

Asie– 2024 – sujet2 - Correction

Exercice 1 (6 points)

1. Effectifs possibles dans une liste de taille 10

Un élément est absolument majoritaire s'il apparaît dans strictement plus de la moitié des positions.

Moitié de 10 = 5

Il faut donc strictement plus de 5.

Les effectifs possibles sont :

6, 7, 8, 9 ou 10

👉 *Commentaire : « Strictement plus de la moitié » signifie $> n/2$ et non $\geq n/2$.*

Partie A – Sans dictionnaire

2. Fonction effectif

```
def effectif(val, lst):  
    compteur = 0  
    for elt in lst:  
        if elt == val:  
            compteur += 1  
    return compteur
```

👉 *Commentaire : On parcourt toute la liste et on compare chaque élément à val.*

3. Nombre de comparaisons pour effectif(1, [1, 4, 1, 6, 1, 7, 2, 1, 1])

La liste contient 9 éléments. Il y a donc 9 comparaisons.

👉 *Commentaire : Une comparaison par élément, donc n comparaisons.*

4. Fonction majo_abs1

```
def majo_abs1(lst):  
    n = len(lst)  
    for elt in lst:  
        if effectif(elt, lst) > n // 2:  
            return elt  
    return None
```

👉 *Commentaire : On teste chaque élément comme candidat potentiel.*

5. Nombre de comparaisons pour majo_abs1

Dans le pire cas :

- 9 appels à effectif
- Chaque effectif fait 9 comparaisons

Donc $9 \times 9 = 81$ comparaisons.

👉 *Commentaire : Complexité quadratique $O(n^2)$.*

Partie B – Avec dictionnaire

6. Fonction eff_dico

```
def eff_dico(lst):  
    dico_sortie = {}  
    for elt in lst:  
        if elt in dico_sortie:  
            dico_sortie[elt] += 1  
        else:  
            dico_sortie[elt] = 1  
    return dico_sortie
```

👉 *Commentaire : On construit progressivement le dictionnaire des effectifs.*

7. Fonction majo_abs2

```
def majo_abs2(lst):  
    dico = eff_dico(lst)  
    n = len(lst)  
    for cle in dico:  
        if dico[cle] > n // 2:  
            return cle  
    return None
```

👉 *Commentaire : On parcourt seulement les clés du dictionnaire. Complexité linéaire $O(n)$.*

Partie C – Diviser pour régner

8. Cas de base $n = 1$

Si la liste contient un seul élément, cet élément est absolument majoritaire.

👉 *Commentaire : Il apparaît 1 fois sur 1, donc strictement plus que $1/2$.*

9. Si ni lst1 ni lst2 n'admet d'élément majoritaire

Alors `lst` n'admet pas d'élément majoritaire.

👉 *Commentaire : Un élément majoritaire dans `lst` devrait être majoritaire dans au moins une des deux sous-listes.*

10. Vérifier si maj1 est majoritaire dans lst

Algorithme :

1. Calculer effectif (maj1, lst)
2. Vérifier si cet effectif > len(lst)//2
3. Si oui, retourner maj1
4. Sinon, retourner None

👉 *Commentaire : On vérifie globalement le candidat trouvé localement.*

11. Fonction récursive majo_abs3

```
def majo_abs3(lst):  
    n = len(lst)  
  
    if n == 1:  
        return lst[0]  
  
    lst1 = lst[:n//2]  
    lst2 = lst[n//2:]  
  
    maj1 = majo_abs3(lst1)  
    maj2 = majo_abs3(lst2)  
  
    if maj1 is not None and effectif(maj1, lst) > n // 2:  
        return maj1  
  
    if maj2 is not None and effectif(maj2, lst) > n // 2:  
        return maj2  
  
    return None
```

👉 *Commentaire : On combine les résultats des deux sous-problèmes.*
Complexité : $O(n \log n)$.

Bilan des trois méthodes

1. Méthode naïve : $O(n^2)$
 2. Dictionnaire : $O(n)$
 3. Diviser pour régner : $O(n \log n)$
-