Fiche d'identité de l' algorithme de **TRI par INSERTION**

**Principe** : on parcourt les éléments de la liste et on …..... → tri en place

**Exemple** : le tri du jeu de cartes, on range une carte piochée dans son jeu déjà classé

| **Pseudo Code** | **Python :** procédure |
|---|---|
| n ← nombre d'éléments de t<br>pour i allant de 1 à n-1<br>  tant que i>0 et t[i]<t[i-1]<br>  permuter t[i] et t[i-1]<br>  i ← i -1<br>  fin tant que<br>fin pour | …..<br>…..<br>…..<br>…..<br>…..<br>…..<br>…..<br>…..<br>….. |

**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** : algorithme lent mais relativement efficace sur de courtes listes (utilisé pour …..)

Dans le pire des cas, avec des données triées à l'envers, les parcours successifs du tableau imposent d'effectuer *(n-1)+(n-2)+(n-3) +...+1* comparaisons et échanges, soit (FORMULE MATH) .....

On a donc une complexité dans le pire des cas du tri par insertion en $\Theta(n^2)$ : c'est un coût **quadratique**

# Exercices

**Exercice 1** – *ANALYSER et CONCEVOIR*

On souhaite 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 insertion permettant de trier la liste *Pers*

**Exercice 2** – *CONCEVOIR et COMPARER*

Afin de comparer les coûts en temps des méthodes de tri, proposer une fonction qui génère aléatoirement un tableau de taille n choisi par l'utilisateur d'entiers positifs.

Compléter à l'aide la fonction déjà utilisée lors de l'introduction du cours d'algorithmique

```
from time import time
debut = time()

# Code dont on mesure le temps

fin = time()
print("Temps passé : ",fin - debut)
```

le tableau ce dessous et comparer avec vos camarades (selon la taille et la machine utilisée)

| Coefficients multiplicateurs | Taille n du tableau | Temps Tri insertion | Coefficients multiplicateurs |
|---|---|---|---|
| | 10 | | |
| | 100 | | |
| | 1000 | | |
| | 5000 | | |
| | 10000 | | |

Si la taille est multipliée par k, le temps de tri semble multiplié par ….....