# Corrigé du QCM n°5

1. On utilise un algorithme de recherche dichotomique :

   1. sur une liste triée
   2. pour rechercher un élément dans une liste
   3. pour rechercher un mot dans un texte
   4. sur une liste non triée

L'algorithme de *recherche dichotomique* est un algorithme de recherche d'un élément dans une liste triée.

2. Lors d'une recherche dichotomique :

   1. la taille de la plage de recherche est au moins divisée par deux à chaque itération
   2. une variable parcourt les positions du début à la fin
   3. on utilise une boucle `for`
   4. deux variables délimitent la plage de recherche

On code la recherche dichotomique par :

```
def rech_dicho(elt,L):
    """Recherche dichotomique de elt dans L liste triée"""
    deb=0
    fin=len(L)-1
    trouve=False
    while not trouve and deb<=fin:
        milieu=(deb+fin)//2
        if L[milieu]==elt:
            trouve=True
        elif L[milieu]>elt:
            fin=milieu-1
        else:
            deb=milieu+1
    return L[milieu]==elt,milieu
```

Les deux variables `deb` et `fin` délimitent la plage de recherche dont la taille est au moins divisée par deux à chaque itération puisqu'on retient la moitié droite stricte ou gauche stricte vis-à-vis du milieu lors de l'itération.

3. L'algorithme de Horner :

   1. sert à évaluer un polynôme en un point
   2. est implémenté avec un parcours des indices d'une liste dans l'ordre décroissant
   3. effectue une recherche dichotomique
   4. est implémenté avec un parcours des indices d'une liste dans l'ordre croissant

Soit $x$ réel et $\mathrm{P} = \sum_{k=0}^{n-1} a_k \mathrm{X}^k \in \mathbb{R}[\mathrm{X}]$. L'algorithme de Horner consiste à calculer efficacement $\mathrm{P}(x)$ en observant

$$\mathrm{P}(x) = \sum_{k=0}^{n-1} a_k x^k = (\ldots((a_{n-1}x + a_{n-2})\,x + a_{n-3})\,x + \ldots)\,x + a_0$$

d'où l'implémentation :

```
def poly(x,P):
    """Calcul de P(x) suivant l'algorithme de Horner
    x : flottant
    P : [a_0, ..., a_{n-1}] liste de flottants"""
    res=0
    n=len(P)
    for k in range(n-1,-1,-1):
        res=x*res+P[k]
    return res
```

Les indices de la liste sont parcourus par ordre décroissant.

4. Une application usuelle de l'algorithme de Horner est :

1. la détermination de la valeur d'un entier à partir de son écriture binaire
2. le calcul d'un coefficient binomial
3. la recherche d'un minimum
4. l'exponentiation rapide

Soit $n$ entier non nul d'écriture binaire

$$n = \langle d_{p-1}, d_{p-2}, \ldots, d_0 \rangle = \sum_{i=0}^{p-1} d_i 2^i \quad \text{avec} \quad (d_0, \ldots, d_{p-1}) \in \{0,1\}^{p-1} \times \{1\}$$

Posant $\mathrm{P} = \sum_{i=0}^{p-1} d_i \mathrm{X}^i$, on a $n = \mathrm{P}(2)$ ce qui est donc un cadre typique d'utilisation de l'algorithme de Horner.

5. La taille de l'écriture binaire d'un entier $n$ non nul est :

1. $\log_2(n)$
2. $\lfloor \log_{10}(n) + 1 \rfloor$
3. $\log_{10}(n)$
4. $\lfloor \log_2(n) + 1 \rfloor$

Si $p$ est le nombre de chiffres de l'écriture binaire de $n$, cela signifie que

$$n = 2^{p-1} + \sum_{i=0}^{p-2} d_i 2^i \quad \text{avec} \quad (d_0, \ldots, d_{p-2}) \in \{0,1\}^{p-1}$$

d'où

$$2^{p-1} \leqslant n \leqslant \sum_{i=0}^{p-1} 2^i = \frac{2^p - 1}{2 - 1} = 2^p - 1 < 2^p$$

Passant au logarithme, fonction strictement croissante, on obtient

$$(p-1)\log 2 \leqslant \log n < p \log 2 \quad \Longrightarrow \quad p \leqslant \log_2(n) + 1 < p + 1$$

On en déduit $p = \lfloor \log_2(n) + 1 \rfloor$.

6. En méthode de complément à 2 :
   1. le signe est codé comme bit de puissance de $-1$
   2. le signe est codé sur le bit de poids fort
   3. le signe est codé sur le bit de poids faible
   4. le signe est codé par soustraction d'une puissance de 2

En *méthode de complément à* 2, le codage d'entiers relatifs est défini par l'écriture binaire

$$\langle d_{p-1}, d_{p_2}, \ldots, d_0 \rangle_{\texttt{CPL2}} = -d_{p-1}2^{p-1} + \sum_{i=0}^{p-2} d_i 2^i$$

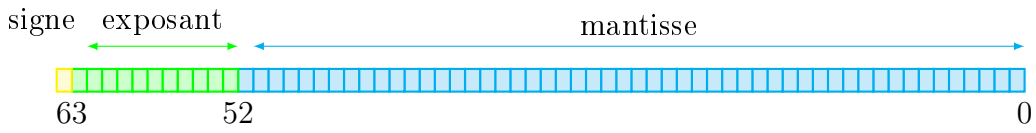
Le signe est donc codé sur le bit de poids fort par soustraction d'une puissance de 2.

7. Le format flottant est codé sur :
   1. 32 bits
   2. 128 bits
   3. 16 bits
   4. 64 bits

D'après la norme IEEE754, Le format est flottant est codé sur 64 bits.

8. Le format flottant se décompose en :
   1. cinq parties
   2. trois parties
   3. deux parties
   4. quatre parties

Le format flottant se décompose en trois parties :
— le bit de signe ;
— l'exposant sur 11 bits ;
— la mantisse sur 52 bits.




9. Le format flottant permet de coder fidèlement :
   1. certains nombres rationnels
   2. les nombres décimaux
   3. les nombres rationnels
   4. les nombres réels

Le format flottant permet de coder fidèlement certains nombres rationnels, ceux dont la mantisse s'écrit comme combinaison de puissances négatives de deux. Les nombres décimaux comme 1/10 dont l'écriture fractionnaire binaire est infinie sont tronqués en machine et donc codés imparfaitement. *A fortiori*, les nombres rationnels et réels ne sont pas codés fidèlement en général.

10. Soient `a` et `b` deux nombres flottants. Pour tester l'égalité entre `a` et `b`, on utilise :

1. `abs(a-b)==0`
2. `abs(a-b)==eps` avec `eps` un seuil choisi par l'utilisateur
3. `abs(a-b)<eps` avec `eps` un seuil choisi par l'utilisateur
4. `a==b`

Compte-tenu des limitations du format flottant, un test d'égalité sur des flottants est à écrire systématiquement avec une tolérance choisi par l'utilisateur.