

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	2
1.1	Division euclidienne	2
1.2	Calcul de puissance	2
1.2.1	Algorithme naïf	2
1.2.2	Exponentiation rapide	2
1.3	Conversion décimale \leftrightarrow binaire	3
1.3.1	Conversion décimale \rightarrow binaire	3
1.3.2	Conversion décimale \leftarrow binaire	3
2	Recherches dans une liste	4
2.1	Recherche d'un nombre dans une liste	4
2.2	Recherche du maximum dans une liste de nombre	4
2.3	Recherche du maximum dans un tableau	4
2.4	Recherche par dichotomie dans une liste triée	5
3	Traitement d'une liste de nombres	5
3.1	Calcul de la moyenne	5
3.2	Calcul de la variance	5
4	Algorithmes par boucles imbriquées	6
4.1	2 valeurs proches dans une liste	6
4.2	Recherche d'un mot dans une chaîne de caractères	6

1 Algorithmes classiques

1.1 Division euclidienne

```
1 def quotient_reste(a,b):
2     """
3     Renvoie le quotient q et le reste r de la division de a par b
4     Arguments:
5         a, int : dividende, entier naturel
6         b, int : diviseur, entier naturel
7     Retour:
8         q, int : quotient
9         r, int : reste
10    """
11    r, q = a, 0
12    while r >= b:
13        r = r - b
14        q = q + 1
15    return(q,r)
```

1.2 Calcul de puissance

1.2.1 Algorithme naïf

```
1 def exponentiation_naive(x,n):
2     """
3     Renvoie x**n par la methode naive.
4     Arguments :
5         x,flt : réel
6         n, int : entier naturel
7     Retour :
8         res,flt : resultat
9     """
10    res = 1
11    while n>=1:
12        res = res * x
13        n=n-1
14    return(res)
```

1.2.2 Exponentiation rapide

```
1 def exponentiation_rapide(x,n):
2     """
3     Renvoie x**n par la methode d'exponentiation rapide.
4     Arguments:
5         x,flt : un nombre réel
6         n, int : un nombre entier naturel
7     Retour :
8         res,flt : resultat
9     """
10    res = 1
11    a = x
```

```

12     while n>0:
13         if n%2 == 1:
14             res=res*a
15         a=a*a
16         n=n//2
17     return(res)

```

1.3 Conversion décimale \leftrightarrow binaire

1.3.1 Conversion décimale \rightarrow binaire

```

1  def Conversion_decimale_binaire(n):
2      """
3      Renvoie l'expression binaire d'un entier positif n
4      Argument:
5          n, int : entier naturel
6      Retour :
7          S,str : expression binaire
8      """
9      S=''
10     while n>0:
11         S=str(n%2)+S
12         n=n//2
13     return(S)

```

1.3.2 Conversion décimale \leftarrow binaire

```

1  def Conversion_binaire_decimale(S):
2      """
3      Renvoie un entier positif correspondant à une expression binaire
4      Keyword arguments:
5      Argument :
6          S,str : expression binaire
7      Retour :
8          n, int : un nombre entier naturel
9      """
10     N=0 #initialisation
11     for i in range(len(S)):
12         N=N+int(S[len(S)-1-i])*2**i
13     return(N)

```

2 Recherches dans une liste

2.1 Recherche d'un nombre dans une liste

```

1 def is_number_in_list(nb,L):
2     """Renvoie True si le nombre entier nb est dans la liste de nombres L
3     Arguments:
4         nb, int : nombre entier
5         L, list : liste de nombres entiers
6     Retour:
7         bool : True or False
8     """
9     for i in range(len(L)):
10        if L[i]==nb:
11            return(True)
12        return(False)

```

2.2 Recherche du maximum dans une liste de nombre

```

1 def what_is_max(L):
2     """
3     Renvoie le plus grand nombre d'une liste
4     Arguments :
5         L: liste de nombres
6     Retour :
7         maxi, flt ou int : maximum de la liste
8     """
9     maxi=-float('inf')
10    for i in range(len(L)):
11        if L[i]>maxi:
12            maxi=L[i]
13    return(maxi)

```

2.3 Recherche du maximum dans un tableau

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

```

1 def what_is_max(T):
2     """
3     Renvoie le plus grand nombre d'un tableau T
4     Arguments :
5         T: liste de listes de nombres
6     Retour :
7         maxi, flt ou int : maximum de la liste
8     """
9     maxi=-float('inf')
10    for i in range(len(T)):
11        for j in range(len(T[i]))
12            if T[i][j]>maxi:
13                maxi=T[i]
14    return(maxi)

```

2.4 Recherche par dichotomie dans une liste triée

```

1  def is_number_in_list_dicho(nb,L):
2      """
3      Recherche d'un nombre par dichotomie dans une liste triée par ordre CROISSANT.
4      Renvoie l'index si le nombre nb est dans la liste de nombres L.
5      Renvoie None sinon.
6      Arguments :
7          nb, int ou flt : nombre
8          L, list : liste de nombres entiers triés
9      Retour :
10         None
11         ou
12         m, int: index entier
13     """
14     g, d = 0, len(L)-1
15     while g <= d:
16         m = (g + d) // 2
17         if L[m] == nb:
18             return(m)
19         if L[m] < nb:
20             g = m+1
21         else:
22             d = m-1
23     return(None)

```

3 Traitement d'une liste de nombres

3.1 Calcul de la moyenne

```

1  def calcul_moyenne(L):
2      """
3      Renvoie la moyenne des valeurs d'une liste de
4      nombres.
5      Argument :
6          L, flt : liste de nombres
7      Retour :
8          flt : moyenne
9      """
10     res = 0
11     for i in range(len(L)):
12         res = res+L[i]
13     return(res/len(L))

```

3.2 Calcul de la variance

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

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

```

1 def calcul_variance(L,m):
2     """
3     Renvoie la variance des valeurs d'un tableau.
4     Argument :
5         L, list : liste de nombres
6     Retour :
7         flt : la variance
8     Nécessite la fonction calcul_moyenne
9     """
10    m=calcul_moyenne(L)
11    res = 0
12    for i in range(len(L)):
13        res = res+(L[i]-m)**2
14    return(res/len(L))

```

4 Algorithmes par boucles imbriquées

4.1 2 valeurs proches dans une liste

```

1 def rech_2proches(L):
2     """ Recherche les deux valeurs les plus proche dans une liste.
3     Renvoie les index de ces deux valeur.
4     Arguments:
5         L,list : liste
6     Retour:
7         tuple : le tuple formé par les deux index
8     """
9     n = len(L)
10    ind = [0 , 1]
11    d = abs( L[0]-L[1])
12    for i in range(n-1):
13        for j in range (i+1,n):
14            dij = abs(L[i]-L[j])
15            if dij < d :
16                d = dij
17                ind = [ i , j ]
18    return(ind)


```

4.2 Recherche d'un mot dans une chaîne de caractères

```

1 def index_of_word_in_text(mot, texte):
2     """ Recherche si le mot est dans le texte.
3     Renvoie l'index si le mot est présent, None sinon.
4     Arguments:
5         mot,str : mot recherché
6         texte, str : texte complet
7     Retour:
8         None ou i,int : index de la première lettre du mot dans le texte
9     """

```



```
10 for i in range(1 + len(texte) - len(mot)):  
11     j = 0  
12     while j < len(mot) and mot[j] == texte[i + j]:  
13         j += 1  
14     if j == len(mot):  
15         return(i)  
16 return(None)
```