TRI ET RECHERCHE :RESUME DU COURS

RECHERCHE SEQUENTIELLE :

Algorithme:	Implémentation en Python

fonction sequentielle (T: TAB ; N, x : entier) : booléen
début
[i ←0,rep← faux] **tantque** (i<N) et (non(rep)) **faire**
rep← T[i] = x
i←i+1
fin tantque
retourner rep
fin
	`def recherche_seq(T,N,x):`
	`  i, rep = 0, False`
	`  while i < N and not rep:`
	`  rep = T[i] == x`
	`  i += 1`
	`  return rep`

RECHERCHE DICHOTOMIQUE :

Principe

Cette méthode de recherche est applicable uniquement sur **une liste préalablement triée**. Le principe de cette méthode consiste à réduire à chaque fois l'espace de recherche sur la moitié de la liste. Le choix de l'une des deux parties dans laquelle se trouve la valeur cherchée est le résultat de l'évaluation d'un test.

A chaque évaluation, on coupera l'espace de recherche en deux parties égales (à un élément près) de part et d'autre de l'élément médiane.

Algorithme:	Implémentation en Python

fonction dichotomie (T: TAB ; N, x : entier) : booléen
début
[g ←0,d← n-1] répéter
m← (g+d) div2
Si **x** <T[m] alors
d ←m-1
sinon
g← m+1
fin si
jusqu'à (x=T[m]) ou (g>d)
si (**x** =T[m]) alors **retourner vrai**
sinon **retourner faux**
fin
	`def recherche_dicho(T,N,x):`
	`  deb, fin,rep = 0,N,False`
	`  while deb < fin and not rep:`
	`  m = (deb + fin) // 2`
	`  if T[m] == x:`
	`  rep=True`
	`  elif T[m] > x:`
	`  fin = m - 1`
	`  else:`
	`  deb = m + 1`
	`  return rep`

TRI PAR SELECTION :

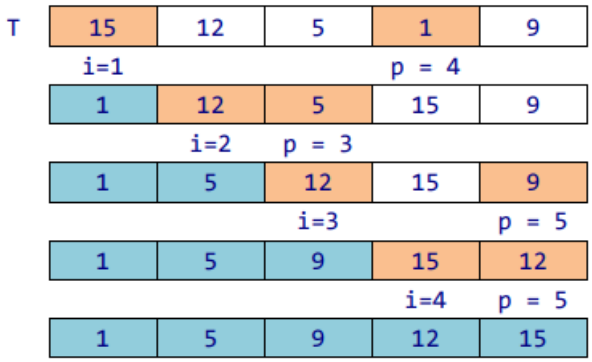
Principe

On recherche le plus petit (ou le plus grand) élément et on l'extrait, on recherche ensuite le plus petit des éléments qui restent et on l'extrait, et on recommence ainsi jusqu'à ce que l'on ait extrait tous les éléments.

Méthode

1. Sélectionner le premier élément du tableau
2. Parcourir le reste du tableau à la recherche de l'élément ayant la plus petite valeur.
3. Si le cas est présent permuter les deux éléments
4. Incrémenter le premier indice
5. Refaire les étapes 2, 3, et 4 jusqu'à la fin du tableau.

Exemple: Trier dans l'ordre croissant le tableau suivant:



Algorithme:	Implémentation en Python
procédure Tri_selection (@ T : tab ; n : entier) **début** pour i de 0 à n-1 faire [posmin ← i] pour j de i+1 à n faire si T[j]< T[posmin] alors posmin ← j fin si fin pour si posmin ≠ i alors aux← T[i] T[i]← T[posmin] T[posmin]←aux Fin si Fin pour **Fin**	**def selection(T,debut) :** indiceDuMin=debut for k in range(debut+1,len(T)) : if T[k]< T[indiceDuMin] : indiceDuMin=k if indiceDuMin !=debut : **T[debut],T[indiceDuMin]=T[indiceDuMin],T[debut]** **def triSelection(T) :** for j in range(0,len(T)-1) : selection(T,j)

Implémentation en Python version 2 :

```
def imin(T,a,b): #fonction qui renvoie la position de la valeur minimale de T
#entre les positions a et b
    imin=a
    N=len(T)
    for i in range(a+1,b):
        if T[i] < T[imin]:
            imin = i
    return imin

def triselection(T):
    N=len(T)
    for i in range(N-1):
        j=imin(T,i,N)
        T[i],T[j] = T[j],T[i]
    return T
```

TRI A BULLES :

Principe :

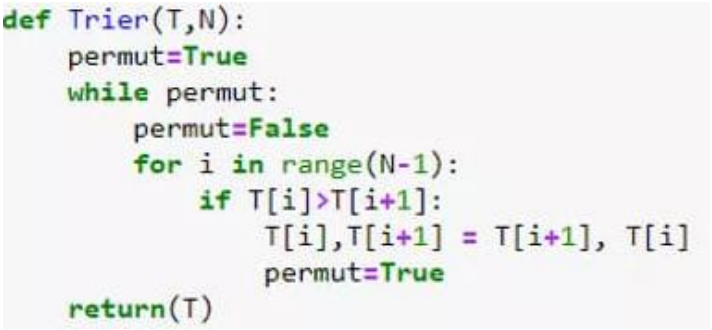
A chaque étape on considère un couple d'éléments; s'ils ne sont pas dans le bon ordre l'un par rapport à l'autre, on les permute; on répète ce procédé en changeant de couple et ce, jusqu'à ce que plus aucun échange ne soit nécessaire.

Méthode :

On effectue des passages successif sur le tableau on examinent les éléments du tableau par paire V[i] et V[i+1] et on échange les valeurs si V[i]>V[i+1], chaque passage s'arrête au niveau du dernier échange du passage précédent ainsi le 1ér passage est allé jusqu'à V[N]. On arrête les passages dés que l'un d'entre eux n'entraîne aucun échange.

Exemple :

T	15	3	5	1	10
	i=1	i+1=2			
	3	15	5	1	10
		i=2	i+1=3		
	3	5	15	1	10
			i=3	i+1=4	
	3	5	1	15	10
				i=4	i+1=5
	3	5	1	10	15
	i=1	i+1=2			
	3	5	1	10	15
		i=2	i+1=3		
	3	1	5	10	15
			i=3	i+1=4	
	3	1	5	10	15
				i=4	i+1=5
	3	1	5	10	15
	i=1	i+1=2			
	1	3	5	10	15
		i=2	i+1=3		
	1	3	5	10	15
		i=2	i+1=3		
	1	3	5	10	15
	i=1	i+1=2			

Algorithme:	**Implémentation en Python**
PROCEDURE Tri_bulle (@ T : tab ; n : entier) **début** **Répéter** [Echange←faux] pour i de 0 à n-1 faire Si (T[i]>T[i+1]) alors Aux← T[i] T[i]← T[i+1] T[i+1]←aux Echange← vrai Fin Si Fin pour N←n-1 **Jusqu'à (n=0) ou non(Echange)** **fin**	

```python
def Trier(T,N):
    permut=True
    while permut:
        permut=False
        for i in range(N-1):
            if T[i]>T[i+1]:
                T[i],T[i+1] = T[i+1], T[i]
                permut=True
    return(T)
```