Applications

Exercices d'application

0.1 Recherche de maximum d'un tableau

Exercice 1 –

Question 1 *Étudier la complexité théorique de la fonction* `maxi`

```
def maxi(t):
    """Renvoie le plus grand élément de t.
       Précondition : t est un tableau non vide"""
    m = t[0]
    for x in t:
        # Invariant : m est le plus grand élément trouvé jusqu'ici
        if x > m:
            m = x # On a trouvé plus grand, on met àjour m
    return m
```

0.2 Appartenance d'un élément dans un tableau

Exercice 2 – **Question 1** *Étudier les complexité théoriques (dans le pire des cas) des fonctions* `appartient` *et* `appartient_dicho`. *Les comparer.*

```
def appartient(e, t):
    """Renvoie un booléen disant si e appartient àt
       Précondition : t est un tableau"""
    for x in t:
        # Invariant : e n'est pas positionné dans t avant x
        if e == x:
            return True # On a trouvé e, on s'arrête
    return False
```

```
def appartient_dicho(e,t):
    """Renvoie un booléen indiquant si e est dans t
       Préconditions : t est un tableau de nombres trié par ordre croissant
                       e est un nombre"""
    g = 0 # Limite gauche de la tranche où l'on recherche e
    d = len(t)-1 # Limite droite de la tranche où l'on recherche e
    while g <= d: # La tranche où l'on cherche e n'est pas vide
        m = (g+d)//2 # Milieu de la tranche où l'on recherche e
        pivot = t[m]
        if e == pivot: # On a trouvé e
            return True
        elif e < pivot:
            d = m-1 # On recherche e dans la partie gauche de la tranche
        else:
```

```
            g = m+1 # On recherche e dans la partie droite de la tranche
    return False
```

0.3 Recherche d'un mot dans une chaine

Exercice 3 –

Question 1 *Étudier la complexité théorique dans le pire des cas de la fonction* `recherche`. *On pourra être amené à la reformuler légèrement.*

```
def recherche(m,s):
    """Recherche le mot m dans la chaÃ@ne s
       Préconditions : m et s sont des chaÃ@nes de caractères"""
    long_s = len(s) # Longueur de s
    long_m = len(m) # Longueur de m
    for i in range(long_s-long_m+1):
        # Invariant : m n'a pas été trouvé dans s[0:i+long_m-1]
        if s[i:i+long_m] == m: # On a trouvé m
            return True
    return False
```

0.4 Conversion d'entier en binaire

Exercice 4 –

Question 1 *Étudier la complexité théorique de la fonction* `conv_b2`

```
def conv_b2(p):
 """Convertit l'entier p en base 2 (renvoie une chaÃ@ne)"""
    x = p
    s = ""
    while x > 1 :
        s = str(x%2) + s
        x = x // 2
    return str(x)+s
```

Question 2 *Étudier les complexités théoriques des fonctions* `calc_b2_naif` *et* `calc_b2_horner`. *Les comparer.*

```
def calc_b2_naif(s):
    """Renvoie l'entier p représente en binaire par s"""
    p = 0
    x = 1 ## 2**0
    for i in range(len(s)):
        p = p+int(s[len(s)-i-1])*x
        x = 2*x
    return p
```

```
def calc_b2_horner(s):
    """Renvoie l'entier p représente en binaire par s"""
    p = int(s[0])
    for i in range(1,len(s)):
        p = int(s[i])+2*p
    return p
```