Algorithmes à connaître

Les algorithmes présentés ici sont à connaître :

- ils font partis de la culture de base en informatique ;
- par de petites modifications, ils permettent de résoudre de nombreux problèmes.

Table des matières

1 Algorithmes classiques

1.1 Division euclidienne

```python
def quotient_reste(a,b):
    """
    Renvoie le quotient q et le reste r de la division de a par b
    Arguments:
        a, int : dividende, entier naturel
        b, int : diviseur, entier naturel
    Retour:
        q, int : quotient
        r, int : reste
    """
    r, q = a, 0
    while r >= b:
        r = r - b
        q = q + 1
    return(q,r)
```

1.2 Calcul de puissance

1.2.1 Algorithme naïf

```python
def exponentiation_naive(x,n):
    """
    Renvoie x**n par la methode naive.
    Arguments :
        x, flt : réel
        n, int : entier naturel
    Retour :
        res,flt : resultat
    """
    res = 1
    while n>=1:
        res = res * x
        n=n-1
    return(res)
```

1.2.2 Exponentiation rapide

```python
def exponentiation_rapide(x,n):
    """
    Renvoie x**n par la methode d'exponentiation rapide.
    Arguments:
        x, flt : un nombre réel
        n, int : un nombre entier naturel
    Retour :
        res,flt : resultat
    """
    res = 1
    a = x
```

```python
    while n>0:
        if n%2 == 1:
            res=res*a
        a=a*a
        n=n//2
    return(res)
```

1.3 Conversion décimale ⇆ binaire

1.3.1 Conversion décimale → binaire

```python
def Conversion_decimale_binaire(n):
    """
    Renvoie l'expression binaire d'un entier positif n
    Argument:
        n, int : entier naturel
    Retour :
        S,str : expression binaire
    """
    S=''
    while n>0:
        S=str(n%2)+S
        n=n//2
    return(S)
```

1.3.2 Conversion décimale ← binaire

```python
def Conversion_binaire_decimale(S):
    """
    Renvoie un entier positif correspondant àune expression binaire
    Keyword arguments:
    Argument :
        S,str : expression binaire
    Retour :
        n, int : un nombre entier naturel
    """
    N=0 #initialisation
    for i in range(len(S)):
        N=N+int(S[len(S)-1-i])*2**i
    return(N)
```

2 Recherches dans une liste

2.1 Recherche d'un nombre dans une liste

```python
def is_number_in_list(nb,L):
    """Renvoie True si le nombre entier nb est dans la liste de nombres L
    Arguments:
        nb, int : nombre entier
        L, list : liste de nombres entiers
    Retour:
        bool : True or False
    """
    for i in range(len(L)):
        if L[i]==nb:
            return(True)
    return(False)
```

2.2 Recherche du maximum dans une liste de nombre

```python
def what_is_max(L):
    """
    Renvoie le plus grand nombre d'une liste
    Arguments :
        L: liste de nombres
    Retour :
        maxi, flt ou int : maximum de la liste
    """
    maxi=L[0]
    for i in range(len(L)):
        if L[i]>maxi:
            maxi=L[i]
    return(maxi)
```

2.3 Recherche du maximum dans un tableau

Un tableau est codé en machine par une liste de listes de nombres.

```python
def what_is_max(T):
    """
    Renvoie le plus grand nombre d'un tableau T
    Arguments :
        T:  liste de listes de nombres
    Retour :
        maxi, flt ou int : maximum de la liste
    """
    maxi=L[0][0]
    for i in range(len(L)):
        for j in range(len(L[i])
            if L[i][j]>maxi:
                maxi=L[i]
    return(maxi)
```

2.4 Recherche par dichotomie dans une liste triée

```python
def is_number_in_list_dicho(nb,L):
    """
    Recherche d'un nombre par dichotomie dans une liste triée par ordre CROISSANT.
    Renvoie l'index si le nombre nb est dans la liste de nombres L.
    Renvoie None sinon.
    Arguments :
        nb, int ou flt : nombre
        L, list : liste de nombres entiers triés
    Retour :
        None
        ou
        m, int: index entier
    """
    g, d = 0, len(L)-1
    while g <= d:
        m = (g + d) // 2
        if L[m] == nb:
            return(m)
        if L[m] < nb:
            g = m+1
        else:
            d = m-1
    return(None)
```

3 Traitement d'une liste de nombres

3.1 Calcul de la moyenne

```python
def calcul_moyenne(L):
    """
    Renvoie la moyenne des valeurs d'une liste de
    nombres.
    Argument :
        L, flt : liste de nombres
    Retour :
        flt : moyenne
    """
    res = 0
    for i in range(len(L)):
        res = res+L[i]
    return(res/len(L))
```

3.2 Calcul de la variance

Soit une série statistique prenant les n valeurs x_1, x_2, ..., x_n. Soit m la moyenne de ces valeurs. La variance est définie par :

$$v = \frac{1}{n}\sum_{i=1}^{n}(x_i - m)^2$$

```python
def calcul_variance(L,m):
    """
    Renvoie la variance des valeurs d'un tableau.
    Argument :
        L, list : liste de nombres
    Retour :
        flt : la variance
    Nécessite la fonction calcul_moyenne
    """
    m=calcul_moyenne(L)
    res = 0
    for i in range(len(L)):
        res = res+(L[i]-m)**2
    return(res/len(L))
```

4 Algorithmes par boucles imbriquées

4.1 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.
    Arguments:
        mot,str : mot recherché
        texte, str : texte complet
    Retour:
        None ou i,int : index de la première lettre du mot dans le 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)
```

4.2 2 valeurs proches dans une liste

```python
def rech_2proches(L):
    """ Recherche les deux valeurs les plus proche dans une liste.
    Renvoie les index de ces deux valeur.
    Arguments:
        L,list : liste
    Retour:
        tuple : le tuple formé par les duex index
    """
    n = len(L)
    ind = [0 , 1]
    d = abs( L[0]-L[1])
```

```python
for i in range(n-1):
    for j in range (i+1,n):
        dij = abs(L[i]-L[j])
        if dij < d :
            d = dij
            ind = [ i , j ]
return(ind)
```