AP08 TRIS PAR INSERTION ET SÉLECTION

I. Tri par insertion

https://www.youtube.com/watch?v=ROalU379l3U

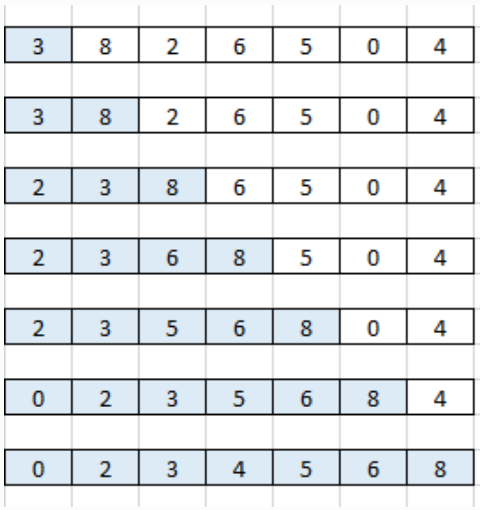
https://physalgo.fr/Dariush_Tris/index.html

Ce type de tri est le système que vous utilisez naturellement lorsqu'on vous distribue des cartes une par une et que vous voulez les trier par ordre croissant.

Au fur et à mesure que vous recevez une carte (la **clé**), vous lui trouvez la bonne place en la comparant aux cartes déjà triées. Cette carte va se décaler vers la gauche jusqu'à trouver sa bonne place.

Propriété 1 : Le tri par insertion

1. La liste contenant un seul élément d'indice 0 est triée.
2. La clé est le premier élément non trié ;
3. On compare la clé avec l'élément à sa gauche si on n'a pas atteint le bord gauche ;
4. Échange des éléments si la clé est inférieure à l'élément de gauche ;
5. Retour à l'étape 3
6. La partie triée gagne un élément ;
7. Retour à l'étape 2 tant qu'il reste des éléments à trier.



Etape 1 :

- au départ la liste triée est constituée de l'élément 3 ;
- la clé est le premier élément non trié (8).
- la clé est supérieure à l'élément de gauche(3) ;
- la clé est déposée à l'indice 1 de la liste ;
- la partie triée gagne un élément (8) ;

Etape 2 :

- la liste triée est donc maintenant `[3, 8]` ;
- la clé est le premier élément non trié (2).
- la clé est inférieure à l'élément de gauche(8) ⇒ la clé est échangée avec l'élément de gauche (8) ;
- la clé est inférieure à l'élément de gauche(3) ⇒ la clé est échangée avec l'élément de gauche (3) ;
- la clé est maintenant tout à gauche de la liste ⇒ la clé est déposée à l'indice 0 de la liste ;
- la partie triée gagne un élément (2) ;

Etape 3 :

- la liste triée est donc maintenant `[2, 3, 8]` ;
- la clé est le premier élément non trié (6).
- la clé est inférieure à l'élément de gauche(8) ⇒ la clé est échangée avec l'élément de gauche (8) ;
- la clé est supérieure à l'élément de gauche(3) ⇒ la clé est déposée à l'indice 2 de la liste ;
- la partie triée gagne un élément (6) ;

Exercice 1

Tri par insertion

1. Compléter le programme suivant afin d'effectuer le tri par insertion. Cette fonction modifie la liste en place et ne renvoie donc rien.

Code Python

```
def tri_insertion(L):
    ''' Algorithme de tri par insertion '''
    n = len(L)
    for i in range(...., ....):
        # Indice de la clé
        k = ....
        # Tant que le bord de la liste n'est pas atteint
        # Et que la clé est inférieure à l'élement de gauche:
        while ..........................................:
            # Échange de la clé avec l'élément de gauche
            ..................................................
            # Décalage de la clé vers la gauche
            ......................

L = [8, 4, 1, 0, -5, 6, 7]                  # Liste non triée
tri_insertion(L)                          # Tri de la liste
assert L == [-5, 0, 1, 4, 6, 7, 8]        # La liste est triée
```

2. Effectuer la trace de cet algorithme pour `L=[3, 6, 0, 2]`. Recopier le tableau ci-dessous et le compléter au fur et à mesure de l'exécution de l'algorithme.

Ligne	L[0]	L[1]	L[2]	L[3]	i	k	L[k-1]	L[k]	while

3. Quel est le variant de boucle? L'algorithme se termine-t-il? ....
.....
.....
.....
.....

4. Quel est l'invariant de boucle? L'algorithme est-il correct? .....
.....
.....
.....
.....

5. Déterminer la complexité de cet algorithme .....

II. Tri par sélection

https://www.youtube.com/watch?v=Ns4TPTC8whw

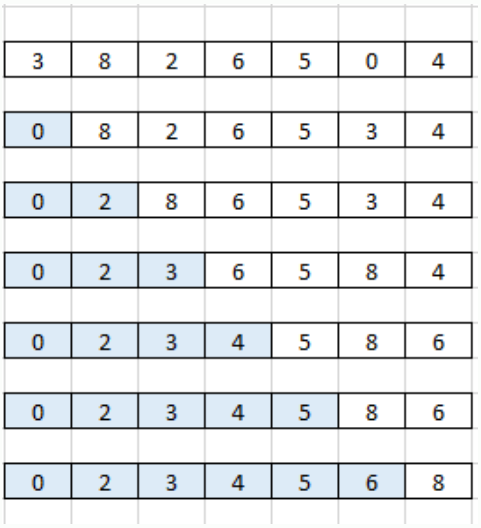
https://physalgo.fr/Dariush_Tris/index.html

Ce type de tri est le système que vous utilisez lorsqu'on vous distribue toutes les cartes d'un coup.

Au départ la liste triée comporte 0 éléments. On cherche le plus petit élément de la liste non triée et on l'échange avec le premier élément de la liste non triée. La partie triée gagne un élément. On recommence tant qu'il reste des éléments à trier.

Propriété 1 : Le tri par sélection

1. la liste triée comporte 0 éléments et la liste non triée n
2. on cherche le minimum de la liste non triée que l'on échange avec le premier élément de la liste non triée ;
3. la liste triée augmente de 1 élément
4. on recommence l'étape 1 jusqu'à ce que la liste triée comporte $n-1$ éléments (le dernier élément est forcément trié).



Etape 1 :

- au départ la liste triée est constituée d'aucun élément ;
- la liste non triée est `[3, 8, 2, 6, 5, 0, 4]`
- le premier élément de la liste non triée est 3 ;
- on cherche le minimum de la liste non triée qui est 0 ;
- on échange 3 et 0 ;
- la liste triée gagne un élément (0) ;
- la liste triée est maintenant `[0]` et la liste non triée est `[8, 2, 6, 5, 3, 4]` ;

Etape 2 :

- la liste triée est donc maintenant `[0]` ;
- le premier élément de la liste non triée est 8 ;
- on cherche le minimum de la liste non triée qui est 2 ;
- on échange 8 et 2 ;
- la liste triée gagne un élément (2) ;
- la liste triée est maintenant `[0, 2]` et la liste non triée est `[8, 6, 5, 3, 4]` ;

Etape 3 :

- la liste triée est donc maintenant `[0, 2]` ;
- le premier élément de la liste non triée est 8 ;
- on cherche le minimum de la liste non triée qui est 3 ;
- on échange 8 et 3 ;
- la liste triée gagne un élément (3) ;
- la liste triée est maintenant `[0, 2, 3]` et la liste non triée est `[8, 6, 5, 4]` ;

Exercice 2

Tri par sélection

1. Compléter le programme suivant afin d'effectuer le tri par sélection. Cette fonction modifie la liste en place et ne renvoie donc rien.

Code Python

```
def tri_selection(L):
    ''' Algorithme de tri par sélection '''
    n = len(L)
    for i in range(...., ....):
        # Recherche du plus petit element de la liste non triee
        mini = i          # indice de plus petit element non triee
        for j in range(...., ....):
            if ..............................:
                mini = j          # j est le nouvel indice du minimum
        # Echange entre L[i] et L[mini]
        ..............................................

L = [8, 4, 1, 0, -5, 6, 7]                    # Liste non triée
tri_selection(L)                             # Tri de la liste
assert L == [-5, 0, 1, 4, 6, 7, 8]           # La liste est triée
```

2. Effectuer la trace de cet algorithme pour `L=[3, 6, 0, 2]`. Recopier le tableau ci-dessous et le compléter au fur et à mesure de l'exécution de l'algorithme.

Ligne	L[0]	L[1]	L[2]	L[3]	i	mini	j	L[j]	L[mini]	IF

3. L'algorithme se termine-t-il? ....
4. Quel est l'invariant de boucle? L'algorithme est-il correct? .....
   .....
   .....
   .....
   .....
5. Déterminer la complexité de cet algorithme .....