Exercices Diviser pour régner

Christophe Viroulaud

Terminale - NSI

Algo 02

Sommaire

Exercice 1

```python
def inserer(tab: list, j: int) -> None:
    if j >= 0 and tab[j] > tab[j+1]:
        tab[j], tab[j+1] = tab[j+1], tab[j]
        inserer(tab, j-1)


def tri_insertion_rec(tab: list) -> None:
    for i in range(len(tab)):
        inserer(tab, i-1)
```

Sommaire

Exercice 2

La fonction **`inserer`** fait *redescendre* l'élément de rang *j* en le comparant avec celui de rang *j-1*. Cette propagation **de proche en proche** s'arrête quand les deux éléments sont égaux. L'ordre relatif est préservé.

```
t = [(1, 5), (3, 4), (1, 1), (2, 9), (1, 2)]
t = [(1, 5), (1, 1), (3, 4), (2, 9), (1, 2)]
t = [(1, 5), (1, 1), (2, 9), (3, 4), (1, 2)]
t = [(1, 5), (1, 1), (1, 2), (2, 9), (3, 4)]
```

Code 1 – Exemple d'exécution du tri insertion

Dans le tri fusion, les éléments sont triés en se déplaçant également de proche en proche. L'ordre relatif est encore préservé.

```
1  def tri_selection(tab: list) -> None:
2      for i in range(len(tab)):
3          i_mini = i
4          for j in range(i+1, len(tab)):
5              if tab[j] < tab[i_mini]:
6                  i_mini = j
7          tab[i], tab[i_mini] = tab[i_mini], tab[i]
```

Code 2 – tri par sélection

Dans un tableau de taille n, le tri par sélection échange l'élément de rang j par le plus petit dans l'intervalle $]j; n[$. Certains éléments peuvent ainsi *sauter* par-dessus certains éléments égaux.

```
1 t = [(1, 5), (2, 4), (1, 1), (2, 9), (1, 2)]
2 t = [(1, 5), (1, 1), (2, 4), (2, 9), (1, 2)]
3 t = [(1, 5), (1, 1), (1, 2), (2, 9), (2, 4)]
```

Code 3 – Exemple de la non stabilité du tri par sélection

Sommaire

Exercice 3

```python
def dichotomie_imp(tab: int, x: int) -> int:
    debut, fin = 0, len(tab) - 1
    while debut <= fin:
        milieu = (debut + fin)//2
        if tab[milieu] == x:
            return milieu
        elif tab[milieu] < x:
            debut = milieu+1
        else:
            fin = milieu-1
    return -1
```

Code 4 – Version impérative

```
1  def dichotomie_rec(tab: list, x: int, debut:
       int, fin: int) -> int:
2      if debut <= fin:
3          milieu = (debut + fin)//2
4          if tab[milieu] == x:
5              return milieu
6          elif tab[milieu] < x:
7              return dichotomie_rec(tab, x,
       milieu + 1, fin)
8          else:
9              return dichotomie_rec(tab, x,
       debut, milieu - 1)
10     else:
11         return -1
```

Code 5 – Version récursive

Sommaire

Exercice 4

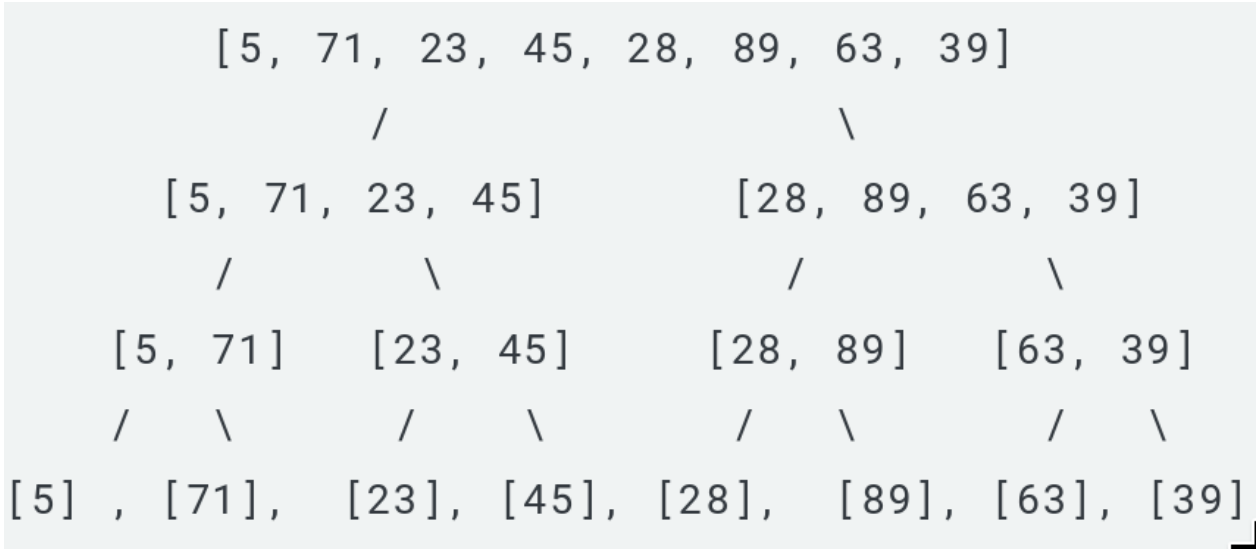

FIGURE 1 – Séparation

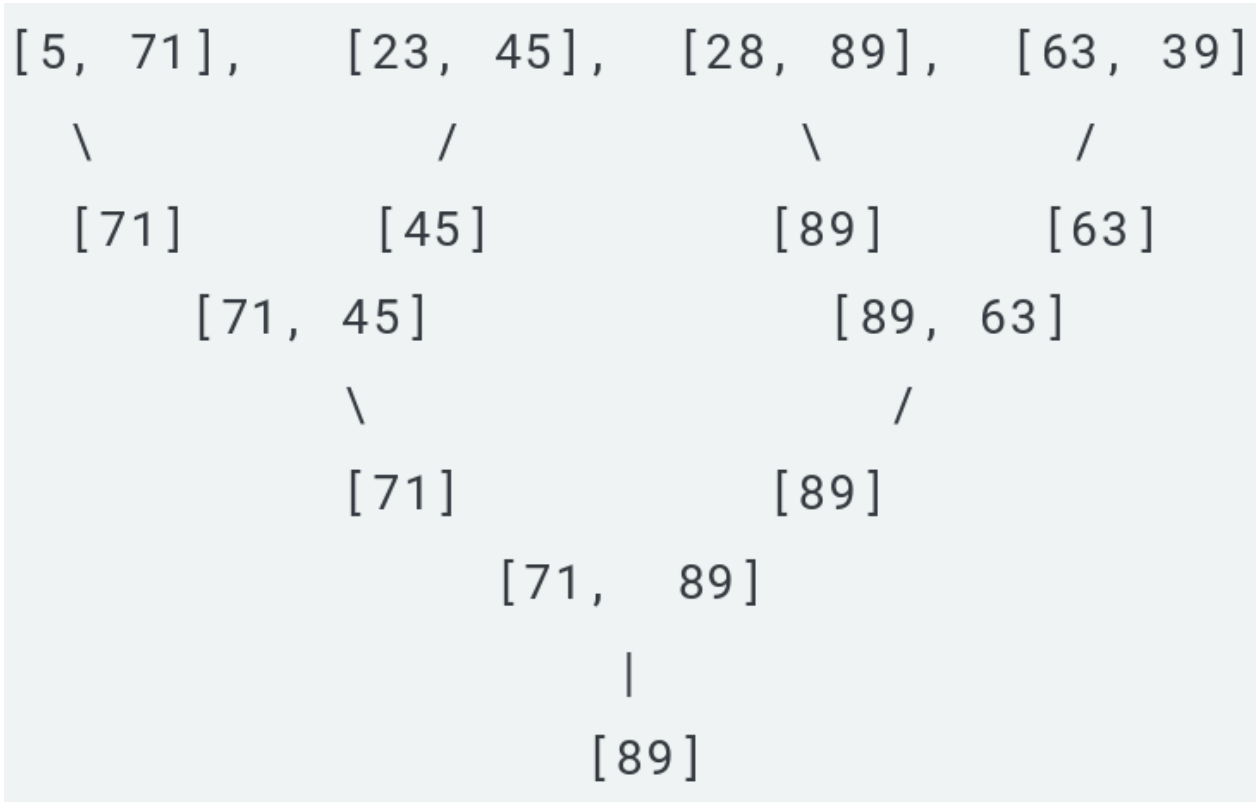

FIGURE 2 – Recombinaison

Cette fonction renvoie le maximum de la liste.

Complexité :

- À chaque appel récursif, la taille du tableau est divisé par 2. Comme pour le tri fusion, la complexité des appels est de l'ordre de $\log_2(n)$.
- À chaque remontée d'appel, on compare deux éléments. Le nombre de comparaisons dépend du niveau. Le total des comparaisons est de l'ordre de n.

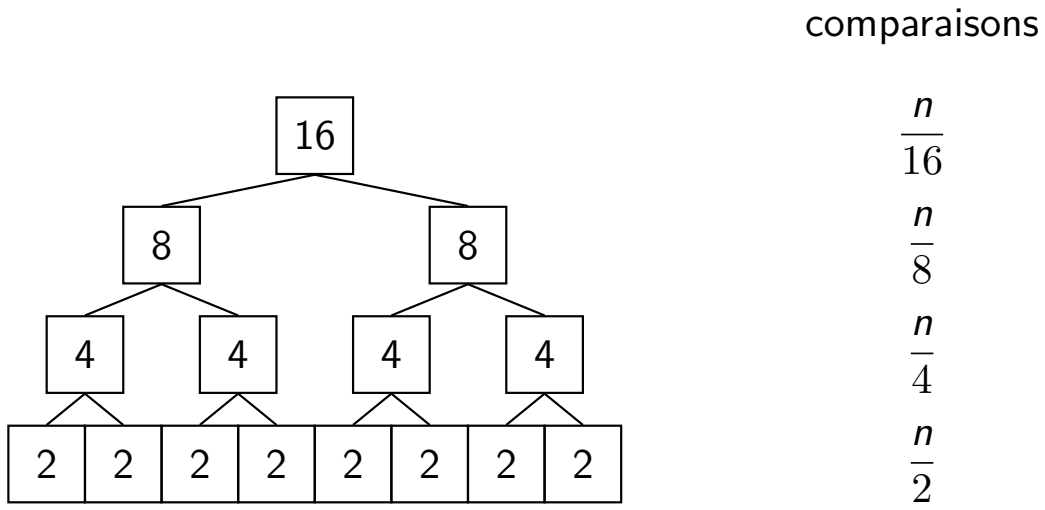

FIGURE 3 – Exemple avec un tableau de 16 éléments

La complexité est quasi-linéaire $n \times \log(n)$.

Sommaire

Exercice 5

```python
def tri_rapide(tab: list) -> list:
    if not tab:
        return []
    else:
        pivot = tab[0]
        petit = [x for x in tab if x < pivot]
        grand = [x for x in tab[1:] if x >= pivot]
        return tri_rapide(petit) + [pivot] +
   tri_rapide(grand)
```

Exercice 5

```
1  def partitionner(tab: list, deb: int, fin: int)
       -> int:
2      pivot = tab[deb]
3      pos = deb
4      for i in range(deb+1, fin):
5          if tab[i] < pivot:
6              pos += 1
7              tab[i], tab[pos] = tab[pos], tab[i]
8      # place le pivot
9      tab[deb], tab[pos] = tab[pos], tab[deb]
10     return pos
```

Exercice 5

```
1  def tri_rapide(tab: list, deb: int, fin: int) ->
       None:
2      """
3      Args:
4          tab (list): tableau d'entiers
5          deb (int): indice de début (inclus)
6          fin (int): indice de fin (exclus)
7      """
8      if deb < fin:
9          pivot = partitionner(tab, deb, fin)
10         tri_rapide(tab, deb, pivot)
11         tri_rapide(tab, pivot+1, fin)
```