \bmod b)$.

Pseudo Code

Data : $a, b \in \mathbb{N}^*$
$x \leftarrow a$
$y \leftarrow b$
tant que $y \neq 0$ **faire**
  $r \leftarrow$ reste de la division euclidienne de x par y
  $x \leftarrow y$
  $y \leftarrow r$
fin
Afficher x.

Codage en Pythonde l'algorithme d'Euclide :

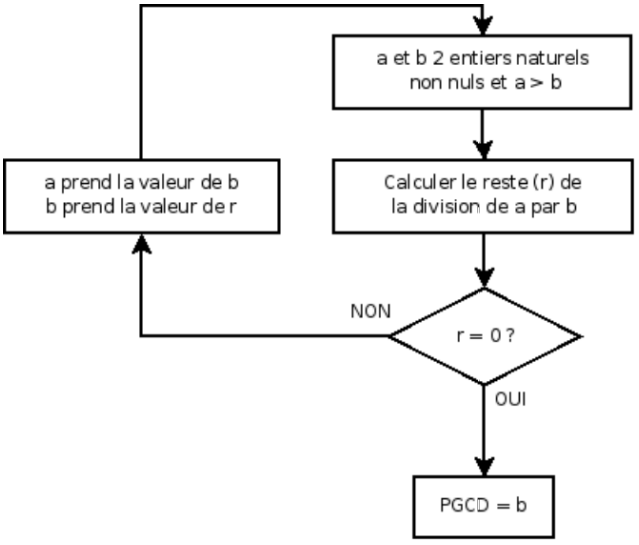
python

```
def Euclide_PGCD(a,b): # on définit le nom de la
                       # fonction et ses variables
                       #d'entrées/d'appel
    r=a%b              # on calcule le reste dans
                       # la division de a par b

    while r!=0:        # tant que ce reste est non nul :
        a=b            # b devient le nouveau a
        b=r            # r devient le nouveau b
        r=a%b          # on recalcule le reste

    return(b)          # une fois la boucle terminée,
                       # on retourne le dernier b
print(pgcd(1525,755))  # on affiche le résultat
                       # retourné par la fonction
```

Pseudo Code

Fonction PGCD : algorithme d'Euclide
Données : a et b : deux entiers naturels non nuls tels que a > b
Résultat : le PGCD de a et b

Euclide_PGCD(a,b)
  Répéter
    r $\leftarrow$ a mod b
    a $\leftarrow$ b
    b $\leftarrow$ r
  Jusqu'à r == 0
  Retourner a



6.3 Recherche des nombres premiers – Crible d'Ératosthène

6.4 Calcul de puissance

6.4.1 Algorithme naïf

6.4.2 Exponentiation rapide

7 Calcul d'un polynôme

7.1 Algorithme naïf

7.2 Méthode de Horner

Références

[1] Patrick Beynet, Cours d'informatique de CPGE, Lycée Rouvière de Toulon, UPSTI.
[2] Adrien Petri et Laurent Deschamps, Cours d'informatique de CPGE, Lycée Rouvière de Toulon, UPSTI.
[3] Damien Iceta, Cours d'informatique de CPGE, Lycée Gustave Eiffel de Cachan, UPSTI.