



EA4 – Éléments d’algorithmique
Contrôle du 16 mars 2023 – Sujet A
Durée : 1 heure 15

*Aucun document autorisé
Appareils électroniques éteints et rangés*

Nom :

Prénom :

Numéro :

Groupe :

Exercice 1 :

On considère l’algorithme `foo` ci-contre.

Dessiner ci-dessous l’arbre des appels récurrents provoqués par l’appel `foo([4, 3, 2, 1], 0, 4)` : pour chaque appel, indiquer le contenu de `T` au début *et* à la fin de l’appel (sauf s’il est inchangé), en encadrant le sous-tableau concerné. En cas d’appels équivalents, vous pouvez ne dérouler que le premier.

```
def foo(T, deb, fin) :  
    if fin-deb <=1 : return  
    m = (deb+fin)//2  
    foo(T, deb, m)  
    foo(T, m, fin)  
    if T[m-1] > T[fin-1] :  
        T[m-1], T[fin-1] = T[fin-1], T[m-1]  
    foo(T, deb, fin-1)
```

Que fait `foo` ? Justifier.

Soit $C(n)$ le nombre de comparaisons effectuées par `foo(T)` si `T` est un tableau de longueur n . Quelle relation de récurrence $C(n)$ satisfait-elle ?

Cette relation implique que, $\forall n \geq 2$, $C(n) \geq \sum_{i=0}^{\lfloor n/2 \rfloor} C(i)$. *(inégalité très large, mais suffisante)*

En déduire par récurrence sur k la propriété suivante :

(hors-barème) « pour tout entier $k \geq 0$, il existe $\alpha_k > 0$ tel que $\forall n \geq 2$, $C(n) \geq \alpha_k n^k$. »

Que pensez-vous de l'efficacité de `foo` ? _____

Exercice 2 :

Soit T un tableau de n éléments comparables, par exemple des entiers positifs. On dit que T possède un *minimum local en position i* si $T[i] \leq T[i-1]$ et $T[i] \leq T[i+1]$ (donc en particulier, il faut que $n \geq 3$ et $0 < i < n-1$).

Entourer les 5 minima locaux du tableau $T = \boxed{9} \boxed{7} \boxed{7} \boxed{2} \boxed{1} \boxed{3} \boxed{7} \boxed{5} \boxed{4} \boxed{7} \boxed{3} \boxed{3} \boxed{6}$.

Soit $n \geq 3$. Tout tableau T de longueur n possède-t-il un minimum local ? Justifier.

On suppose dorénavant que T satisfait la propriété : $T[0] \geq T[1]$ et $T[n-2] \leq T[n-1]$.
Justifier que sous cette hypothèse, T possède au moins un minimum local.

Décrire un algorithme *le plus efficace possible* dans le pire cas pour déterminer un minimum local d'un tel tableau.

Quelle est sa complexité ? _____

Exercice 3 :

Dans cet exercice, on représente des ensembles par des tableaux *sans doublon*, et on considère la fonction foo suivante :

```
def foo(E, F) :  
    res = E[:]  
    for f in F :  
        if f not in E : res.append(f)  
    return res
```

Que calcule foo(E, F) ?

Quelle est sa complexité dans le pire cas si E et F sont supposés de même longueur n ? Justifier.

Supposons maintenant que les ensembles sont représentés par des tableaux *triés* (et toujours sans doublon). Proposer un algorithme le plus efficace possible pour effectuer le même calcul.

Quelle est sa complexité? _____

Exercice 4 :

On s'intéresse au problème suivant : étant donné une liste L de nombres (non nécessairement entiers) de longueur n , déterminer le *vainqueur* de L , *i.e.* l'élément de L qui y apparaît le plus de fois (ou l'un quelconque d'entre eux, en cas d'égalité).

Décrire un algorithme naïf permettant de résoudre ce problème sans modifier la liste L , et avec mémoire auxiliaire constante.

Quel est la complexité (en temps) de cet algorithme? Justifier.

Comment résoudre ce problème avec une complexité (dans le pire cas) strictement meilleure?

Quelle est la complexité de cette méthode? _____