Exercices tris
Correction

Christophe Viroulaud

Première - NSI

Algo 05

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Exercice 1

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

carré du nombre éléments	16000^2	1000000^2
durée	6,8	

$$\frac{6,8 \times 1000000^2}{16000^2} = 26560s = 7h23min$$

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Exercice 2

```
1  def tri_insertion(tab: list) -> None:
2      for i in range(len(tab)):
3          # mémoriser
4          en_cours = tab[i]
5          pos = i
6          # décaler
7          while pos > 0 and en_cours < tab[pos-1]:
8              tab[pos] = tab[pos-1]
9              pos = pos-1
10         # insérer
11         tab[pos] = en_cours
```

```
1  tab = [randint(0, 100) for _ in range(10)]
2  tri_insertion(tab)
3  print(tab)
```


Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

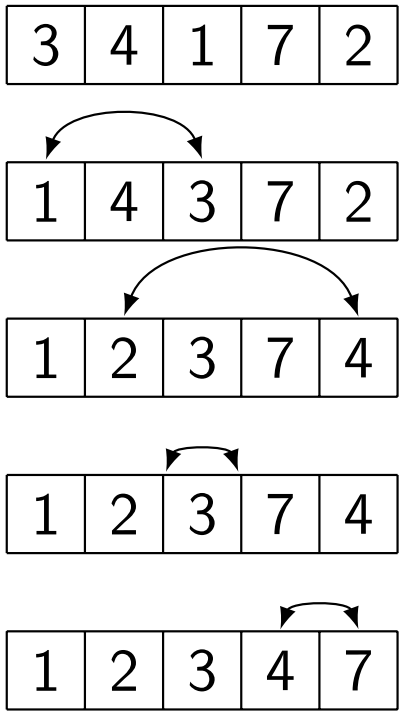
Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

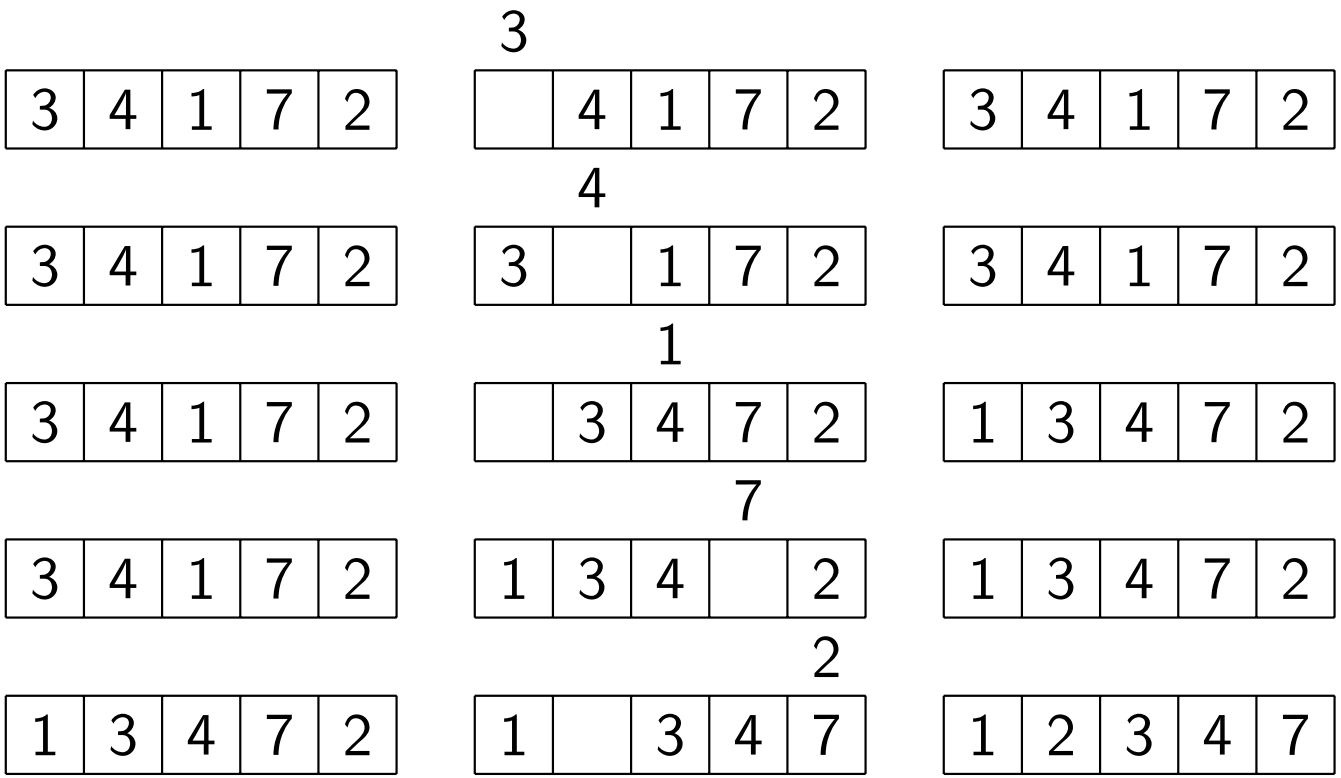
Exercice 3

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8



Code 1 – Tri par sélection

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8



Code 2 – Tri par insertion

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Exercice 4

```python
def comparer(tab1: list, tab2: list) -> bool:
    for i in range(len(tab1)):
        if not tab1[i] == tab2[i]:
            # stoppe à la première différence
            return False
    # tous les éléments ont été comparés
    return True
```

```python
t1 = [3, 5, 9, 0, 1, 8, 2]
t2 = [9, 5, 3, 2, 8, 1, 0]
tri_insertion(t1)
tri_insertion(t2)
print(comparer(t1, t2))
```


Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Exercice 5

Exercice 1
Exercice 2
ercice 3
ercice 4
ercice 5
ercice 6
ercie 7
ercice 8

```
1  def tri_insertion(tab: list) -> list:
2      tab_trie = []
3      for i in range(len(tab)):
4          # mémoriser
5          en_cours = tab[i]
6          tab_trie.append(en_cours)
7          pos = len(tab_trie)-1
8          # décaler
9          while pos > 0 and en_cours < tab_trie[pos-1]:
10             tab_trie[pos] = tab_trie[pos-1]
11             pos = pos-1
12         # insérer
13         tab_trie[pos] = en_cours
14     return tab_trie
```

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

```
1 t = [randint(0, 100) for _ in range(10)]
2 print(tri_insertion(t))
3 # le tableau initial n'est pas modifié
4 print(t)
```

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Exercice 6

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

```
1  def echanger(tab: list, i: int, j: int) -> None:
2      temp = tab[i]
3      tab[i] = tab[j]
4      tab[j] = temp
5
6  def inserer(tab: list, j: int) -> None:
7      # Le changement se fait dans la comparaison
8      while j-1 >= 0 and tab[j-1][0] > tab[j][0]:
9          echanger(tab, j-1, j)
10         j = j-1
11
12 def tri_insertion(tab: list) -> None:
13     for i in range(len(tab)):
14         inserer(tab, i)
```

À retenir

Le tri par insertion est stable. Ce n'est pas le cas du tri

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

```python
def max_occurrences(tab: list) -> tuple:
    tri_insertion(tab)
    # départ 1° série
    en_cours = tab[0]
    serie_en_cours = 1
    elt_max = tab[0]
    serie_max = 1
    for i in range(1, len(tab)):
        # cas même élément que le précédent
        if en_cours == tab[i]:
            serie_en_cours += 1
        else:
            # vérifie alors la taille de la dernière série
            if serie_en_cours > serie_max:
                serie_max = serie_en_cours
                elt_max = en_cours
            # départ nouvelle série
            en_cours = tab[i]
            serie_en_cours = 1
    return (elt_max, serie_max)
```

ce 1
ce 2
ce 3
ce 4
ce 5
ce 6
e 7
ce 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
ercice 6
ercie 7
ercice 8

```
1 t = [randint(0, 10) for _ in range(100)]
2 maxi = max_occurrences(t)
3 print(f"Le nombre {maxi[0]} est apparu {maxi[1]} fois.")
```

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4
5. Exercice 5
6. Exercice 6
7. Exercie 7
8. Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Exercice 8

```
1  def echanger(tab: list, i: int, j: int) -> None:
2      """
3      échange deux éléments de tab
4      """
5      temp = tab[i]
6      tab[i] = tab[j]
7      tab[j] = temp
```


Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

Exercice 1
Exercice 2
Exercice 3
Exercice 4
Exercice 5
Exercice 6
Exercie 7
Exercice 8

```
1 def tri_bulles(tab: list) -> None:
2     for i in range(len(tab)):
3         # on s'arrête avant la partie déjà triée
4         for j in range(1, len(tab)-i):
5             # remonte l'élément le plus grand
6             if tab[j-1] > tab[j]:
7                 echanger(tab, j-1, j)
```