DS Session 2 - Algorithmique 1 - Master 1 Bioinformatique

11 juin 2015 - Durée : 1h30 minutes

Les documents, les ordinateurs et les téléphones ne sont pas autorisés. Les modules python ne sont pas autorisés. Vous pouvez utiliser les listes mais vous ne pouvez pas utiliser les fonctions avancées des listes, comme par exemple, `l.sort()`, `l.reverse()`, `l.index(e)`.

Exercice 1: 2 points

Soit $(U_n)_{n\geq 0}$ la suite définie par :

$$\begin{cases} U_{n+1} = U_n - 2.U_{n-1} + 4.U_{n-2}, \\ U_0 = 0, \\ U_1 = 1, \\ U_2 = 2. \end{cases}$$

Proposer une fonction `suite(n)` qui renvoie la valeur de U_n.

Exercice 2: 2 points

Proposer une fonction `palindrome(l)` qui prends en paramètre une liste l d'entiers et qui renvoie vraie si l est un palindrome et faux sinon.

Une liste est un palindrome si elle ne change pas lorsqu'on la renverse.

Par exemple, $[1, 4, 3, 4, 1]$ est un palindrome et $[1, 4, 3, 4, 2]$ n'est pas un palindrome.

Exercice 3: 4 points

Proposer un algorithme `est_permutation(l)` qui prends en paramètre un liste `l` d'entiers et qui renvoie vrai si la liste contient tous les entiers de 0 à $n-1$ où n est la taille de la liste.

Par exemple, `est_permutation([5,2,3,0,4])` renvoie faux, alors que `[2,1,3,0,4]` renvoie vrai.

La complexité de cet algorithme devra être linéaire en la taille de la liste, c'est à dire en $O(n)$. Vous justifierez la complexité de votre algorithme.

Exercice 4: 4 points

Proposez une fonction `rechercher_element(l,e)` qui prends en paramètre une liste l triée et un élément e et qui renvoie vraie si e est dans l et faux sinon.

Vous proposerez un algorithme optimal (dont la complexité est la meilleure possible). Donnez, sans justifier, sa complexité.

Exercice 5: 2 points

Quelle est la différence entre une file et une pile ?

Exercice 6: 6 points

On suppose que l'on a accès à une implémentation des piles dont l'API est la suivante :

```
def creer_pile():
    # Renvoie une nouvelle pile vide.
def empiler( P, e ):
    # Empile l'élément 'e' dans la pile 'P'.
def depiler( P ):
    # Dépile un élément de la pile 'P' et renvoie cet élément.
def taille_pile( P ):
        # Renvoie le nombre d'éléments contenu dans la pile
```

Proposer un algorithme `trier(p_entree,p_sortie)`, qui prends en paramètre 2 piles, une pile `p_entree` contenant des entiers dans le désordre et une pile `p_sortie` vide et qui met dans `p_sortie` les entiers de `p_sortie` dans l'ordre croissant (l'élément le plus petit est situé au fond de la pile). A la fin de l'algorithme, la pile `p_entree` doit être vide. Cet algorithme devra utiliser uniquement la pile `p_entree`, la pile `p_sortie`, les comparaisons, les boucles et les variables d'entiers.