	**Algorithmique**	**NSI T^{ale}**
	Diviser pour régner Exercices	Exos

*

1. Notation

Étant donné un tableau **T** et deux entiers naturels **a** et **b** inférieurs à **len(T)**, on note

- **T[a:b[** le sous-tableau **[T[a], T[a+1], …, T[b-1]]** si **a < b**
- le sous-tableau vide **[]** dans le cas contraire.

En Python, le tableau **T[a:b[** est noté **T[a:b]**

2. Ex1 Recherche du couple (minimum, maximum) dans un tableau

L'objectif est d'écrire une fonction récursive **def min_et_max(T, a, b)** renvoyant le couple (minimum, maximum) du tableau **T[a:b[**.
Un algorithme du type « diviser pour régner » permet de déterminer ce couple en exploitant le fait qu'une seule comparaison suffit pour obtenir à la fois le minimum et le maximum d'un tableau de taille 2.
Cet algorithme est le suivant :

```
si T[a:b[ ne contient qu'un élément :
    Renvoyer (T[a], T[a])
sinon si T[a:b[ contient deux éléments :
    Renvoyer (T[a], T[b]) ou (T[b], T[a]) dans l'ordre (minimum, maximum)
sinon :
    On pose c = (a + b)//2.
    On calcule récursivement les couples (minimum, maximum) dans T[a:c[ et dans T[c:b[.
    On compare les résultats obtenus et on renvoie le couple (minimum, maximum) sur T[a:b[
```

A) Écrire la fonction **def min_et_max(T, a, b)**.

B) Complexité

a) Combien de comparaisons faut-il effectuer pour rechercher la maximum et le minimum dans une liste de 10 nombres de manière classique ?

b) Combien de comparaisons faut-il effectuer pour rechercher la maximum et le minimum dans une liste de 10 nombres avec l'algorithme diviser pour régner ?

c) Et avec un tableau de 20 nombres ? 40 nombres ?

3. Exercice 2

3.1. Partie A

On considère la fonction Python suivante, dans laquelle **tab** est un tableau contenant des nombres positifs et des nombres négatifs.

```
def tranche(tab):
    n = len(tab)
    smax = tab[0]
    imax = jmax = 0
    for i in range(n):
        s = 0
        for j in range(i, n):
            s += tab[j]
            if s > smax:
                smax = s
                imax = i
                jmax = j
    return copie_partielle(tab, imax, jmax) # renvoie une copie du
    sous-tableau tab[imax:jmax+1[
```

A) Écrire la fonction **def copie_partielle(tab, imax, jmax)** : qui renvoie une copie du sous-tableau **tab[imax:jmax+1[**.

B) Afin de documenter la fonction tranche, indiquer son rôle en une ou deux lignes.

C) Que renvoie l'instruction tranche(**[3, -4, 2, -1, 5, -3]**) ?

D) Quelle est la complexité de la fonction tranche ?

E) Proposer une modification à la fonction tranche pour qu'entre plusieurs résultats possibles, elle renvoie celui de plus petite taille.

3.2. Partie B

A) Proposer un algorithme du type diviser pour régner résolvant le même problème que la fonction tranche.

B) Récupérer les fichiers **tranche.py** et **tabletest.py**.

C) Écrire la fonction **def tranche_dpr_(tab, a, b)** permettant d'implémenter cet algorithme.

D) Exécuter la fonction **test_tranche_dpr()** pour tester votre script.

Source : http://tnsi.free.fr/