

## Initiation à l'algorithmique

— structures linéaires —

**Jacques TISSEAU**

ÉCOLE NATIONALE D'INGÉNIEURS DE BREST  
Technopôle Brest-Iroise  
CS 73862 - 29238 Brest cedex 3 - France

enib©2009

**Séquence** : suite ordonnée d'éléments,

**Séquence** : suite ordonnée d'éléments, éventuellement vide,

**Séquence** : suite ordonnée d'éléments, éventuellement vide, accessibles par leur rang dans la séquence

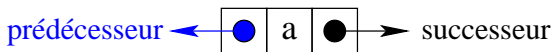
**Séquence** : suite ordonnée d'éléments, éventuellement vide, accessibles par leur rang dans la séquence

Exemple de séquence : main au poker



**Séquence** : suite ordonnée d'éléments, éventuellement vide, accessibles par leur rang dans la séquence

Exemple de séquence : main au poker



**n-uplet**

## n-uplet

```
>>> s = 1,7,2,4
```

## n-uplet

```
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## liste

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## liste

```
>>> s = [1,7,2,4]
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## liste

```
>>> s = [1,7,2,4]
>>> type(s)
<type 'list'>
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## liste

```
>>> s = [1,7,2,4]
>>> type(s)
<type 'list'>
>>> len(s)
4
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## liste

```
>>> s = [1,7,2,4]
>>> type(s)
<type 'list'>
>>> len(s)
4
>>> 3 in s
False
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## liste

```
>>> s = [1,7,2,4]
>>> type(s)
<type 'list'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
```

## n-uplet

```
>>> s = 1,7,2,4
>>> type(s)
<type 'tuple'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + (5,3)
(1, 7, 2, 4, 5, 3)
```

## chaîne

```
>>> s = '1724'
>>> type(s)
<type 'str'>
>>> len(s)
4
>>> '3' in s
False
>>> s[1]
'7'
>>> s + '53'
'172453'
```

## liste

```
>>> s = [1,7,2,4]
>>> type(s)
<type 'list'>
>>> len(s)
4
>>> 3 in s
False
>>> s[1]
7
>>> s + [5,3]
[1, 7, 2, 4, 5, 3]
```

**n-uplet** séquence non modifiable d'éléments.

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)      paire      : (a,b)

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)      paire : (a,b)

triplet : (a,b,c)      quadruplet : (a,b,c,d)

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire            : (a,b)

triplet     : (a,b,c)    quadruplet : (a,b,c,d)

...         :

n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)      paire      : (a,b)

triplet    : (a,b,c)    quadruplet : (a,b,c,d)

...        :

n-uplet   : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)      paire      : (a,b)

triplet    : (a,b,c)    quadruplet : (a,b,c,d)

...        :

n-uplet   : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)      paire      : (a,b)

triplet    : (a,b,c)    quadruplet : (a,b,c,d)

...        :

n-uplet   : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()
>>> type(s)
<type 'tuple'>
>>> s = 1,7,2,4
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)      paire      : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)      paire      : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire        : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire        : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>  
>>> s = (5,)  
>>> type(s)  
<type 'tuple'>
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire        : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>  
>>> s = (5,)  
>>> type(s)  
<type 'tuple'>  
>>> s =  
(5,)+(6,)+(7,9)  
>>> s  
(5, 6, 7, 9)
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire            : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>  
>>> s = (5,)  
>>> type(s)  
<type 'tuple'>  
>>> s =  
(5,)+(6,7,9)  
>>> s  
(5, 6, 7, 9)
```

```
>>> s = (5,6,7,9)
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire        : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>  
>>> s = (5,)  
>>> type(s)  
<type 'tuple'>  
>>> s =  
(5,)+(6,7,9)  
>>> s  
(5, 6, 7, 9)
```

```
>>> s = (5,6,7,9)  
>>> s[1:3]  
(6, 7)
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire        : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>  
>>> s = (5,)  
>>> type(s)  
<type 'tuple'>  
>>> s =  
(5,)+(6,7,9)  
>>> s  
(5, 6, 7, 9)
```

```
>>> s = (5,6,7,9)  
>>> s[1:3]  
(6, 7)  
>>> s[1:]  
(6, 7, 9)
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire            : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>  
>>> s = (5,)  
>>> type(s)  
<type 'tuple'>  
>>> s =  
(5,)+(,)+(6,7,9)  
>>> s  
(5, 6, 7, 9)
```

```
>>> s = (5,6,7,9)  
>>> s[1:3]  
(6, 7)  
>>> s[1:]  
(6, 7, 9)  
>>> s[:2]  
(5, 6)
```

**n-uplet** séquence non modifiable d'éléments.

singleton : (a)            paire        : (a,b)  
triplet    : (a,b,c)    quadruplet : (a,b,c,d)  
...        :  
n-uplet    : (a,b,c,d,e,f,g,h,i,j,...)

```
>>> s = ()  
>>> type(s)  
<type 'tuple'>  
>>> s = 1,7,2,4  
>>> type(s)  
<type 'tuple'>  
>>> s  
(1, 7, 2, 4)
```

```
>>> s = (5)  
>>> type(s)  
<type 'int'>  
>>> s = (5,)  
>>> type(s)  
<type 'tuple'>  
>>> s =  
(5,)+(,)+(6,7,9)  
>>> s  
(5, 6, 7, 9)
```

```
>>> s = (5,6,7,9)  
>>> s[1:3]  
(6, 7)  
>>> s[1:]  
(6, 7, 9)  
>>> s[:2]  
(5, 6)  
>>> s[-2:]  
(7, 9)
```

**chaîne** séquence non modifiable de caractères.

**chaîne** séquence non modifiable de caractères.

```
>>> s = 'une chaîne'
```

**chaîne** séquence non modifiable de caractères.

```
>>> s = 'une chaîne'  
>>> s  
'une chaîne'
```

**chaîne** séquence non modifiable de caractères.

```
>>> s = 'une chaîne'  
>>> s  
'une chaîne'  
>>> s = "une autre chaîne"  
>>> s  
'une autre chaîne'
```

**chaîne** séquence non modifiable de caractères.

```
>>> s = 'une chaîne'  
>>> s  
'une chaîne'  
>>> s = "une autre chaîne"  
>>> s  
'une autre chaîne'  
>>> s = 'chaîne entrée sur \  
... plusieurs lignes'  
>>> s  
'chaîne entrée sur plusieurs lignes'
```

**chaîne** séquence non modifiable de caractères.

```
>>> s = 'une chaîne'
>>> s
'une chaîne'
>>> s = "une autre chaîne"
>>> s
'une autre chaîne'
>>> s = 'chaîne entrée sur \
... plusieurs lignes'
>>> s
'chaîne entrée sur plusieurs lignes'
>>> s = 'chaîne entrée \n sur 1
ligne'
>>> s
chaîne entrée
sur 1 ligne
```

**chaîne** séquence non modifiable de caractères.

```
>>> s = 'une chaîne'
>>> s
'une chaîne'
>>> s = "une autre chaîne"
>>> s
'une autre chaîne'
>>> s = 'chaîne entrée sur \
... plusieurs lignes'
>>> s
'chaîne entrée sur plusieurs lignes'
>>> s = 'chaîne entrée \n sur 1
ligne'
>>> s
chaîne entrée
sur 1 ligne
```

```
>>> s = 'c\'est ça \"peuchère\"'
>>> s
'c\'est ça "peuchère"'
>>> print(s)
c'est ça "peuchère"
```

**chaîne** séquence non modifiable de caractères.

```
>>> s = 'une chaîne'
>>> s
'une chaîne'
>>> s = "une autre chaîne"
>>> s
'une autre chaîne'
>>> s = 'chaîne entrée sur \
... plusieurs lignes'
>>> s
'chaîne entrée sur plusieurs lignes'
>>> s = 'chaîne entrée \n sur 1
ligne'
>>> s
chaîne entrée
sur 1 ligne
```

```
>>> s = 'c\'est ça \"peuchère\"'
>>> s
'c\'est ça "peuchère"'
>>> print(s)
c'est ça "peuchère"
>>> s = ''' a ' \ " \n z '''
>>> s
' a \' \\ " \n z '
>>> s = 'des caractères'
>>> s[9]
't'
>>> for c in s : print(c,end=' ')
...
d e s   c a r a c t è r e s
>>> s[4:9]
'carac'
>>> s[len(s)-1]
's'
>>> s[:4] + 'mo' + s[9] + s[-1]
'des mots'
```

**liste** séquence modifiable d'éléments.

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
>>> for c in s : print(c,end=' ')
...
1 3 5 7
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
>>> for c in s : print(c,end=' ')
...
1 3 5 7
>>> s[1:3]
[3,5]
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
>>> for c in s : print(c,end=' ')
...
1 3 5 7
>>> s[1:3]
[3,5]
>>> s[1:3] = [2,4]
>>> s
[1, 2, 4, 7]
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
>>> for c in s : print(c,end=' ')
...
1 3 5 7
>>> s[1:3]
[3,5]
>>> s[1:3] = [2,4]
>>> s
[1, 2, 4, 7]
>>> s[len(s):len(s)] = [8,9]
>>> s
[1, 2, 4, 7, 8, 9]
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
>>> for c in s : print(c,end=' ')
...
1 3 5 7
>>> s[1:3]
[3,5]
>>> s[1:3] = [2,4]
>>> s
[1, 2, 4, 7]
>>> s[len(s):len(s)] = [8,9]
>>> s
[1, 2, 4, 7, 8, 9]
```

```
>>> s = [1,2,3]
>>> t = s
>>> t[0] = 9
>>> t
[9, 2, 3]
>>> s
[9, 2, 3]
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
>>> for c in s : print(c,end=' ')
...
1 3 5 7
>>> s[1:3]
[3,5]
>>> s[1:3] = [2,4]
>>> s
[1, 2, 4, 7]
>>> s[len(s):len(s)] = [8,9]
>>> s
[1, 2, 4, 7, 8, 9]
```

```
>>> s = [1,2,3]
>>> t = s
>>> t[0] = 9
>>> t
[9, 2, 3]
>>> s
[9, 2, 3]
>>> s = [1,2,3]
>>> t = []
>>> t[:0] = s[0:]
>>> t
[1, 2, 3]
```

**liste** séquence modifiable d'éléments.

```
>>> s = [1,3,5,7]
>>> s[2]
5
>>> s[len(s)-1]
7
>>> for c in s : print(c,end=' ')
...
1 3 5 7
>>> s[1:3]
[3,5]
>>> s[1:3] = [2,4]
>>> s
[1, 2, 4, 7]
>>> s[len(s):len(s)] = [8,9]
>>> s
[1, 2, 4, 7, 8, 9]
```

```
>>> s = [1,2,3]
>>> t = s
>>> t[0] = 9
>>> t
[9, 2, 3]
>>> s
[9, 2, 3]
>>> s = [1,2,3]
>>> t = []
>>> t[:0] = s[0:]
>>> t
[1, 2, 3]
>>> t[0] = 9
>>> t
[9, 2, 3]
>>> s
[1, 2, 3]
```

## Piles



## Piles



- tester si la pile est vide,
- accéder au sommet de la pile,
- empiler un élément au sommet de la pile,
- dépiler l'élément qui se trouve au sommet de la pile.

## Piles



- tester si la pile est vide,
- accéder au sommet de la pile,
- empiler un élément au sommet de la pile,
- dépiler l'élément qui se trouve au sommet de la pile.

LIFO : Last In, First Out

## Piles



- tester si la pile est vide,
- accéder au sommet de la pile,
- empiler un élément au sommet de la pile,
- dépiler l'élément qui se trouve au sommet de la pile.

LIFO : Last In, First Out

## Files



FIFO : First In, First Out

**liste multidimensionnelle** liste dont les éléments sont des listes.

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5], [1,2,3], [6,7,8,9]]
```

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]  
>>> type(s)  
<type 'list'>
```

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]  
>>> type(s)  
<type 'list'>  
>>> len(s)  
3
```

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> type(s)
<type 'list'>
>>> len(s)
3
>>> type(s[2])
<type 'list'>
```

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> type(s)
<type 'list'>
>>> len(s)
3
>>> type(s[2])
<type 'list'>
>>> s[2]
[6, 7, 8, 9]
```

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> type(s)
<type 'list'>
>>> len(s)
3
>>> type(s[2])
<type 'list'>
>>> s[2]
[6, 7, 8, 9]
>>> s[2][1]
7
>>> s[1][2]
3
```

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> type(s)
<type 'list'>
>>> len(s)
3
>>> type(s[2])
<type 'list'>
>>> s[2]
[6, 7, 8, 9]
>>> s[2][1]
7
>>> s[1][2]
3
```

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> for c in s : print(c)
...
[4, 5]
[1, 2, 3]
[6, 7, 8, 9]
```

**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> type(s)
<type 'list'>
>>> len(s)
3
>>> type(s[2])
<type 'list'>
>>> s[2]
[6, 7, 8, 9]
>>> s[2][1]
7
>>> s[1][2]
3
```

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> for c in s : print(c)
...
[4, 5]
[1, 2, 3]
[6, 7, 8, 9]
>>> s[2]
[6, 7, 8, 9]
```

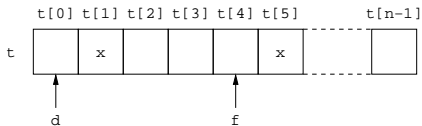
**liste multidimensionnelle** liste dont les éléments sont des listes.

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> type(s)
<type 'list'>
>>> len(s)
3
>>> type(s[2])
<type 'list'>
>>> s[2]
[6, 7, 8, 9]
>>> s[2][1]
7
>>> s[1][2]
3
```

```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> for c in s : print(c)
...
[4, 5]
[1, 2, 3]
[6, 7, 8, 9]
>>> s[2]
[6, 7, 8, 9]
>>> for c in s :
...     for e in c : print(e,end=' ')
...     print()
...
4 5
1 2 3
6 7 8 9
```

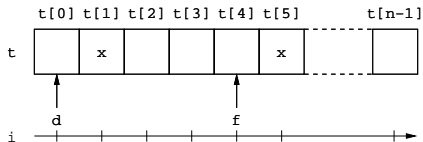
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



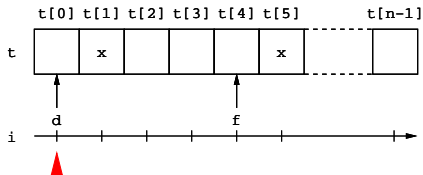
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



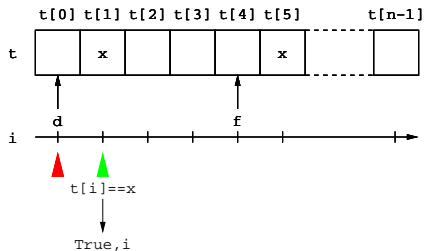
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



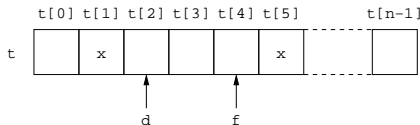
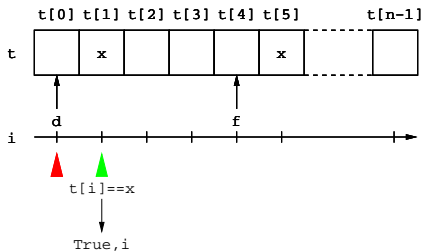
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



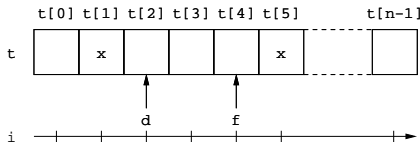
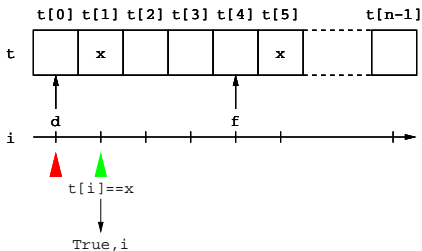
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



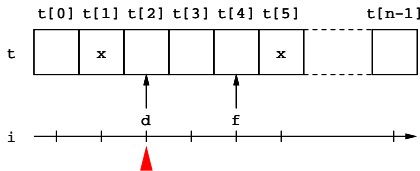
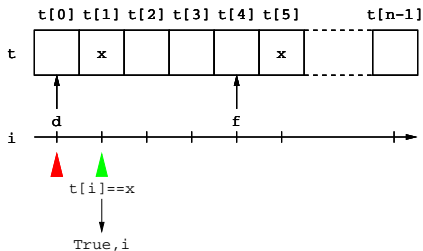
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



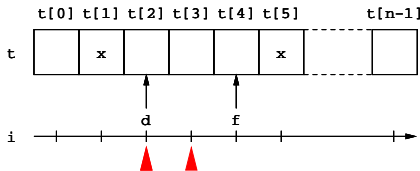
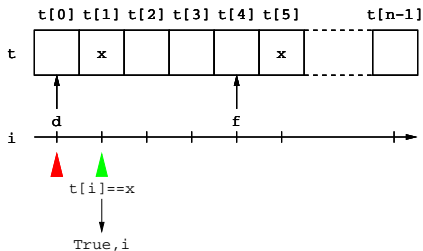
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



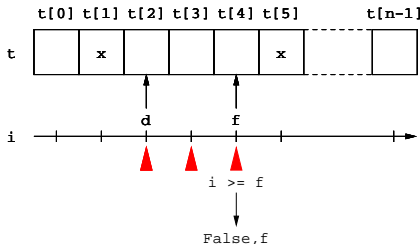
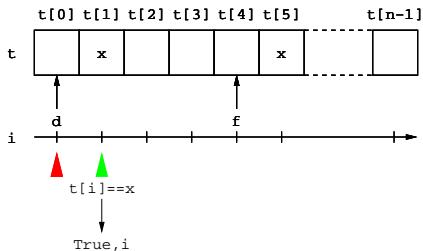
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



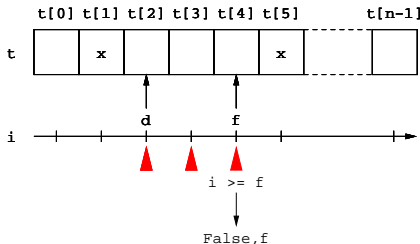
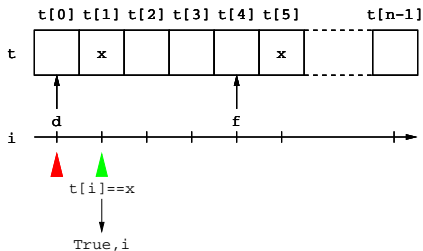
**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence



**recherche** retrouver une information stockée en mémoire vive, sur un disque dur ou sur le réseau.

## Recherche dans une séquence

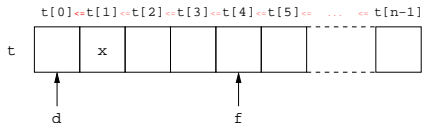


Complexité linéaire ( $O(n)$ )

## Recherche dans une séquence triée

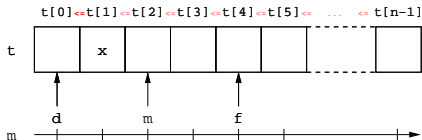
## Recherche dans une séquence triée

$m = (d+f)/2$  : milieu de la plage de recherche



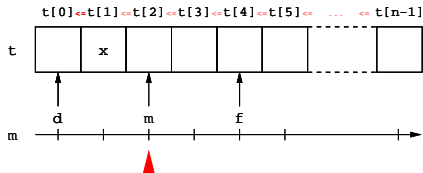
## Recherche dans une séquence triée

$m = (d+f)/2$  : milieu de la plage de recherche



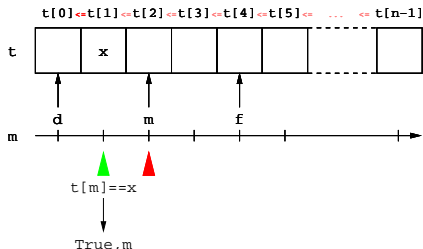
## Recherche dans une séquence triée

$m = (d+f)/2$  : milieu de la plage de recherche



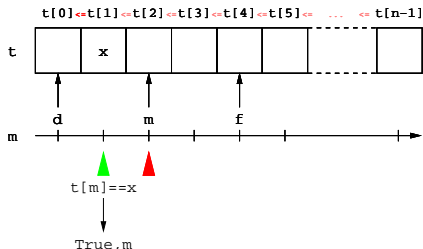
## Recherche dans une séquence triée

$m = (d+f)/2$  : milieu de la plage de recherche



## Recherche dans une séquence triée

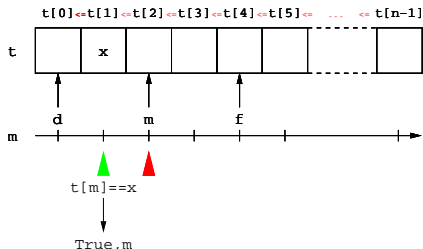
$m = (d+f)/2$  : milieu de la plage de recherche



- si  $x == t[m]$ , on a trouvé une solution et la recherche s'arrête ;

## Recherche dans une séquence triée

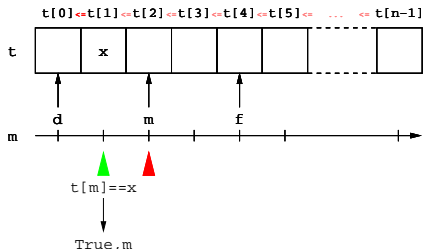
$m = (d+f)/2$  : milieu de la plage de recherche



- si  $x == t[m]$ , on a trouvé une solution et la recherche s'arrête ;
- si  $x < t[m]$ , poursuivre la recherche dans la moitié gauche de la liste ;

## Recherche dans une séquence triée

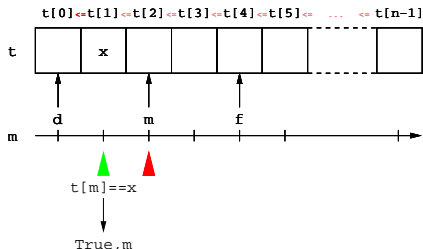
$m = (d+f)/2$  : milieu de la plage de recherche



- si  $x == t[m]$ , on a trouvé une solution et la recherche s'arrête ;
- si  $x < t[m]$ , poursuivre la recherche dans la moitié gauche de la liste ;
- si  $x > t[m]$ , poursuivre la recherche dans la moitié droite de la liste.

## Recherche dans une séquence triée

$m = (d+f)/2$  : milieu de la plage de recherche



- si  $x == t[m]$ , on a trouvé une solution et la recherche s'arrête ;
- si  $x < t[m]$ , poursuivre la recherche dans la moitié gauche de la liste ;
- si  $x > t[m]$ , poursuivre la recherche dans la moitié droite de la liste.

Complexité logarithmique ( $O(\log(n))$ )

## relation d'ordre total (notée $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

## relation d'ordre total (notée $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
---	---	---	---	---	---

## relation d'ordre total (notée $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
1	4	6	3	5	2

## relation d'ordre total (notée $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
---	---	---	---	---	---

1	4	6	3	5	2
---	---	---	---	---	---

1	2	6	3	5	4
---	---	---	---	---	---

## relation d'ordre total (notée $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
---	---	---	---	---	---

1	4	6	3	5	2
---	---	---	---	---	---

1	2	6	3	5	4
---	---	---	---	---	---

1	2	3	6	5	4
---	---	---	---	---	---

relation d'ordre total (notée  $\leq$ )

- réflexivité :  $x \leq x$
- antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
---	---	---	---	---	---

1	4	6	3	5	2
---	---	---	---	---	---

1	2	6	3	5	4
---	---	---	---	---	---

1	2	3	6	5	4
---	---	---	---	---	---

1	2	3	4	5	6
---	---	---	---	---	---

## relation d'ordre total (notée $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
1	4	6	3	5	2
1	2	6	3	5	4
1	2	3	6	5	4
1	2	3	4	5	6
1	2	3	4	5	6

## relation d'ordre total (notée $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
<b>1</b>	4	<b>6</b>	3	5	2
1	<b>2</b>	6	3	5	<b>4</b>
1	2	<b>3</b>	<b>6</b>	5	4
1	2	3	<b>4</b>	5	<b>6</b>
1	2	3	4	<b>5</b>	6
1	2	3	4	5	<b>6</b>

relation d'ordre total (notée  $\leq$ )

- 1 réflexivité :  $x \leq x$
- 2 antisymétrie :  $(x \leq y) \text{ and } (y \leq x) \Rightarrow x = y$
- 3 transitivité :  $(x \leq y) \text{ and } (y \leq z) \Rightarrow (x \leq z)$

6	4	1	3	5	2
1	4	6	3	5	2
1	2	6	3	5	4
1	2	3	6	5	4
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6

Complexité quadratique :  $O(n^2)$

**tri par insertion** trier successivement les premiers éléments de la liste

**tri par insertion** trier successivement les premiers éléments de la liste  
A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

**tri par insertion** trier successivement les premiers éléments de la liste  
A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

6	4	1	3	5	2
---	---	---	---	---	---

**tri par insertion** trier successivement les premiers éléments de la liste  
A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

6	4	1	3	5	2
4	6	1	3	5	2

**tri par insertion** trier successivement les premiers éléments de la liste  
A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

6	4	1	3	5	2
4	6	1	3	5	2
1	4	6	3	5	2

**tri par insertion** trier successivement les premiers éléments de la liste  
 A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

6	4	1	3	5	2
<b>4</b>	6	1	3	5	2
<b>1</b>	4	6	3	5	2
1	<b>3</b>	4	6	5	2

**tri par insertion** trier successivement les premiers éléments de la liste  
A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

6	4	1	3	5	2
4	6	1	3	5	2
1	4	6	3	5	2
1	3	4	6	5	2
1	3	4	5	6	2

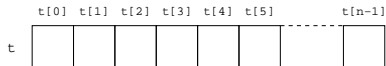
**tri par insertion** trier successivement les premiers éléments de la liste  
A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

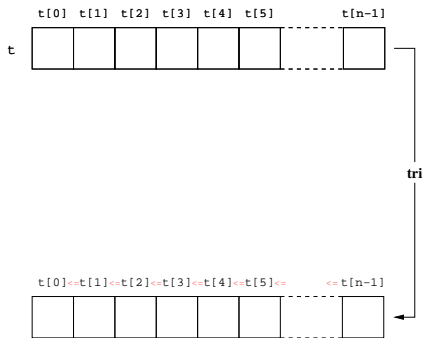
6	4	1	3	5	2
4	6	1	3	5	2
1	4	6	3	5	2
1	3	4	6	5	2
1	3	4	5	6	2
1	2	3	4	5	6

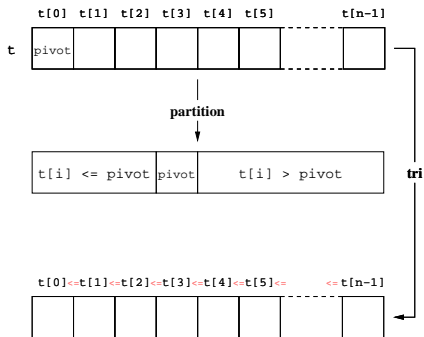
**tri par insertion** trier successivement les premiers éléments de la liste  
 A la  $i^{\text{ème}}$  étape, on insère le  $i^{\text{ème}}$  élément à son rang parmi les  $i - 1$  éléments précédents qui sont déjà triés entre eux.

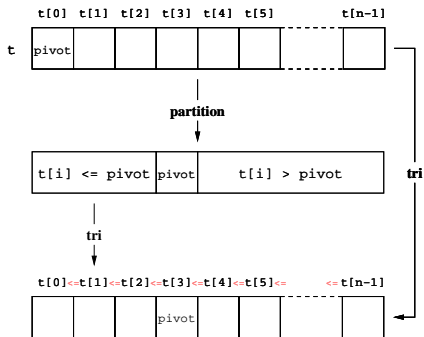
6	4	1	3	5	2
4	6	1	3	5	2
1	4	6	3	5	2
1	3	4	6	5	2
1	3	4	5	6	2
1	2	3	4	5	6

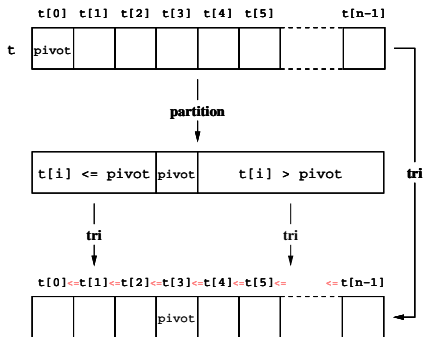
Complexité quadratique :  $O(n^2)$

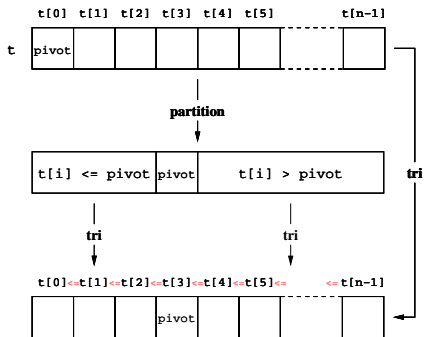




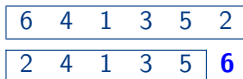
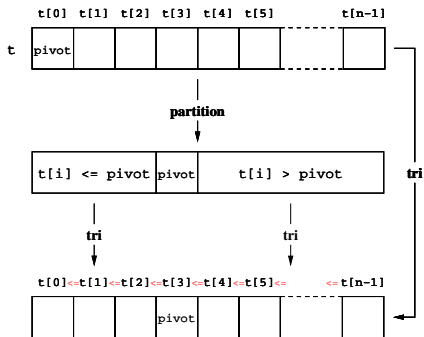


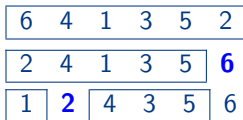
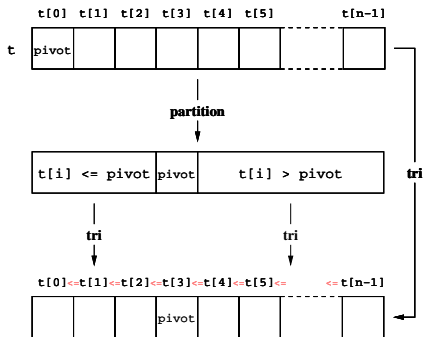


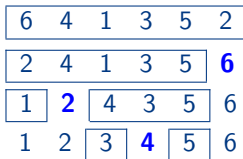
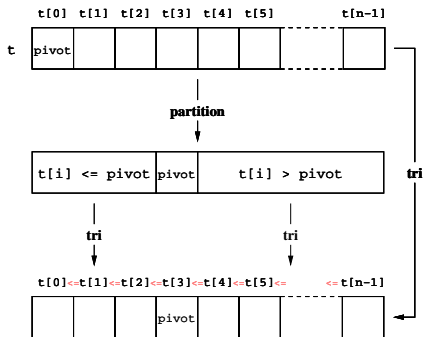


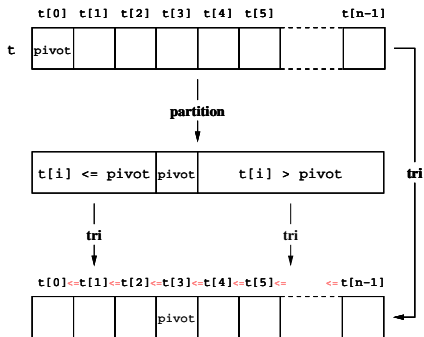


6 4 1 3 5 2

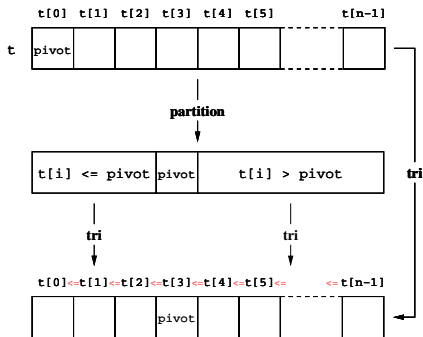








6	4	1	3	5	2
2	4	1	3	5	6
1	2	4	3	5	6
1	2	3	4	5	6
1	2	3	4	5	6



6	4	1	3	5	2
2	4	1	3	5	6
1	2	4	3	5	6
1	2	3	4	5	6
1	2	3	4	5	6

Complexité quasi-linéaire ( $O(n \log(n))$ )