Algorithmique des tableaux (2)

Compétences

- Utiliser l'indexation pour accéder aux éléments d'un tableau.
- Analyser un problème avec plusieurs algorithmes possibles.

Exercices

* **Ex. 1 —** Écrire une fonction python `rechercher(t, n, val)` qui prend en paramètres un tableau de valeurs entières, la taille du tableau, la valeur à chercher dans le tableau et qui renvoie True si la valeur est présente et False sinon. Évaluer le nombre de comparaisons effectuées pour trouver cette valeur dans le pire cas et le meilleur cas ?

** **Ex. 2 —** Le jeu du nombre caché : Laura a choisi un nombre entier compris entre 1 et 100, Clara doit le deviner. Clara fait des propositions et Laura répond *trop grand*, *trop petit* ou *gagné*. Le jeu s'arrête lorsque Clara a trouvé le nombre.

1. Quelle(s) différence(s) faites-vous avec la recherche de l'exercice précédent ?
2. Faire une partie et consigner sur papier toutes étapes dans un tableau.
3. Que faut-il mémoriser ?
4. Que faut-il archiver dans le tableau ?
5. Quelle technique de jeu peut employer Clara pour gagner le plus rapidement possible ?
6. Dans le cas général, évaluer le nombre de comparaisons (dans le pire des cas) à effectuer lorsque Clara utilise la méthode optimisée appelée dichotomie, en fonction de la taille N du tableau.

** **Ex. 3 —** On remplace maintenant Clara par un ordinateur. À partir des observations précédentes écrire une fonction python `dichotomie(t, n, val)` qui prend en paramètes un tableau de valeurs entières, la taille du tableau, la valeur `val` à trouver et qui renvoie l'indice de position de `val` si elle est présente et -1 sinon.

1. tester votre algorithme avec le tableau `[0,1,2,3,4,5,6,7,8,9]` et les valeurs à chercher suivantes : $4, 1, -1, 9, 0$
2. Dans le cas où la valeur cherchée n'est pas présente, comment évolue la taille du tableau dans lequel se fait la recherche lorsque la taille du tableau de départ est $N = 2^k - 1$?

*** **Ex. 4 — Exercice complémentaire sur la dichotomie ne faisant pas appel à la notion de tableau**
Pour résoudre numériquement une équation non linéaire de la forme $f(x) = 0$, il convient de :

- localiser grossièrement le ou les zéros de f, on notera x_0 cette solution grossière
- construire à partir de x_0 une suite telle que $\lim_{n \longrightarrow \infty} x_n = x_r$ où $f(x_r) = 0$ (x_r est un zéro de f)

Sur le principe de la dichotomie implémenter une fonction `dichotomie_f` dont le prototype est : `dichotomie_f(f, a, b, epsilon)` qui prend en paramètres une fonction mathématique f un intervalle d'étude ($a, b \in \mathbb{R}^2$). Que peut on dire de `epsilon` ?

** **Ex. 5 —** Écrire une fonction `rotationAdroite(t, n)` qui applique aux `n` éléments d'un tableau `t` un décalage d'une position à droite. L'élément en position `n-1` est placé en position 0 après l'appel de la fonction.
Exemple :
si `t` a 10 cases et 7 éléments : `t = [0,1,2,3,4,5,6,None,None,None]`, après l'appel de la fonction `rotationAdroite(t,7)` on aura `t = [6,0,1,2,3,4,5,None,None,None]`. Le tableau ne sera modifié que s'il contient au moins deux éléments.

** **Ex. 6 —** On veut maintenant écrire un algorithme qui applique aux éléments d'un tableau une rotation à droite de `k` positions.
Exemple :
si `t = [0,1,2,3,4,5,6,None,None,None]` et qu'on demande une rotation à droite de 3 positions, après l'appel de la fonction on aura `t = [4,5,6,0,1,2,3,None,None,None]`. Deux algorithmes sont possibles : 1) le premier consiste à réutiliser la fonction rotationAdroite, 2) le deuxième utilise un tableau auxiliaire de taille égale au nombre d'éléments dans t.

1. Analysez les deux méthodes. Comparer le nombre de recopies d'éléments du tableau effectuées par chaque méthode.
2. Comment éviter d'effectuer des recopies inutiles si $k > n$?
3. Écrire et tester les deux versions de la fonction `rotationADroiteDeK_V1(t, n, k)` et `rotationADroiteDeK_V2(t, n, k)` (respectivement avec la première méthode et avec la seconde méthode).
   Remarque : si `t` contient `n` éléments avec `n < len(t)`, les cases de `t` d'indice supérieur à `n-1` ne seront jamais utilisées par les algorithmes demandés.

*** **Ex. 7 —** Écrire une troisième version `rotationADroiteDeK_V3(t, n, k)` permettant toujours d'effectuer une rotation de `t` de `k` positions vers la droite, en temps linéaire et sans avoir recours à un tableau auxiliaire.