Fiche d'identité de l' algorithme de **TRI par SÉLECTION (ou *tri par extraction*)**

Principe : on va chercher le plus petit élément de la liste pour le mettre en …....., puis on repart du …..... élément et on va chercher le plus petit élément de la liste pour le mettre en …....., etc...

Exemple : le photographe, qui place les enfants sur le banc en fonction de leurs tailles,passe en revue les enfants à la recherche du plus petit. Une fois trouvé, il le place sur le banc. Il réitère cette opération sur les enfants restant en plaçant à chaque fois l'enfant sélectionné à la prochaine place disponible sur le banc.

Pseudo Code	**Python :** procédure
	….. …..
n ← nombre d'éléments de t	…..
pour i allant de 1 à n-2	…..
i_min ← minimum(t,j) // appel de la fonction	…..
minimum : renvoi son indice	…..
si i_min différent de j alors	…..
permuter t[i] et t[j]	…..
fin si	…..
fin pour	…..

Algorithme valide

On choisit comme invariant de boucle H : « la liste t[0 : i+1] est triée par ordre croissant à l'issue de l'itération i »

Initialisation (**est ce vrai avant d'entrer dans la boucle** ?)

Conservation = hérédité (**reste vraie après une itération, si elle était vraie avant**)

Terminaison (**donne le résultat attendu en fin de boucle**)

Complexité en temps : on s'intéresse au nombre de comparaisons effectuées. La liste est triée par ordre croissant. La boucle *pour* de la fonction *tri* effectue n-1 opérations. A chaque itération, la fonction *minimum* est appelée et effectue n-(j+1) itérations. Sa complexité est donc en $\Theta(n^2)$ de l'ordre de $n \times n$

Formalisation

1°) Faire fonctionner à la main le tri par selection sur un nouveau jeu de 4 cartes

puis sur une liste, [9,3,1,6] par exemple qui pourra être aussi illustrée si besoin par les 4 cartes de valeur correspondantes → Implémenter alors en pseudo code puis en Python cet algorithme

2°)

Quel est le pire des cas ? Combien y-a-il alors d'itérations dans la boucle *pour*? Combien y-a-il d'appel à la fonction minimum dans chaque tour de boucle? Combien d'itérations effectue cette dernière fonction ?

3°) Définir et trouver un *Invariant de boucle*

Exercices

Exercice 1 – *ANALYSER et CONCEVOIR*

On souhaite (ENCORE!) trier la liste de listes *Pers* selon le numéro situé en 2ème position dans les sous-listes que contient *Pers* :

```
Pers =[['Portillon',4],['Sam',3],['Julie',1],['Tom',2],['Charlie',5]]
```

Proposer une fonction mettant en œuvre un algorithme de tri par sélection permettant de trier la liste *Pers*

Exercice 2 – *ANALYSER et TRADUIRE*

On souhaite maintenant trier la liste de listes *Pers* selon le numéro situé en 2ème position dans les sous-listes que contient *Pers* :

```
Pers =[['Sam',3],['Julie',1],['Tom',2],['Portillon',4],['Charlie',5]]
```

Mais cette fois ci, seul un élément est mal trié. Il s'agit de la liste ['Sam',3] qui correspond à l'indice 0 dans la liste *Pers* . Proposer une fonction mettant en œuvre un algorithme de tri conçu par vous même permettant de trier la liste *Pers.*
On ne remplacera que l'élément mal placé, sans trier toute la liste.
(cette situation peut être illustrée par le rangement d'un livre selon son titre dans une bibliothèque dont les ouvrages sont classés par ordre alphabétique : on ne trie pas toute la bibliothèque mais on insère l'ouvrage à sa place)

Aides : *del Pers[0]* supprime l'élément mal placé
Pers.insert(i,mal_placee) insère la liste *mal_placee* (à déterminer) à l'indice *i* (à déterminer)

Exercice 3 – *ANALYSER et TRADUIRE*

En supposant que le tri par sélection prend un temps directement proportionnel à n^2 et qu'il prend 6,8 secondes pour trier 16000 valeurs (voir tableau des temps de ce tri), calculer le temps qu'il faudrait pour trier un million de valeurs

Comparer ces temps avec les tris de Python

On pourra générer aléatoirement des listes de 16000 éléments (liste_1) entre 1 et 10 000, puis de 1 million d'éléments (liste_2) toujours entre 1 et 10 0000 et comparer à l'aide du petit programme de comparaison d'écoulement du temps les temps de tri de *liste*.sort() ; un tri en place et sorted(*liste*) qui crée, elle, une copie triée dans Python.

```
from time import time
debut = time()
# Code dont on mesure le temps
fin = time()
print("Temps passé : ",fin - debut)
```