Initiation à l'algorithmique

— structures linéaires —

Jacques TISSEAU

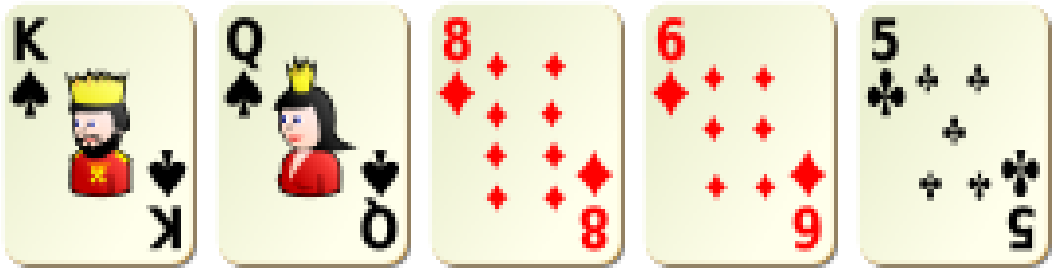
Enib–Cerv

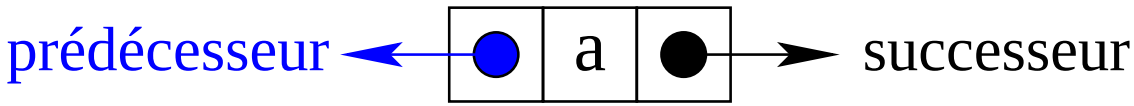
SÉQUENCES

DÉFINITION

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

Exemple de séquence : main au poker





SÉQUENCES

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

N-UPLETS

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 DE CARACTÈRES

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'
```

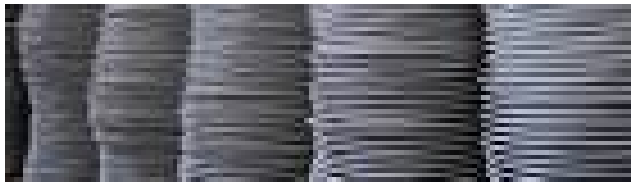
LISTES

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



- 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

LISTES

LISTES MULTIDIMENSIONNELLES

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
```

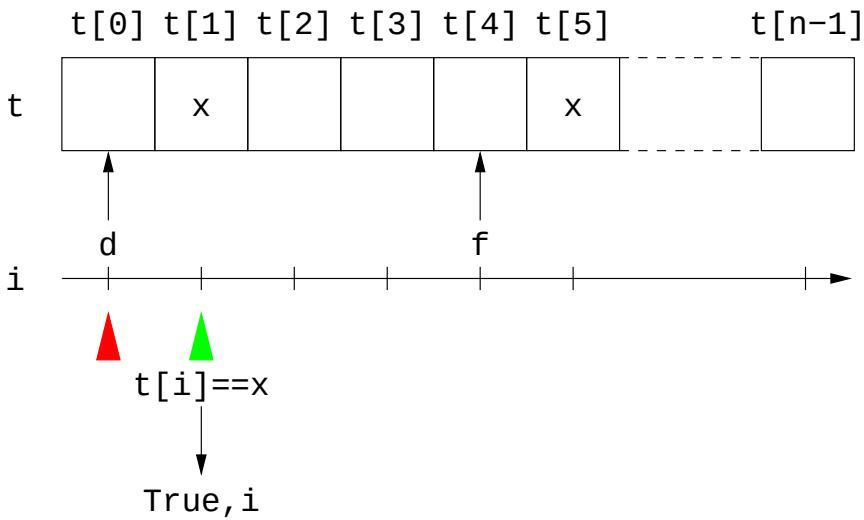
```
>>> s = [[4,5],[1,2,3],[6,7,8,9]]
>>> for c in s: print(c)
...
[4, 5]
[1, 2, 3]
[6, 7, 8, 9]
>>> s[2]
[6, 7, 8, 9]
>>> for c in s:
...     for e in c: print(e,end=' ')
...     print()
...
4 5
1 2 3
6 7 8 9
```

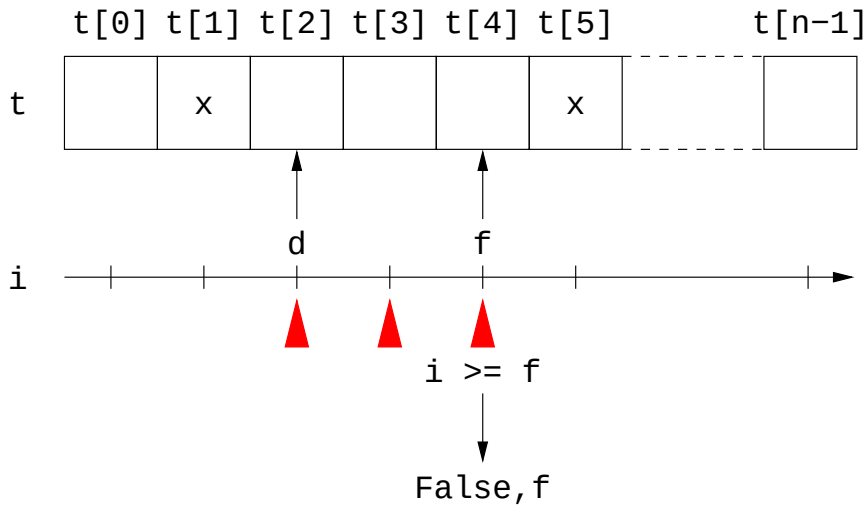
RECHERCHE D'UN ÉLÉMENT

RECHERCHE SÉQUENTIELLE

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

Recherche dans une séquence





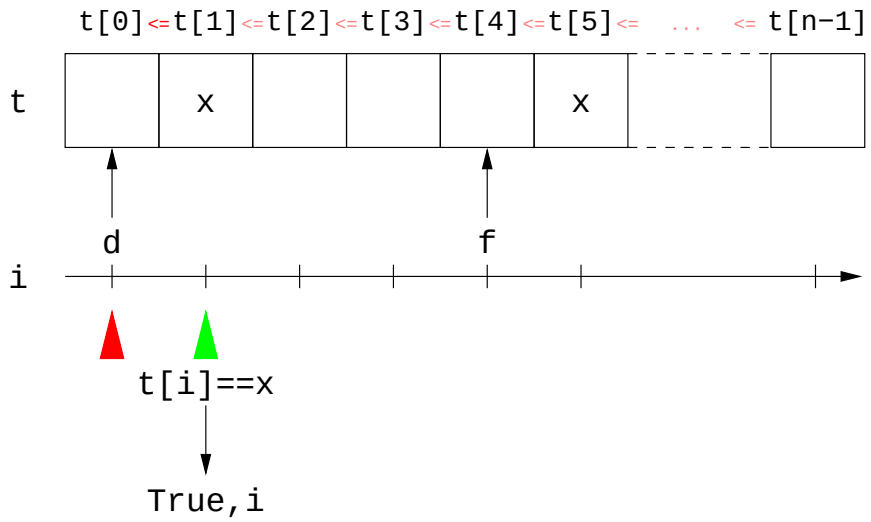
Complexité linéaire : $O(n)$

RECHERCHE D'UN ÉLÉMENT

RECHERCHE DICHOTOMIQUE

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 : $O(\log(n))$

TRI D'UNE SÉQUENCE

TRI PAR SÉLECTION

relation d'ordre total (notée $\leq$)

1. réflexivité : $x \leq x$
2. antisymétrie : $(x \leq y)$ and $(y \leq x) \Rightarrow x = y$
3. transitivité : $(x \leq y)$ 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 D'UNE SÉQUENCE

TRI PAR INSERTION

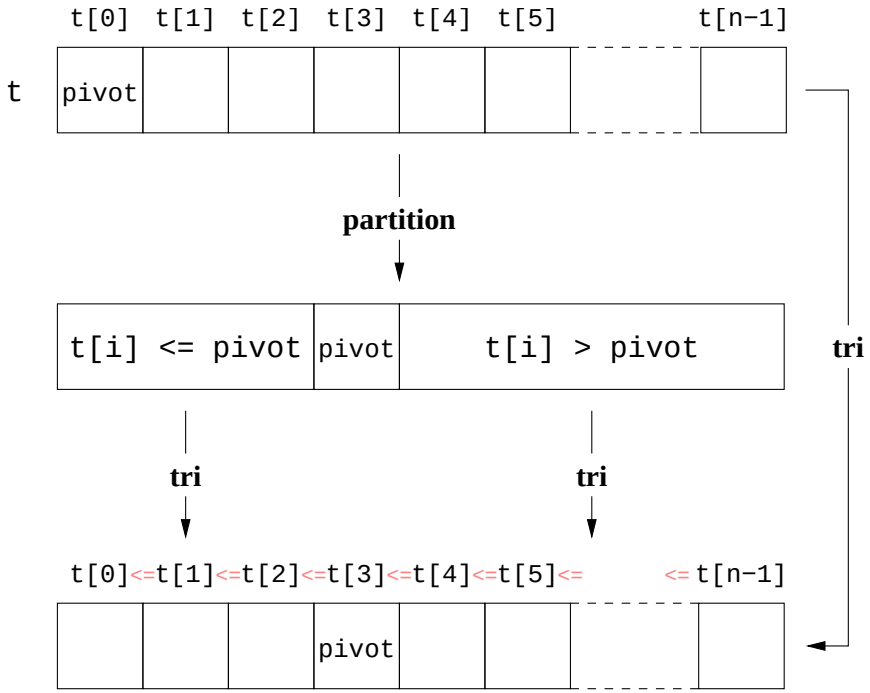
tri par insertion : trier successivement les premiers éléments de la liste
A la $i^{ème}$ étape, on insère le $i^{ème}$ élément à son rang parmi les $i-1$ éléments précédents qui sont déjà triés entre eux.

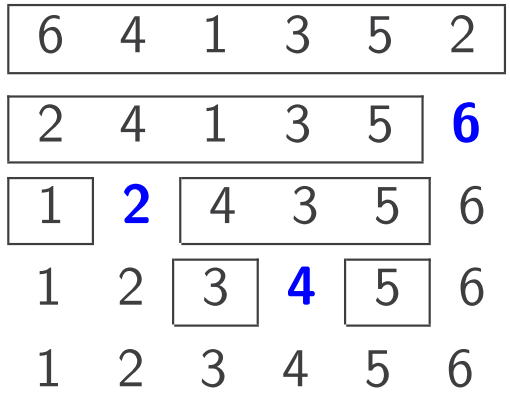
6	4	1	3	5	2
4	6	1	3	5	2
1	4	6	3	5	2
1	**3**	4	6	5	2
1	3	4	**5**	6	2
1	**2**	3	4	5	6

Complexité quadratique : $O(n^2)$

TRI D'UNE SÉQUENCE

TRI RAPIDE





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