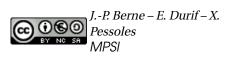
# Thèmes d'étude

	_					
1	l Parco	urs d	'une li	iste d	e list	es

2



<sup>2</sup> Recherche de facteur dans un mot

Thème: Algorithmes opérant sur une structure séquentielle par boucles imbriquées. Commentaires:

- recherche d'un facteur dans un texte;
- recherche des deux valeurs les plus proches dans un tableau;
- tri à bulles;
- notion de complexité quadratique
- outils pour valider la correction de l'algorithme

# 1 Parcours d'une liste de listes

Les listes de listes permettent de mettre les données en deux dimensions.

## ■ Exemple

Grille de mots mêlés.

	Е	Т	Е	
	S	Е	С	
'				
grille =	[[1]	L','	Ε','	S'],['E',
'T','	$\mathbf{E}^{+}$	, ['S	','E	','C']]

L E S

Table de multiplication

×	1	2	3
1	1	2	3
2	2	4	6
3	3	6	9

Température en fonction du temps.

T(s)	1	2	3	4
T°C	18	19	21	24

```
data = [[1,18],
[2,19],[3,21],[4,24]]
```

Pour parcourir les éléments d'un tableau on procède de la même façon que pour une recherche séquentielle. Prenons l'exemple d'un tableau tab de n lignes et p colonnes.

# Utilisation de boucles while

```
n = len(tab)
p = len(tab[0])
i, j = 0,0
while i<n :
    while j<p :
        print(tab[i][j])
        j = j+1
    j=0
    i = i+1</pre>
```

#### Utilisation de boucles for

```
n = len(tab)
p = len(tab[0])
for i in range(n) :
    for j in range(p) :
        print(tab[i][j])
```

#### Écriture de boucles for en Python

```
for t in tab :
   for e in t :
     print(e)
```



Il est possible de dénombrer le nombre d'itérations réalisées par les algorithmes ci-dessus. Dans chaque cas, la première boucle est réalisée n fois. La seconde boucle, imbriquée dans la première est parcourue p fois. On peut donc dénombrer le nombre de fois que la fonction print est appelée :  $n \times p$ .

Une estimation grossière du nombre d'opérations réalisées en tout est donc  $n \times p$ . On dit que la complexité dans ces algorithmes, dans le pire des cas est  $\mathcal{O}(np)$ . Si n=p, la complexité est de  $\mathcal{O}(n^2)$ . On parle de complexité quadratique.



## 2 Recherche de facteur dans un mot

Rechercher un facteur dans un mot signifie rechercher une (sous-)chaîne de caractères dans une chaîne de caractères (ou encore un mot dans une chaîne).

```
def recherche(m,s):
   """Recherche le mot m dans la chaine s
      Préconditions : m et s sont des chaines 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]
       j = 0
       while j < long_m and m[j] == s[i+j]:
          # Invariant : m[:j] == s[i:i+j]
          j = j+1
          # Invariant : m[:j] == s[i:i+j]
       if j == long_m:
          # Invariant précédent : m == s[i:i+long_m]
          return True
   return False
```

Cet algorithme est simplifiable en utilisant le slicing.

```
def recherche(m,s):
    """Recherche le mot m dans la chaine s
    Préconditions : m et s sont des chaines 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
```

En utilisant les possibilités de Python, il est possible de simplifier encore l'algorithme.

```
def recherche(m,s):
    """Recherche le mot m dans la chaine s
    Préconditions : m et s sont des chaines de caractères"""
    return m in s
```