ÉVALUATION SUR LE TRI PAR SÉLECTION

Partie 1 - Principe du tri par sélection

On souhaite trier une **suite** de n **cartes** munies d'un ordre total (on peut les **comparer**).
À chaque carte est associé un **indice** allant de **0** à $n-1$.

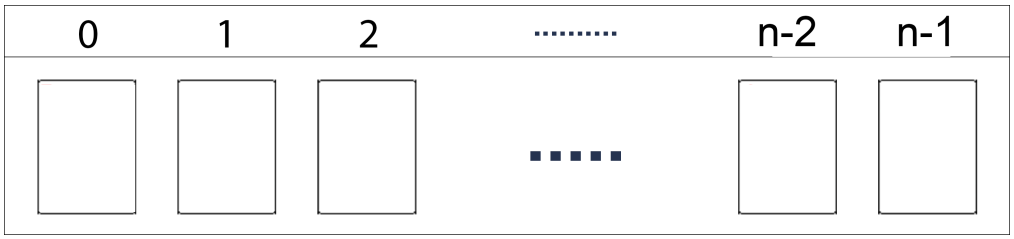

Figure 1: une suite de cartes

L'objectif est de trier ces cartes de la plus **faible** à la plus **forte**.

Exercice 1

Complétez le principe du **tri par sélection** du **minimum** d'une **suite de cartes** suivant (vous pouvez écrire sur cette feuille directement) :

- Chercher l'indice i de la carte la plus ..... de la suite à partir de l'indice ..., puis échanger la carte d'indice ... avec celle d'indice ... . La ..... carte est maintenant la plus **faible**.
- Chercher l'indice i de la carte la plus ..... de la suite à partir de l'indice ..., puis échanger la carte d'indice ... avec celle d'indice ... . Les ... premières cartes sont maintenant **triées**.
- [...]
- Chercher l'indice i de la carte la plus ..... de la suite à partir de l'indice $n-2$, puis échanger la carte d'indice ... avec celle d'indice ... . Les ... cartes sont maintenant **triées** (la dernière carte étant **déjà** à sa bonne position).

On s'intéresse maintenant à des **tableaux d'entiers**, que l'on souhaite **trier** dans l'ordre croissant.

Exercice 2

Déroulez le **tri par sélection** du **minimum** sur le **tableau d'entiers** `[8, 3, 2, 7, 1, 5]` en vous aidant du tableau suivant. Écrivez en bleu les **entiers triés** et **en noir** ceux qui ne le sont **pas encore**.

Indice	**0**	**1**	**2**	**3**	**4**	**5**
Recherche du minimum...	8	3	2	7	**1**	5
Après permutation...	1	3	2	7	8	5
Recherche du minimum...						
Après permutation...						
Recherche du minimum...						
Après permutation...						
Recherche du minimum...						
Après permutation...						
Recherche du minimum...						
Après permutation...						

Partie 2 - Algorithme

On souhaite écrire l'algorithme du **tri par sélection** du **minimum**, permettant de trier **en place** un tableau d'éléments comparables.

Exercice 3

Que signifie "trier **en place**" ?

On séparera l'**algorithme** en **3 sous-algorithmes** :
- l'algorithme principal `tri_selection` effectuant le **tri par sélection** en place d'un *tableau*,
- un algorithme `minimum` permettant la recherche du **minimum** dans un *tableau* à partir d'un **indice** donné,
- un algorithme `echanger` permettant d'**échanger** deux éléments d'indices donnés.

Exercice 4

Complétez les algorithmes de `minimum` et `echanger` suivants (vous pouvez écrire sur cette feuille) :

minimum

ALGORITHME : minimum
ENTRÉES :
`tableau` : un **tableau** d'éléments
`debut` : .....
SORTIE : l'**indice** de l'élément de **valeur minimale** dans l'intervalle `[debut, longueur(tableau)-1]` du tableau
DÉBUT
indice_min ← debut
POUR i **ALLANT DE** ..... **À** .....
SI tableau[i] < ....., **ALORS**
indice_min ← ...
FIN POUR
Renvoyer indice_min
FIN ALGORITHME

echange

ALGORITHME : echanger
ENTRÉES :
`tableau` : un **tableau** d'éléments
`i` : l'**indice** d'un élément du tableau
`j` : l'**indice** d'un autre élément
SORTIE : .....
DÉBUT
..... ← tableau[i]
..... ← tableau[j]
..... ← .....
Renvoyer **RIEN**
FIN ALGORITHME

Exercice 5

Enfin, **écrivez** l'algorithme `tri_selection` effectuant le tri par sélection (en place) d'un tableau.

tri_selection

ALGORITHME : tri_selection
ENTRÉES :
`tableau` : un **tableau** d'éléments pouvant être **comparés**
SORTIE : aucune (tri *en place*)
DÉBUT
n ← longueur(tableau)
POUR .....
indice_min ← .....
.....
FIN POUR
Renvoyer **RIEN**
FIN ALGORITHME

Partie 3 - Implémentation du tri par sélection

Exercice 6

Implémentez les 3 algorithmes `tri_selection`, `minimum` et `echanger` en **Python**.
Les **en-têtes** de ces fonctions sont données ci-dessous (il n'y a pas obligation de ré-écrire la *docstring*) :

```python
def minimum(tableau: 'list[int]', debut: int) -> int:
    ''' Renvoie l'indice de la valeur minimale du tableau dans l'
       intervalle [debut, len(tableau) - 1].
    :param tableau: (list[int]) un tableau d'entiers
    :param debut: (int) l'indice à partir duquel on recherche le
       minimum
    :return: (int) l'indice du minimum '''

    ...

def echanger(tableau: 'list[int]', i: int, j: int) -> None:
    ''' Échanger deux éléments d'un tableau
    :param tableau: (list[int]) un tableau d'entiers
    :param i: (int) l'indice d'un élément du tableau
    :param j: (int) l'indice d'un élément du tableau '''

    ...

def tri_selection(tableau: 'list[int]') -> None:
    ''' Effectue le tri par sélection en place des éléments d'un
       tableau donné.
    :param tableau: (list[int]) un tableau d'entiers à trier '''

    ...
```

Exercice bonus

Ré-implementer l'algorithme du **tri par sélection** mais cette fois **en une seule fonction** `tri_selection`.

Partie 4 - Coût du tri par sélection

On souhaite maintenant étudier le **coût algorithmique** du **tri par sélection**.
On déterminera ce coût en comptant le **nombre de comparaisons** entre éléments effectué.

Exercice 7

On souhaite calculer le **nombre de comparaisons** effectué pour trier le tableau de l'exercice 2 : `[8, 3, 2, 7, 1, 5]`.

Complétez le tableau suivant en indiquant le **nombre de comparaisons** effectué après **chaque itération** de la **boucle principale** de l'algorithme `tri_selection`, c'est-à-dire le **nombre de comparaisons** effectué par l'algorithme `minimum(tableau, debut)` pour chaque valeur de `debut` de 0 à $n-2$, n étant la **longueur du tableau** (= **nombre d'éléments** dans le tableau).

itération k =	debut	tableau après itération k	nombre de comparaisons
0 (avant la boucle)	/	[8, 3, 2, 7, 1, 5]	/
1			
2			
3			
4			

Quel est ainsi le **nombre total de comparaisons** effectué pour ce tableau de taille $n = 6$?

Exercice 8

Si l'on change les éléments contenus dans le tableau précédent, sans changer sa taille, le **nombre de comparaisons** changera t-il ? Pourquoi ?

Exercice 9

Complétez le tableau suivant en indiquant le **nombre de comparaisons** effectué pour trier un tableau de taille 2, 3, 4, 6 et 8.

tableau de taille n =	nombre de comparaisons
2	
3	
4	
6	
8	

Complétez cette phrase :
"Lorsque l'on **double** le nombre d'éléments du tableau d'entrée, on ..... le nombre de **comparaisons** effectuées.

Déduisez-en le **coût algorithmique** du tri par sélection. Est-il *linéaire*, *quadratique*, *quasi-linéaire*, *logarithmique*… ?

Exercice 10

Calculez le **nombre de comparaisons**, qu'on notera $C(n)$, nécessaire pour trier un **tableau** de taille n avec la méthode de **tri par sélection**.