

CI 2 : ALGORITHMIQUE & PROGRAMMATION

ALGORITHMES D'INFORMATIQUE

1	Recherches dans une liste	1
1.1	Recherche d'un nombre dans une liste	1
1.2	Recherche du maximum dans une liste de nombre	2
1.3	Recherche par dichotomie dans un tableau trié	2
2	Gestion d'une liste de nombres	3
2.1	Calcul de la moyenne	3
2.2	Calcul de la variance	3
2.3	Calcul de la médiane	3
3	Chaînes de caractères	3
3.1	Recherche d'un mot dans une chaîne de caractères	3
4	Calcul numérique	4
4.1	Recherche du zéro d'une fonction continue monotone par la méthode de dichotomie	4
4.2	Recherche du zéro d'une fonction continue monotone par la méthode de Newton	4
4.3	Méthode des rectangles pour le calcul approché d'une intégrale sur un segment	5
4.4	Méthode des trapèzes pour le calcul approché d'une intégrale sur un segment	5
4.5	Méthode d'Euler pour la résolution d'une équation différentielle	5
5	Algorithmes de tris	5
5.1	Tri par insertion	5
5.2	Tri rapide «Quicksort»	5
5.3	Tri fusion	5
6	Algorithmes classiques	5
6.1	Division euclidienne	5
6.2	Algorithme d'Euclide	5
6.3	Recherche des nombres premiers – Crible d'Ératosthène	6
6.4	Calcul de puissance	6
7	Calcul d'un polynôme	6
7.1	Algorithme naïf	6
7.2	Méthode de Horner	6

1 Recherches dans une liste

1.1 Recherche d'un nombre dans une liste



```
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



```
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



```
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:
```



```

        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



```

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



```

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 :  $f, a, b, \varepsilon$ 
   $g \leftarrow a$ 
   $d \leftarrow b$ 
   $f_g \leftarrow f(g)$ 
   $f_d \leftarrow f(d)$ 
  tant que  $(d - g) > 2\varepsilon$  faire
     $m \leftarrow (g + d)/2$ 
     $f_m \leftarrow f(m)$ 
    si  $f_g \cdot f_m \leq 0$  alors
       $d \leftarrow m$ 
       $f_d \leftarrow f_m$ 
    sinon
       $g \leftarrow m$ 
       $f_g \leftarrow f_m$ 
    fin
  fin
  retourner  $(g + d)/2$ 
Fin
  
```

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 :  $f, f', a, \varepsilon$ 
   $g \leftarrow a$ 
   $c \leftarrow g - \frac{f(g)}{f'(g)}$ 
  tant que  $|c - g| > \varepsilon$  faire
     $g \leftarrow c$ 
     $c \leftarrow c - \frac{f(c)}{f'(c)}$ 
  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  $x$ .
```

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