Aix*Marseille université

Domaine Sciences et Technologies
Portail 1 René Descartes

Science informatique : TD 1

Année 2022-23

Algorithmes et tableaux

Exercice 1 L'élévation d'un nombre à une puissance entière positive est une opération de base cruciale lorsqu'on veut calculer avec des valeurs numériques. C'est même l'étape de base dans la méthode cryptographique RSA. Étant donné un nombre x (cela peut-être un entier ou un réel, qu'on représente en machine à l'aide d'un flottant) et un entier naturel n, on souhaite donc calculer le nombre $x^n = \underbrace{x \times x \times \cdots \times x}_{n \text{ fois}}$.

1. Écrire une fonction prenant en arguments x et n et qui renvoie x^n.
2. Exécuter cet algorithme lors du calcul de 2^{10}, en précisant la valeur intermédiaire des variables. Combien de multiplications a-t-on effectué au total pour calculer 2^{10} ?
3. Dans le cas général, combien de multiplications effectue cet algorithme pour calculer x^n, en fonction de x et n ?

   On peut réaliser l'élévation à la puissance de manière plus rapide, en distinguant les puissances paires et impaires :

```
def puissance_rapide(x, n):
    a = 1
    b = x
    m = n
    while m > 0:
        if m % 2 == 1:
            a = a * b
        m = m // 2
        b = b * b
    return a
```

4. Exécuter cet algorithme lors du calcul de 2^{10}, en précisant la valeur des variables lors de chaque étape de la boucle `while`.

x	n	a	b	m

5. Les opérations élémentaires coûteuses d'un algorithme d'exponentiation (c'est le nom qu'on donne à l'*élévation à la puissance*) sont les multiplications. Combien de multiplications sont effectuées par l'algorithme d'exponentiation rapide lors du calcul de 2^{10}. Plus généralement, pouvez-vous donner une approximation du nombre de multiplications effectuées pour calculer x^n par l'algorithme, en fonction de n ?
6. **(À la maison)** Répondre à nouveau à la question 1 en utilisant une fonction récursive (et donc pas de boucle).

Exercice 2 Un palindrome est un mot qui peut se lire à l'endroit ou à l'envers. Par exemple, le mot « kayak » est un palindrome. Voici un code testant si une chaîne de caractères est un palindrome :

```
def palindrome(t):
    n = len(t)
    for i in range(n//2):
        if t[i] != t[n-1-i]:
            return False
    return True
```

Exécuter ce code sur les chaînes « serres », « resister » et « kayak ».

Exercice 3 Un des algorithmes les plus élémentaires sur un tableau consiste à rechercher un élément dedans, à l'aide, par exemple, de la recherche séquentielle :

```
def rechercher_séquentiel(t, e):
    n = len(t)
    for i in range(n):
        if t[i] == e:
            return i
    return -1
```

1. Modifier l'algorithme de recherche séquentielle pour qu'il renvoie vrai ou faux, selon que l'élément a été trouvé dans le tableau ou non : ainsi, la recherche de l'élément 8 dans le tableau $[3, 5, 8, 2, 8, 1]$ devra renvoyer `True`, alors que la recherche de 9 dans ce même tableau devra renvoyer `False`.
2. Modifier ensuite l'algorithme pour qu'il renvoie l'indice de la *dernière occurrence* de l'élément recherché : ainsi, la recherche de l'élément 8 dans le tableau $[3, 5, 8, 2, 8, 1]$ renverra désormais 4.
3. Améliorer l'algorithme de recherche séquentielle dans le cas où le tableau donné en entrée est supposé trié, pour qu'il s'arrête dans son parcours du tableau dès lors qu'il a trouvé l'élément à chercher, ou bien qu'il est sûr que l'élément à chercher ne se trouve pas dans le tableau.
4. Quelle est l'ordre de grandeur de complexité dans le pire des cas de votre nouvel algorithme ? Comparer avec l'algorithme de recherche séquentielle non modifié.

Exercice 4

1. Écrire une fonction Python qui calcule la valeur moyenne d'un tableau non vide de valeurs numériques ;
2. Écrire une fonction Python qui calcule la valeur minimale apparaissant dans un tableau.
3. **(À la maison)** Une matrice est un tableau bidimensionnel, qu'on représente en Python comme un tableau de tableaux. Par exemple, le tableau de tableaux `t = [[1, 2], [3, 4], [5, 6]]` représente la matrice
$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$
On peut accéder à l'élément situé sur la ligne d'indice i (numérotée à partir de 0) et la colonne d'indice j grâce à `t[i][j]`. Par exemple, `t[0][1]` vaut 2. Écrire une fonction Python qui calcule la somme des coefficients de la matrice. Pour l'exemple précédent, on renverra donc $1 + 2 + 3 + 4 + 5 + 6 = 21$.
4. **(À la maison)** Écrire une fonction Python qui calcule la *transposée* d'une matrice donnée en argument. La transposée d'une matrice est la matrice obtenue par une symétrie vis-à-vis de la diagonale partant du coin en haut à gauche. La transposée de la matrice donnée en exemple dans la question précédente est la matrice
$$\begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$
Tester votre code en Python !