Nicolas Méloni, Université de Toulon

I21 - Algorithmique et Programmation - Exercices d'Algorithmiques

L1 Infomatique, 2nd semestre 2022-2023

Table des matières

Chapitre 1

Conception des algorithmes

Exercice 1 Logique élémentaire

Donner un contre exemple pour chacune des affirmations suivantes :

1. $\forall a, b \in \mathbb{R}, a + b \geqslant \max(a, b)$.
2. $\forall a > 0, b > 0, a \times b \geqslant \min(a, b)$.
3. Si la somme des chiffres d'un entier n est multiple de 6 alors n est multiple de 6.
4. Si p est impair et p est premier alors $p + 2$ ou $p + 4$ est premier.
5. $n^2 - n + 41$ est premier pour tout $n \geqslant 0$.
6. L'équation $ax^2 + bx + c = 0$ a toujours 2 solutions (dans $\mathbb{R}$ ou $\mathbb{C}$).
7. L'équation $ax^2 + bx + c = 0$ a au moins une solution (dans $\mathbb{R}$ ou $\mathbb{C}$).

Exercice 2

Pour chaque algorithme dire quelles propriétés vérifient les variables si le dernier bloc d'instruction est exécuté.

```
1  ALGO 1
2     SI x > 0 ALORS
3        BLOC 1
4     SINON SI x < −1 ALORS
5        BLOC 2
6     SINON
7        BLOC 3
8     FSI
```

```
1  ALGO 2
2     SI x > 0 OU y > 0 ALORS
3        BLOC 1
4     SINON SI x < −1 ALORS
5        BLOC 2
6     SINON
7        BLOC 3
8     FSI
```

```
1  ALGO 3
2     SI xy = 0 ALORS
3        BLOC 1
4     SINON SI x = 0 ALORS
5        BLOC 2
6     SINON
7        BLOC 3
8     FSI
```

```
1  ALGO 4
2     SI x > 0 ET y = 0 ALORS
3        BLOC 1
4     SINON SI x = 0 ET y > 0 ALORS
5        BLOC 2
6     SINON SI x = 0 ALORS
7        BLOC 3
8     SINON
9        BLOC 4
10    FSI
```

```
1  ALGO 5
2    SI x > 0 ET y > 0 ALORS
3      BLOC 1
4    SINON SI |x−y| < 1 ALORS
5      BLOC 2
6    SINON SI xy ⩽ 0 ALORS
7      BLOC 3
8    SINON
9      BLOC 4
10   FSI
```

```
1  ALGO 6
2    SI x < z ET y < z ALORS
3      BLOC 1
4    SINON SI |x−z| < |y−z| ALORS
5      BLOC 2
6    SINON SI |x−y| > |x−z|  ALORS
7      BLOC 3
8    SINON
9      BLOC 4
10   FSI
```

Exercice 3

Prouver les résultats suivants :

1. $\forall n \geqslant 0, \sum_{i=1}^{n}(2i+1) = n^2 + 2n$.
2. $\forall n \geqslant 0, \sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$..
3. $\forall n \geqslant 0, \sum_{i=1}^{n} i^3 = n^2(n+1)^2/4$.
4. $\forall n \geqslant 0, \sum_{i=1}^{n} i(i+1) = n(n+1)(n+2)/3$.
5. $\forall n \geqslant 0, \sum_{i=1}^{n} i(i+1)(i+2) = n(n+1)(n+2)(n+3)/4$.
6. $\forall n \geqslant 0, \sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$.
7. $\forall n \geqslant 0, \sum_{i=1}^{n}(-1)^{i-1}i^2 = (-1)^{n-1}n(n+1)/2$.
8. $\forall n \geqslant 0, \sum_{i=1}^{n} ii! = (n+1)! - 1$.
9. $\forall n \geqslant 2, 6^n$ est un multiple de 9.
10. $\forall n \geqslant 0, n^3 - n$ est un multiple de 3.
11. $\forall n \geqslant 0, 2^{4n+2} + 3^{n+2}$ est un multiple de 13.
12. $\forall n \geqslant 1, n$ impair, $n^2 - 1$ est un multiple de 8.
13. $\forall n \geqslant 1, 4^n + 15n - 1$ est un multiple de 9.
14. $\forall n \geqslant 10, n^3 \leqslant 2^n$.
15. $\forall n \geqslant 4, n! > 2^n$.

Exercice 4

Donner le terme général des suites définies par récurrence suivantes.

1. $u_0 = 3, u_{n+1} = u_n + 2$
2. $u_0 = 0, u_{n+1} = u_n + n - 1$
3. $u_0 = 1, u_{n+1} = u_n + 2^n$
4. $u_0 = 1, u_{n+1} = 2u_n - 1$
5. $u_0 = 1, u_{n+1} = 2u_n + 1$

Exercice 5

Pour chaque problème proposer une instance qui invalide l'algorithme sensé le résoudre.

Problème : Double échange

1. *Entrée* : Tableau d'entiers de taille $n \geqslant 1$ et 4 entiers $i_1 \dots i_4$
   Sortie : Une permutation des éléments d'indices i_1, i_2 et i_3, i_4

```
1  ALGORITHME EchangeDouble(T, i1 , i2 , i3 , i4 ):
2  DEBUT
3     temp = T[i1]
4     T[i1] = T[i2]
5     T[i2] = temp
6     temp  = T[i3]
7     T[i3] = T[i4]
8     T[i4] = temp
9  FIN
```

2. *Problème* : Plus grand entier
   Entrée : Tableau d'entiers de taille $n \geqslant 1$
   Sortie : Le plus grand entier du tableau

```
1  ALGORITHME Max(T): entier
2  DEBUT
3     max←0
4     i←1
5     TQ i ⩽ n FAIRE
6        SI T[i]>max ALORS
7           max←T[i]
8        FSI
9        i←i+1
10    FTQ
11    RENVOYER max
12 FIN
```

3. *Problème* : Deux plus grands entiers
   Entrée : Tableau d'entiers de taille $n \geqslant 2$
   Sortie : Les deux plus grands entiers du tableau

```
1  ALGORITHME DeuxMax(T): tableau d entiers
2  DEBUT
3     i←1
4     max1←T[1]
5     max2←T[1]
6     TQ i ⩽ n FAIRE
7        SI T[i]>max1 ALORS
8           max2←max1
9           max1←T[i]
10       SINON SI T[i]>max2 FAIRE
11          max2←T[i]
12       FSI
13       i←i+1
14    FTQ
15    RENVOYER [max1,max2]
16 FIN
```

4. *Problème* : Deux plus grandes valeurs
   Entrée : Tableau d'entiers de taille $n \geqslant 2$ contenant au moins 2 valeurs différentes
   Sortie : Les deux plus grandes valeurs (différentes) du tableau

```
1  ALGORITHME DeuxMaxVal(T): tableau d entiers
2  DEBUT
3     i←3
4     SI T[1] ⩾ T[2] ALORS
5        max1←T[1]
6        max2←T[2]
7     SINON
8        max1←T[1]
9        max2←T[2]
10    FSI
11    TQ i ⩽ n FAIRE
12       SI T[i]>max1 ALORS
13          max2←max1
14          max1←T[i]
15       SINON SI T[i]>max2 FAIRE
16          max2←T[i]
17       FSI
18       i←i+1
19    FTQ
20    RENVOYER [max1,max2]
21 FIN
```

Problème : Somme des diviseurs propres

5. *Entrée* : Un entier $n \geqslant 0$

Sortie : La somme des diviseurs propres de n

```
1  ALGORITHME SomDivPropre(n): entier
2  DEBUT
3     d ← 2
4     s ← 1
5     TQ d < n FAIRE
6        SI n  mod  d = 0 ALORS
7           s ← s+d
8        FSI
9        d←d+1
10    FTQ
11    RENVOYER s
12 FIN
```

Problème : Sac à dos

6. *Entrée* : Un ensemble $S = \{s_1, \ldots, s_n\}$ de n entiers et un nombre cible T

Sortie : Un sous-ensemble de S dont la somme des éléments est la plus proche T par défaut.

```
1  ALGORITHME SacADos1(S,P):
2  DEBUT
3     sol←{}
4     i ← 1
5     TQ i ⩽ n FAIRE
6        SI S[i] <= P ALORS
7           P←P–S[i]
8           rajouter S[i] a sol
9        FSI
10       i←i+1
11    FTQ
12    RENVOYER sol
13 FIN
```

```
1  ALGORITHME SacADos2(S,P):
2  DEBUT
3     Trier S dans l ordre croissant
4     sol←{}
5     i ← 1
6     TQ i ⩽ n FAIRE
7        SI S[i] <= P ALORS
8           P←P–S[i]
9           rajouter S[i] a sol
10       FSI
11       i←i+1
12    FTQ
13    RENVOYER sol
14 FIN
```

```
1  ALGORITHME SacADos3(S,P):
2  DEBUT
3     Trier S dans l ordre decroissant
4     sol←{}
5     i ← 1
6     TQ i ⩽ n FAIRE
7        SI S[i] <= P ALORS
8           P = P-S[i]
9           rajouter S[i] a sol
10       FSI
11       i←i+1
12    FTQ
13    RENVOYER sol
14 FIN
```

Chapitre 2

Analyse asymptotique

Exercice 6

Quelle est la plus petite valeur de n telle qu'un algorithme dont le temps d'exécution est $256n^2$ s'exécute plus vite qu'un algorithme dont le temps d'exécution est 2^n sur la même machine ?

Exercice 7

Un algorithme met 1 seconde pour traiter un tableau de 1000 éléments sur votre ordinateur personnel. Quel temps lui faudra t-il pour traiter 100000 éléments si

1. son temps d'exécution est à peu près proportionnel à n ?
2. son temps d'exécution est à peu près proportionnel à $\sqrt{n}$?
3. son temps d'exécution est à peu près proportionnel à n^2 ?
4. son temps d'exécution est à peu près proportionnel $n\log(n)$?

Exercice 8

On suppose disposer d'une machine capable d'effectuer 2^{32} opérations par seconde. On considère des algorithmes dont les complexités asymptotiques sont les suivantes : $\Theta(\log_2(n))$, $\Theta(n)$, $\Theta(n^2)$ et $\Theta(2^n)$. Pour chaque complexité, donner la taille des problèmes que l'on peut résoudre en une seconde.

Exercice 9

Pour chaque paire de fonctions f et g, dire si $f = O(g)$, $f = \Omega(g)$ et $f = \Theta(g)$ (en justifiant).

a) $f(n) = n + 1;\ g(n) = 2n^2 + n$

b) $f(n) = 100n^2;\ g(n) = 0.01n^3$

c) $f(n) = \log(n^5);\ g(n) = \log(\sqrt{n})$

d) $f(n) = \log^{10}(n);\ g(n) = \sqrt{n}$

e) $f(n) = n^2 + n\log_2(n);\ g(n) = n^2$

f) $f(n) = 2^n;\ g(n) = 2^{n+1}$

g) $f(n) = 2^n;\ g(n) = 2^{2n}$

h) $f(n) = 2^n;\ g(n) = 2^{\frac{n}{2}}$

i) $f(n) = \frac{n}{\log(n)};\ g(n) = \log^2(n)$

j) $f(n) = \log_2(2^n);\ g(n) = \log_2(\sqrt{n^3})$

k) $f(n) = n\sqrt{n};\ g(n) = (\sqrt{n} + 1000\log(n))^2$

l) $f(n) = \log(2^n);\ g(n) = 2^{\log(n)}$

m) $f(n) = n!;\ g(n) = (n+1)!$

n) $f(n) = (n+2)!;\ g(n) = n^2(n-1)!$

o) $f(n) = 2n + (2n-1) + \cdots + (n+1);\ g(n) = n + (n-1) + \cdots + 1$

p) $f(n) = 1 + 2 + 3 + \cdots + (2n+1);\ g(n) = 1 + 2 + 3 + \cdots + n$

q) $f(n) = 2^n + 2^{n-1} + \cdots + 2 + 1;\ g(n) = 2^{n/2}$

Exercice 10

Soit f, g, h trois fonctions numériques strictement croissantes. Prouver que la propriété suivante est fausse.

$$f(g(x)) = O(f(h(x)) \Rightarrow g(x) = O(h(x)).$$

Exercice 11

Donner la complexité de chaque algorithme en fonction de n. On rappelle que le symbole mathématique | signifie *divise*.

```
1  BOUCLE 1
2    DEBUT
3      i ← 1
4      TQ i ⩽ n FAIRE
5        j ← 2
6        TQ j<√n FAIRE
7          j ← j+1
8        FTQ
9        i ← i+1
10     FTQ
11   FIN
```

```
1  BOUCLE 2
2    DEBUT
3      i,j ← 1,n
4      TQ i ⩽ j FAIRE
5        i ← i+1
6        j ← ⌊j/2⌋
7      FTQ
8    FIN
```

```
1  BOUCLE 3
2    DEBUT
3      i ← 1
4      j ← n
5      TQ  i×j ⩽ n² FAIRE
6        i ← i+1
7        j ← j+1
8      FTQ
9    FIN
```

```
1  BOUCLE 4
2
3    DEBUT
4      i ← n
5      TQ  i ⩾ 1 FAIRE
6        j ← 1
7        TQ j ⩽ i FAIRE
8          j ← j+1
9        FTQ
10       i ← ⌊i/3⌋
11     FTQ
12   FIN
```

```
1  BOUCLE 5
2    DEBUT
3      i ← 1
4      TQ i ⩽ n FAIRE
5        j ← 1
6        TQ j ⩽ n FAIRE
7          SI 2|j ALORS
8            k ← 1
9            TQ k ⩽ j FAIRE
10             k ← k+1
11           FTQ
12         FSI
13         j ← j+1
14       FTQ
15       i ← i+1
16     FTQ
17   FIN
```

```
1  BOUCLE 6
2    DEBUT
3      i ← 1
4      TQ i ⩽ n FAIRE
5        j ← 1
6        TQ j ⩽ i FAIRE
7          k ← 1
8          TQ k ⩽ 2^j
9            k ← k+1
10         FTQ
11         j ← j+1
12       FTQ
13       i ← i+1
14     FTQ
15   FIN
```

```
BOUCLE 7
  DEBUT
    i ← n
    TQ 1 < i FAIRE
      j ← 1
      TQ j < n FAIRE
        j ← j+1
      FTQ
      i ← ⌊ i/2 ⌋
    FTQ
  FIN
```

```
BOUCLE 8
  DEBUT
    i ← 1
    TQ i ⩽ n FAIRE
      j ← 1
      TQ j ⩽ n FAIRE
        k ← 1000
        TQ k ⩾ 1 FAIRE
          k ← ⌊k/2⌋
        FTQ
        j ← j+i
      FTQ
      i ← i+1
    FTQ
  FIN
```

```
BOUCLE 9 (Examen 2019)
  DEBUT
    i ← 1
    TQ i*i ⩽ n^3 FAIRE
      j ← 1
      TQ j ⩽ n
        j ← j+2
      FTQ
      i ← i+1
    FTQ
  FIN
```

```
BOUCLE 10 (Examen 2019)
  DEBUT
    i ← 1
    TQ i ⩽ n FAIRE
      j ← n
      TQ j ⩾ 1 FAIRE
        j ← ⌊ j/2 ⌋
      FTQ
      i ← i+1
    FTQ
  FIN
```

Exercice 12 Examen 2019

On considère l'algorithme suivant :

```
DEBUT
  i ← 1
  TQ i ⩽ ⌊n/2⌋ FAIRE
    j ← 1
    TQ j ⩽ i*i FAIRE
      j ← j+1
    FTQ
    i ← i+1
  FTQ
FIN
```

1. Soit la proposition $\mathcal{P}(n) : \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$. Montrer que $\mathcal{P}(n)$ est vraie pour tout $n \geqslant 0$.
2. En déduire une preuve de la complexité de l'algorithme.

Exercice 13 Examen 2020

1. Montrer par récurrence que

$$\forall n \geqslant 1, \sum_{i=1}^{n-1} i2^i = 2 - n2^n + (n-1)2^{n+1}.$$

*(**Indication :** $n = \frac{n-1}{2} + \frac{n+1}{2}$.)*

2. Faire une analyse de la complexité asymptotique de la boucle suivante :

```
1  DEBUT
2     i ← 1
3    TQ i ⩽ n FAIRE
4        j ← 1
5       TQ j⩽ 2^i FAIRE
6          k ← i
7          TQ k ⩾ 0 FAIRE
8             k ← k−1
9          FTQ
10         j ← j+1
11      FTQ
12      i ← i+1
13   FTQ
14 FIN
```

Exercice 14 Examen 2022

1. Montrer par récurrence que

$$\forall n \geqslant 0, \sum_{i=0}^{n-1} i3^i = \frac{3 - n3^n + (n-1)3^{n+1}}{4}$$

2. En déduire que

$$\sum_{i=0}^{n-1} i3^i = \Theta(n3^n)$$

3. Analyser la complexité de la boucle suivante.

```
1    BOUCLE 1
2      DEBUT
3         i ← 1
4         TQ i∗i ⩽ n FAIRE
5            j ← 1
6            TQ j ⩽ i
7               k ← 0
8               TQ k < 3^i FAIRE
9                  k ← k+1
10              FTQ
11              j ← j+1
12           FTQ
13           i ← i+1
14        FTQ
15     FIN
```

Chapitre 3

Bases de l'algorithmique

Exercice 15

Pour chaque question on considère un tableau de taille n.

1. Écrire un algorihtme affichant les éléments d'un tableau de façon alterné en commençant par le premier élément, puis le dernier, puis le second etc. Par exemple, pour le tableau `[1,2,3,4,5]` il affichera `1,5,2,4,3`.
2. Écrire un algorihtme affichant les éléments d'un tableau de façon alterné en commençant par l'élément du milieu, puis en alternant les éléments à droite et à gauche.Par exemple, pour le tableau `[1,2,3,4,5]` il affichera `3,4,2,5,1`.

Exercice 16

Pour chaque question on considère une matrice carrée de taille $n \times n$.

1. Écrire un algorihtme affichant les éléments d'une matrice carrée, diagonale par diagonale, en commençant par l'élément le plus à droite et le plus haut. Par exemple, pour la matrice

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

   il affichera `3,2,6,1,5,9,4,8,7`.
2. Écrire un algorihtme affichant les éléments d'une matrice carrée, diagonale par diagonale, en commençant celle du milieu puis en alternant les diagonales au-dessus et au-dessous de celle-ci . Par exemple, pour la matrice M il affichera `1,5,9,2,6,4,8,3,7`.
3. Écrire un algorihtme affichant les éléments d'une matrice carrée, en serpentin, c'est-à-dire en commençant par les élements du bord et en se rapprochant du milieu. Par exemple, pour la matrice M il affichera `1,2,3,6,9,8,7,4,5`.

Exercice 17

On considère un tableau `T` de $n+1$ entiers compris entre 1 et n. Chaque entier n'apparait qu'une fois dans le tableau sauf l'un d'eux, apparaissant deux fois, noté `x`.

1. Écrire une relation simple entre la somme des éléments du tableau et `x`.
2. En déduire un algorithme permettant de trouver l'élément en double en effectuant un seul parcours de tableau.

Exercice 18

On considère un tableau `T` de $n+2$ entiers compris entre 1 et n. Chaque entier n'apparait qu'une fois dans le tableau sauf deux d'entre eux, apparaissant deux fois, notés `x` et `y`.

1. Écrire une relation simple entre $x+y$, xy et le polynome unitaire de degré 2 dont x et y sont les racines.
2. En déduire un algorithme permettant de trouver les éléments en double en effectuant un seul parcours de tableau (on admet pouvoir calculer directement $n!$).

Exercice 19

On considère un tableau T de n entiers. On rappellera sous-tableau de T un tableau constitué d'éléments consécutifs de T. On cherche à résoudre le problème suivant :

Problème : Plus grande sous-somme
Entrées : Tableau d'entiers de taille $n \geqslant 1$ et un entier $k \leqslant n$
Sortie : La plus grande somme d'éléments d'un sous-tableau de taille k

Dans cet exexrice on considère que les opérations arithmétiques sur les entiers sont de complexité $O(1)$.

1. Écrire un algorithme `Somme(T,i,j)` qui renvoie la somme des éléments du tableau T entre les indices i et j (compris).
2. En déduire un algorithme naïf de résolution du problème précédent et donner sa complexité en fonction de n et k.
3. Soit $S_{i,j}$ la somme des éléments de T entre les indices i et j. Écrire une relation simple entre $S_{i,j}$ et $S_{i+1,j+1}$.
4. En déduire un algorihtme de complexité $O(n)$ de résolution du problème de la plus grande sous-somme.

Exercice 20

On considère une matrice de taille $n \times m$ dont chaque ligne est remplie uniquement de 0 et de 1, rangés dans l'ordre. Écrire un algorithme de complexité $O(m + n)$ retournant l'indice de la ligne contenant le plus de 1. Par exemple pour la matrice

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

l'algorithme retournera 5.

Exercice 21

On appelle carré magique de taille n un tableau de dimension $n \times n$, contenant l'ensemble des nombres entiers de 1 à n^2, et dont la somme de éléments de chaque ligne, chaque colonne ou chaque diagonale est égale à la même valeur appelée constante magique. Par exemple

$$M = \begin{pmatrix} 8 & 3 & 4 \\ 1 & 5 & 9 \\ 6 & 7 & 2 \end{pmatrix}$$

est un carré magique de taille 3 et sa constante magique est 15.

1. Montrer que la somme des éléments d'un carré de taille n est $\frac{n^2(n^2+1)}{2}$.
2. En déduire que n'importe quel carré magique de taille n a pour constante magique $\frac{n(n^2+1)}{2}$.

Exercice 22 Addition et multiplication binaire

On représente un entier positif en base 2 par un tableau de 0 et de 1. Pour se simplifier l'écriture des algorithmes on décide de placer le chiffre des unités à la première case du tableau.

Adapter les algorithmes d'addition et de multiplication classiques à deux entiers binaires de n bits.

Exercice 23

On représente un entier positif par un tableau de nombres compris entre 0 et 9. Pour se simplifier l'écriture des algorithmes on décide de placer le chiffre des unités à la première case du tableau, suivi de celui des dizaines etc. Par exemple, le nombre 31415 sera représenté par le tableau `[5,1,4,1,3]`.

1. Rappeler l'algorithme de multiplication des entiers vu en cours.

2. Soit

$$A = \sum_{i=1}^{n} a_i 10^{i-1} = a_1 + a_2 10 + a_3 10^2 + \cdots + a_n 10^{n-1}$$

un nombre entier. Montrer que

$$A^2 = \sum_{i=1}^{n} a_i^2 10^{2(i-1)} + 2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_i a_j 10^{i+j-2}.$$

3. En déduire un algorithme de calcul du carré d'un nombre donc la complexité est à peu près moitié moindre que celle de l'algorithme de multiplication.

Exercice 24

Écrire un algorithme itératif qui retourne `VRAI` si une chaîne de caractères est un palindrome et `FAUX` sinon. Analyser sa complexité en meilleur et pire cas.

Exercice 25

Écrire un algorithme traitant 2 chaînes de caractères et qui retourne `VRAI` si la première chaîne est un préfixe de la seconde et `FAUX` sinon. Par exemple, `'algo'` est un préfixe de `'algorithme'` par contre `'ramer'` n'est pas un préfixe de `'rame'`. Faire une analyse en pire et en meilleur cas. Quelle est la complexité générale de l'algorithme ?

Exercice 26

Écrire un algorithme traitant 2 chaînes de caractères et qui retourne `VRAI` si la première chaîne est un suffixe de la seconde et `FAUX` sinon. Par exemple, `'rithme'` est un suffixe de `'algorithme'` par contre `'rme'` n'est pas un suffixe de `'rame'`. Faire une analyse en pire et en meilleur cas. Quelle est la complexité générale de l'algorithme ?

Exercice 27

Écrire un algorithme traitant un tableau d'entiers et qui retourne `VRAI` s'il est trié dans l'ordre croissant et `FAUX` sinon. Faire une analyse en pire et en meilleur cas. Quelle est la complexité générale de l'algorithme ?

Exercice 28 (Examen 2018)

On dit qu'un tableau est trié en vague si pour tout élément du tableau (à l'exception du premier et du dernier) est soit plus grand soit plus petit que ses deux voisins. Par exemple les tableaux `[2,4,1,3,0]` et `[1,1,1,1]` sont triés en vague mais pas le tableau `[2,3,4,1]`.

1. Écrire un algorithme `EstTrie(T)` qui retourne `VRAI` si le tableau `T` (de taille n) est trié en vague et `FAUX` sinon.
2. Faire une analyse en meilleur et pire cas de la complexité de l'algorithme et donner sa complexité générale.

Exercice 29

On considère une matrice de taille $n \times m$ dont chaque ligne est remplie d'entiers. On dit qu'une est creuse si strictement plus de la moitié de ses coefficients sont égaux à 0.

1. Écrire un algorithme `Creuse(M)` qui retourne `VRAI` si la matrice `M` (de taille $n \times m$) est creuse et `FAUX` sinon.
2. Faire une analyse en meilleur et pire cas de la complexité de l'algorithme et donner sa complexité générale.

Exercice 30

On dit qu'une matrice $n \times n$ est triangulaire supérieure si tout les nombres en dessous de la diagonale sont nuls. Par exemple

$$M_1 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 4 & 6 \\ 0 & 0 & 9 \end{pmatrix} \text{ et } M_2 = \begin{pmatrix} 5 & 0 & 0 \\ 0 & 0 & 6 \\ 0 & 0 & 2 \end{pmatrix}$$

sont triangulaires supérieures. Par contre

$$M_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 1 & 2 \end{pmatrix} \text{ et } M_4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

ne le sont pas.

Écrire un algorithme `TRIANGLE_SUP` qui prend en entrée une matrice M sous forme de tableau de tableaux de taille $n \times n$ et qui retourne `VRAI` si est M est triangulaire supérieure et `FAUX` sinon. Faire une analyse en pire et en meilleur cas. Quelle est la compléxité générale de l'algorithme ?

Exercice 31 Point d'équilibre d'un tableau (Examen 2019)

Soit T un tableau d'entier de taille n. On dit qu'un indice k est un point d'équilibre du tableau si la somme des éléments d'indice strictement inférieur à k est égale à celle des éléments d'indice strictement supérieur. Autrement dit :

$$T[1] + T[2] + \cdots + T[k-1] = T[k+1] + T[k+2] + \cdots + T[n]$$

Par exemple, pour le tableau $T = [-2, 5, 7, 6, -1, 2, -4]$, 3 est un point d'équilibre car $-2 + 5 = 6 - 1 + 2 - 4$.

1. Supposons déjà calculés $S_1 = T[1] + T[2] + \cdots + T[k-1]$ et $S_2 = T[k+1] + T[k+2] + \cdots + T[n]$. Comment vérifier si $k+1$ est un point d'équilibre en seulement 2 opérations arithmétiques et une comparaison ?
2. En déduire un algorihtme `PointEq(T)` de complexité $O(n)$ qui renvoie l'indice du point d'équilibre d'un tableau s'il existe et 0 sinon.
3. Faire une analyse en pire et meilleur cas pour justifier sa complexité.

Exercice 32

Test de primalité

On rappelle qu'un nombre premier est un entier positif, différent de 0 et 1 dont les seuls diviseurs sont 1 et lui-même. On dit qu'un entier d divise un entier n si le reste de la division entière de n par d vaut 0. On peut ainsi définir le problème de la primalité d'un nombre entier comme suit :

Problème :	Primalité
Entrée :	Un entier naturel n.
Sortie :	`VRAI` si n est premier, `FAUX` sinon.

1. L'algorithme naïf suivant permet de résoudre le problème de primalité.

```
1  ALGORITHME Primalite(n):entier
2  DONNEES:
3  n: entier
4  VARIABLES:
5  d: entier
6  DEBUT
7  d ← 2
8  TQ d ⩽ n−1 FAIRE
9  SI n % d = 0:
10 RENVOYER FAUX
11 FSI
12 d ← d+1
13 FTQ
14 RENVOYER VRAI
15 FIN
```

   Donner sa complexité en meilleur cas, pire cas et cas général.
2. Montrer que n ne possède pas de diviseur strictement plus grand que $n/2$. La proriété précédente permet-elle d'améliorer la complexité générale de l'algorithme précédent et/ou sa vitesse d'exécution en pratique ?
3. Montrer que si n n'est pas premier alors il existe un diviseur d de n tel que $d \leqslant \sqrt{n}$. En déduire un algorithme de test de primalité plus efficace asymptotiquement que l'algorithme naïf précédent (prouver sa complexité).

Chapitre 4

Algorithmes de tri

Exercice 33

Dans tout l'exercice, `T` est un tableau d'entiers de taille n.

1. Écrire un algorithme `Max2(T,fin)` qui retourne les indices des deux plus grands éléments d'un tableau de taille n (on supposera que $n \geqslant 2$) entre les indices 1 et `fin`.
2. Combien de comparaisons sont effectuées dans le meilleur et le pire cas (en fonction du paramètre `fin`) ? Quelle est la complexité de l'algorithme ?
3. Écrire un algorithme `Tri2Selection(T)` qui tri un tableau de taille **paire** $n \geqslant 2$, à l'aide de l'algorithme précédent.
4. Quelle la complexité de l'algorithme ?

Exercice 34 Examen 2018

On considère un tableau d'entiers `T` de taille `n` et un entier `x`. La distance d'un élément du tableau avec `x` est simplement la valeur absolue de la différence entre `x` et l'élément en question.

Exemple :

```
T = [3, 1, 5, 9, 4], x = 7
distance entre T[1] et x: |7 - 3| = 4
distance entre T[2] et x: |7 - 1| = 6
etc
```

1. Écrire un algorithme `MaxDistance(T,i,x)` qui retourne l'indice de l'élément de `T` qui maximise la distance avec `x` parmi les éléments compris entre l'indice i et l'indice n. Par exemple, avec les données précédentes `MaxDistance(T,1,x)` retournera 2 et `MaxDistance(T,3,x)` retournera 5.
2. Écrire un algorithme de tri `TriDistance(T,x)` qui tri les éléments du tableau en fonction de leur distance à l'entier `x`, du plus éloigné au plus proche.

Exercice 35 Tri cocktail

Le tri cocktail est une amélioration du tri à bulles visant à réduire le coût du pire cas en mélangeant les versions gauche-droite et droite-gauche. La complexité asymptotique n'est par contre pas modifiée.

1. Rappeler quel est le pire cas du tri à bulles. Quel est le pire cas de la version de droite à gauche ?
2. Proposer un algorithme de tri basé sur le tri à bulles alternant les parcours de droite à gauche puis de gauche à droite. Quelle est la complexité du pire cas ?
3. Améliorer l'algorithme en tenant compte du dernier indice où une permutation a eu lieu.

Exercice 36

Tri en vague. Un tableau est dit trié en vague si chaque élément - hors éléments de début et de fin - est soit supérieur soit inférieur à ses deux voisins. Par exemple, le tableau `[2,1,5,2,4,3]` est trié en vague.

1. Proposer un algorithme qui convertit un tableau trié en tableau trié en vague.
2. Écrire un algorithme qui trie en vague un tableau quelconque. Est-il plus efficace de trier dans l'ordre un tableau quelconque puis de le trier en vague ou de le trier en vague directement ?

Exercice 37 Valuation d'un entier

On appelle valuation d'un entier $n > 0$ (que l'on note $val(n)$) la plus grande puissance de 2 qui divise n. Par exemple $val(24) = 3$ car $8 = 2^3$ divise 24 mais pas $16 = 2^4$.

1. Donner la valuation des entiers $12, 56, 256, 2047$.
2. Écrire un algorithme `VAL` de complexité $O(\log_2(m))$ qui renvoie la valulation d'un entier m. Faire une analyse en pire et meilleur cas pour justifier sa complexité.
3. Écrire un algorithme `TriVal(T,m)` qui tri un tableau de taille n d'entiers compris entre 1 et m **basé sur l'algorithme du tri sélection**, de complexité en temps $O(n^2 \log_2(m))$ et de complexité en espace $\Theta(1)$ (c'est-à-dire n'utilisant que des variables entières). Justifier briévement la complexité.
4. Écrire un nouvel algorithme de tri `TriVal2(T,m)`, **basé sur l'algorithme du tri par insertion**, de complexité en temps $O(n^2 + n \log_2(m))$ et de complexité en espace $\Theta(n)$ (on pourra utiliser cette fois une variable de type tableau d'entiers de taille n).

Exercice 38

On suppose avoir deux tableaux triés T_1 et T_2 de tailles respectives m et n.

1. Écrire un algorithme de complexité $\theta(m+n)$ permettant de fusionner ces deux tableaux en un nouveau tableau trié T_3 de taille $m+n$.
2. On suppose que n est divisible par un entier k. On veut fussioner k tableaux de taille $\frac{n}{k}$. Une façon naïve de le faire consiste à d'abord fusionner deux tableaux en un tableau de taille $\frac{2n}{k}$ puis à fusionner ce nouveau tableau avec un autre tableau de taille $\frac{n}{k}$ pour obtenir un tableau de taille $\frac{3n}{k}$ etc. Quel est le coût de cette méthode (en fonction de n et k) ?
3. On considère un tableau d'entiers de taille n. On décide pour le trier de trier indépendamment chaque moitié du tableau puis de les fusionner. Quel est le coût du tri d'une moitié de tableau par rapport au tri complet ? Cette méthode est elle efficace par rapport au tri standard asymptotiquement ? En pratique ?
4. On suppose que n est divisible par un entier k. On souhaite généraliser la méthode précédente en commençant par découper le tableau en k sous tableaux que l'on triera séparément avant de les fusionner. Quel est la complexité d'une telle méthode ? Pour qu'elle valeur de k obtient on le meilleur résultat ?

Exercice 39

Le but de l'exercice est de proposer plusieurs méthodes pour résoudre le problème suivant :

Problème :	Comptage
Entrée :	un tableau d'entiers de taille n.
Sortie :	Le nombre d'entiers différents contenus dans le tableau

Par exemple pour l'instance `[2,0,-1,7,0,0,4,2]` la sortie attendue est 5.

1. Proposer un algorithme naïf reposant sur la méthode qui consiste à vérifier pour chaque élément du tableau si celui-ci est présent ailleurs. Quelle est sa complexité ?
2. Quel est la complexité d'un tri comparatif optimal ? On suppose maintenant disposer d'un tel algorithme de tri : `TRI`. Proposer un nouvel algorithme de résolution du problème de comptage. Quelle est sa complexité ?
3. On restreint le problème à des tableaux dont les éléments sont des entiers compris entre -499 et 500. Proposer un algorithme encore plus efficace. Quelle est sa complexité ?

Exercice 40 Mode d'un tableau (examen 2022)

Soit T un tableau d'entiers de taille n. On appelle *mode* de ce tableau le nombre qui apparait le plus. Si le *mode* n'est pas unique, on choisit le plus petit d'entre eux.

Par exemple, pour le tableau $T = [-2, 0, 7, 0, 5, 0, 5, 2, 5, 4]$, deux nombres apparaissent plus que les autres : 0 et 5. Le mode est donc 0.

1. Écrire un algorithme `Mode(T)` qui renvoie le mode du tableau T (sans le trier).
2. On dispose maintenant d'un algorithme de tri `Tri(T)` de compléxité $O(n\log(n))$. Écrire un algorithme `Mode2(T)` de complexité $O(n\log(n))$ qui renvoie le mode du tableau T.
3. On suppose maintenant que l'ensemble des éléments du tableau sont compris entre $-n$ et n (inclus). Écrire un algorithme `Mode3(T)` de complexité en temps et en espace $\Theta(n)$ qui renvoie le mode du tableau T.

Exercice 41 Examen 2022

On considère dans cet exercice un tableau T d'entiers de taille n dont tous les éléments sont disctincts deux à deux. La position réelle d'un élément du tableau est l'indice auquel il se trouverait si le tableau était trié. Par exemple, pour $T = [2, 0, 8, 5, 4, -1]$ la position réelle de $T[1] = 2$ est 3, celle de $T[6] = -1$ est 1 et celle de $T[3] = 8$ est 6.

1. Énumérer les positions réelles des éléments du tableau $T = [2, -1, 7, 0, 3, 5, 1]$.
2. Écrire un algorithme `Position(T,i)`, de complexité $\Theta(n)$, qui renvoie la position réelle de l'élément d'indice i du tableau T.
3. Quelle valeur de retour de la fonction `Position(T,i)` garantit que l'élément `T[i]` est à sa position réelle ?
4. En déduire un algorithme de tri de complexité $\Theta(n^2)$ faisant appel au plus n fois à la fonction `Echanger`.

Exercice 42

On considère un tableau d'entier `T` de taille n.

1. Écrire un algorithme de complexité $\theta(n)$ permettant de ranger un tableau de n entiers de sorte que tous les entiers pairs soient à droite des entiers impairs. Par exemple, pour l'entrée `[6,2,1,4,3,7]` une sortie possible est `[7,1,3,4,2,6]`.
2. On suppose que le tableau contient autant de nombres pairs que de nombres impairs. En faisant appel à l'algorithme de la question 1 ainsi qu'à un des algos de tri standards, écrire un algorithme rangeant les entiers de la même manière que la première question tout en triant les entiers dans l'ordre croissant. Par exemple, pour l'entrée `[4,7,1,2,6,3]` la sortie sera `[1,3,7,2,4,6]`. Évaluer la complexité de l'algorithme.

Exercice 43

On considère un tableau d'entier `T` de taille n. Écrire un algorithme de complexité $\theta(n)$ permettant de ranger un tableau de n entiers de sorte que tous les entiers négatifs soient à gauche, les 0 au milieu et les entiers positifs à droite. Par exemple, pour l'entrée `[6,0,-1,0,-3,7]` une sortie possible est `[1-,3,0,0,6,7]`.

Exercice 44

On considère un tableau d'entier `T` de taille $2n$ contenant autant d'entiers pairs que d'entiers impairs.

1. Écrire un algorithme de complexité $\theta(n)$ permettant de ranger le tableau de sorte que tous les entiers pairs soient stockés sur les cases d'indices pairs. Par exemple, pour l'entrée `[6,0,2,1,-3,7]` une sortie possible est `[1,0,-3,2,7,6]`.

Exercice 45 Point fixe d'un tableau

On considère le problème algorithmique suivant :

Problème	:	Point fixe
Entrée	:	Tableau T d'entiers **distincts** de taille $n \geqslant 1$
Sortie	:	l'indice i d'un élément du tableau tel que $T[i] = i$ 0 si aucun élément ne satisfait la propriété

Par exemple, pour $T_1 = \{-2, 10, 3, 1, 8, -3\}$ on a le point fixe $T_1[3] = 3$, pour $T_2 = \{11, 0, 2, 7, 5, 6\}$ on a $T_2[5] = 5$ ou $T_2[6] = 6$ et enfin $T_3 = \{2, 3, 4, 5, 6, 7\}$ n'a pas de point fixe.

1. Écrire un algorithme `PointFixe(T)`,de complexité $O(n)$, qui résoud le problème du point fixe. Faire une analyse en pire et meilleur cas pour justifier sa complexité.

2. On suppose maintenant que le tableau T est trié dans l'ordre croissant. Montrer que si i est un point fixe du tableau alors

$$\forall j > i, T[j] \geqslant j.$$

On admettra de plus que $\forall j < i, T[j] \leqslant j$. En déduire un algorithme `PointFixeTrie(T)` de recherche du point fixe dans un tableau trié de complexité $O(\log(n))$ (inutile de la justifier).

3. On considère maintenant une matrice M de taille $m \times n$ dont chacune des m lignes est un tableau de taille n trié dans l'ordre croissant. Écrire un algorithme `Ranger(M)` de complexité $O(m \log(n))$ qui range tous les tableaux contenant un point fixe au début.

Exemple : un rangement possible est le suivant

```
[[-2,0,3,4,10,11],              [[-2,0,3,4,10,11],
 [0,4,7,8,9,10],   Ranger        [-1,0,1,2,5,8],
 [2,3,4,5,6,7],  ----------->    [0,0,0,4,5,6],
 [0,0,0,4,5,6],                  [2,3,4,5,6,7],
 [-1,0,1,2,5,8]]                 [0,4,7,8,9,10]]
```

Remarque : On pourra utiliser sans la définir la procédure `EchangerLigne(M,i,j)` *qui échange les lignes d'indice i et j de la matrice M en temps $\Theta(1)$.*

Chapitre 5

Algorithmes de recherche

Exercice 46

1. Rappeler les complexités des algorithmes de recherche séquentielle et par dichotomie.
2. Analyser le nombre moyen de comparaisons d'éléments des deux algorithmes pour des tableau de 5 éléments.
3. On considère un tableau contenant n éléments valant soit `NOIR` soit `BLANC` rangé par couleur (les éléments noir en premier et on admet qu'il contient toujours les deux couleurs). Écrire un algorithme qui retourne le nombre d'éléments noirs et de complexité $\log(n)$.

Exercice 47 Examen 2018

1. Écrire un algorithme de compléxité $\Theta(n)$ `NegatifPositif(T)` qui traite un tableau d'entiers de taille n et range tous les entiers négatifs ou nuls à gauche du tableau et tous les nombres strictement positifs à droite.
2. On considère un tableau rangé par l'algorithme précédent. Écrire un algorithme en $O(\log(n))$ qui retourne le nombre d'éléments positifs du tableau.

Exercice 48 Recherche par saut

On considère un tableau trié de taille n dans l'ordre croissant. La recherche par saut utilise un paramètre fixe k et consiste à tester dans un premier les éléments d'indices $1, 1+k, 1+2k$ etc puis à faire une recherche séquentielle entre les indices pour lesquels l'élement recherché doit se trouver. Par exemple avec `T=[1,2,3,4,5,6,7,8,9,10]` et $k = 3$, pour rechercher 6 l'algorithme va tester les nombres :1,4,7,5,6.

1. Écrire l'algorithme de recherche par saut qui retourne l'indice de l'élément recherche et 0 si l'élement n'est pas dans le tableau. L'algorithme doit dans un premier temps trouver dans quel sous tableau se trouve potentiellement l'élément puis faire une recherche séquentielle dedans.
2. Étudier la complexité de l'algorithme (meilleur cas, pire cas, cas général).
3. Quelle valeur de k est optimale ?

Exercice 49 Recherche de maximum

On considère un tableau de taille n trié de telle sorte que ses éléments sont d'abord rangé dans l'ordre croissant puis décroissant (ex : [1,2,3,4,6,4,3,0]). On supposera de plus que la suite de nombres est strictement croissante puis strictement décroissante.

1. Étant donnée deux éléments **différents** du tableau `T[i]` et `T[j]`, discuter la position du maximum en fonction de l'ordre relatif de ces deux éléments (`T[i]<T[j]`,`T[i]>T[j]` et `T[i]=T[j]`).
2. En déduire un algorithme de recherche du maximum et étudier sa complexité.

Exercice 50 Médiane de deux tableaux triés

On appelle médiane d'un tableau un nombre séparant la moitié inférieure de la moitié supérieure de l'ensemble des éléments. Dans le cas d'un tableau trié, la médiane sépare le tableau en deux sous-tableaux de même longueur.

Par exemple, 12 est la médiane du tableau `[1,2,6,12,13,15,19]`. Dans le cas où il y a un nombre pair d'éléments, on conviendra que la médiane est la moyenne des deux éléments du milieu. Par exemple 5 est la médiane du tableau `[2,3,4,6,8,15]`.

Dans la suite de cet exercice on considère deux tableaux triés dans l'ordre croissant de taille n. La médiane de deux tableaux est alors la médiane de l'ensemble des entiers des deux tableaux.

1. Écrire un algorithme `Mediane(T)` qui renvoie la médiane d'un tableau trié.
2. Donner la valeur de la médiane de deux tableaux de taille n pour $n = 1$ et $n = 2$.
3. Proposer un premier algorithme de compléxité en temps et en espace $\Theta(n)$ reposant sur l'algorithme `Fusionner` vu précédemment.
4. Soient `m1` et `m2` les medianes de chaque tableau. Discuter de la position de la médiane des deux tableaux en fonction de l'ordre relatif de `m1` et `m2`.
5. En déduire un algorithme de complexité en espace $\Theta(1)$ de recherche de la médiane des deux tableaux et étudier sa complexité.

Exercice 51

Le responsable du cours d'algorithmique monsieur M. souhaite mesurer l'alcoolo-resistance de ses étudiants, qu'il estime être la même pour chacun d'eux. Pour cela il fait l'acquisition de n bouteilles d'alcools de degrés divers qu'il classe du degré le plus faible à celui le plus fort. Le protocole de test consiste alors à faire boire un verre d'une bouteille donnée à un étudiant, si celui-ci s'évanoui c'est que l'alcool était trop fort et l'étudiant en question de peut plus servir pour la suite de l'étude, sinon on le fait vomir immédiatement pour qu'il puisse servir à nouveau pour un autre test.

1. En supposant que l'étudiant est une ressource abondante dans la nature, proposer un algorithme permettant de connaitre le degré maximum que les étudiants peuvent ingurgiter sans faire de coma éthylique en un minimum de tests.
2. Comment faire si on ne dispose que d'un seul étudiant ? De deux ?

Exercice 52 Examen 2019

On considère des tableaux de taille n contenant uniquement les nombres 0 et 1.

1. Écrire un algorithme `Ranger(T)` de complexité $\Theta(n)$ qui tri le tableau `T` dans l'ordre croissant. On pourra utiliser la fonction `Echanger(T,i,j)` vue en cours.

   Par exemple l'algorithme modifiera le tableau `[0,1,0,0,1,0]` en `[0,0,0,0,1,1]`.
2. Écrire un algorithme `NbUn(T)` de complexité $O(\log(n))$ qui prend en entrée un tableau déjà trié et renvoie le nombre de 1 dans le tableau.

   Par exemple pour l'entrée `[0,0,0,0,1,1]` l'algorithme renverra 2.
3. On considère maintenant une matrice de taille $m \times n$ dont chaque ligne ne contient que des 0 et de 1 et est triée dans l'ordre croissant. On dispose d'un algorithme `EchangerLignes(M,i,j)` de complexité $\theta(n)$ qui permute les lignes i et j de la matrice.

   Écrire un algorithme `TriLignes(M)` qui tri les lignes de la matrice `M` dans l'ordre croissant du nombre de 1 quelles contiennent. Par exemple

```
[[0,0,1,1,1,1],                [[0,0,0,0,1,1],
 [0,0,0,0,1,1],   TriLignes     [0,0,0,1,1,1],
 [0,0,0,1,1,1],  ----------->   [0,0,1,1,1,1],
 [1,1,1,1,1,1],                 [0,1,1,1,1,1],
 [0,1,1,1,1,1]]                 [1,1,1,1,1,1]]
```

4. Quelle est sa complexité ?

Exercice 53 Examen 2019

On appelle tableau binaire de taille n un tableau de n éléments ne contenant que les nombres 0 et 1. Dans la suite de l'exercice, `M` désigne une matrice de de taille m dont chaque ligne est un tableau binaire de taille n.

On pourra utiliser si besoin l'algorithme `EchangerLignes(M,i,j)` de complexité $\theta(1)$ qui permute les lignes i et j de la matrice.

1. Écrire un algorithme `PLSUn(T)` qui renvoie la longueur de la plus longue séquence de 1 consécutifs du tableau T. Par exemple pour le tableau `[1,1,0,1,,0,0,1,1,1,1,0,1,1,1]` l'algorithme renverra 4.
2. Écrire un algorithme `Progpager(M,i)` qui parcours la matrice `M` de la ligne d'indice 1 jusqu'à la ligne d'indice i et, pour chaque indice j, échange les lignes consécutives `M[j]` et `M[j+1]` si la plus longue séquence de 1 consécutifs de `M[j]` est supérieure à celle de `M[j+1]`.
3. Écrire un algorithme `TriLignes(M)` qui tri les lignes de la matrice `M` en fonction de la longueur de la plus longue séquence de 1 consécutifs (de la plus petite vers la plus grande). L'algorithme devra être optimisé pour terminer plus vite en cas d'instance favorable.
4. Faire une étude en meilleur et pire cas et donner la complexité générale de l'algorithme `TriLignes(M)`.
5. On suppose maintenant que la matrice `M` a été triée à l'aide de l'algorithme précédent. Écrire un algorithme `ChercheLigne(M,k)` de complexité $O(n\log(m))$ qui renvoie 1 s'il existe une ligne dont la plus longue séquence de 1 consécutifs est égale à k et 0 sinon.

Exercice 54 Examen 2022

1. On considère un tableau T d'entiers de taille n. Écrire un algorithme `RangerMod3(T)` de complexité $\Theta(n)$ qui range les multiples de 3 au début du tableau, suivis des nombres dont le reste vaut 1 modulo 3 et enfin ceux dont le reste vaut 2 modulo 3.
2. On suppose que le tableau T est rangé à l'aide de l'algorithme précédent. Écrire un algorithme `RechercheMod3(T,r)` de complexité $O(\log(n))$ qui renvoie le plus petit indice d'un élément du tableau dont le reste vaut r modulo 3, 0 s'il n'existe pas de tel élément.

   Exemples :

   — pour l'entrée `T=[6,3,9,7,10,1,11,8]` et $r = 1$ l'algorithme renverra 4 car $T[4] = 1 \mod 3$ et $T[3] \neq 1 \mod 3$;

   — pour l'entrée `T=[6,3,9,7,10,1,10,11,8]` et $r = 2$ l'algorithme renverra 8 car $T[8] = 2 \mod 3$ et $T[7] \neq 2 \mod 3$;

   — pour l'entrée `T=[6,3,9,11,8]` et $r = 1$ l'algorithme renverra 0 car il n'existe pas d'élément égal à 1 modulo 3.
3. On considère maintenant deux tableaux d'entiers T_1 et T_2 de taille n rangés à l'aide de l'algorithme `RangerMod3`. On cherche à compter le nombres de paires d'éléments, un dans T_1 et un dans T_2, dont la somme est un multiple de 3. Écrire un algorithme `CompteMod3(T1,T2)` de complexité $O(\log(n))$ qui renvoie ce nombre de paires.

   Exemple : pour l'entrée `T1=[6,9,4,11,8]` et `T2=[0,3,7,10,2]` il y a 9 paires (6,0), (6,3), (9,0), (9,3), (4,2), (11,7), (11,10), (8,7), (8,10).

Chapitre 6

Piles et Files

Exercice 55

Expliquer comment implanter deux piles dans un tableau `T` de taille `n` de manière qu'aucune ne déborde à moins que le nombre total des éléments des deux piles vaille `n`.

Exercice 56

Expliquer comment implanter une file à l'aide de deux piles. Analyser le temps d'exécution des opérations de file.

Exercice 57

Expliquer comment implanter une pile à l'aide de deux files. Analyser le temps d'exécution des opérations de files.

Exercice 58

Une file à double entrée est une file qui autorise l'insertion et la suppression d'éléments à chaque bout.

1. Écrire quatre procédures en $O(1)$ pour insérer et supprimer des éléments de chaque coté d'une file double entrée.
2. On considère une pile dont les éléments sont soit des 0 soit des 1. Écrire un algorithme déplace les éléments d'une telle pile dans une file double entrées de sorte que les 0 soit en queue de file et les 1 en tête.

Exercice 59

Écrire un algorithme qui retourne `VRAI` si une chaîne de caractères est correctement parenthésée et `FAUX` sinon. On considèrera non seulement les parenthèses classiques `()`, mais aussi les crochets `[]` et les accolades {}. Par exemple, pour l'entrée `()[()]` l'algorithme retournera `VRAI` et pour {(}) il retournera `FAUX`.

Exercice 60

Les expressions arithmétiques sont traditionnellement écrites de manière infixe : l'opérateur est placé entre les deux opérandes. Il existe également une notation postfixée avec laquelle l'opérateur est placé après les opérandes. Par exemple $3 + 14$ s'écrit $314+$ et $(2 + 5) * 11$ s'écrit $25 + 11*$. L'intérêt de cette notation est que l'on peut se passer de la priorité des opérateurs pour évaluer une expression.

Écrire un algorithme en $\Theta(n)$ qui prend en entrée une expression arithmétique postfixé donnée sous la forme d'un tableau de n données représentant soit des nombres, soit des opérateurs et qui retourne son évaluation.

On pourra utiliser la fonction `TYPE(x)` qui renvoie `NB` ou `OP` selon que `x` est un nombre ou un opérateur et la fonction `EVAL(a,b,op)` qui retourne le résultat de l'expression `a op b` où `a` et `b` sont des nombres et `op` un opérateur.

Exercice 61

On considère un tableau `T` d'entiers positifs ou nuls de taille n. Étant donné l'élément d'indice i on définit le plus grand élément suivant `T[i]` comme étant le premier élément vérifiant $T[i] < T[j]$ et $i < j$ ou bien -1 si tous les éléments suivants lui sont inférieurs ou égaux. Par exemple, le plus grand élément suivant le dernier élément d'un tableau est toujours -1 et pour le tableau `T=[4,5,2,12]` le plus grand élément suivant les nombres 4,5,2 et 12 est, respectivement, 5,12,12,-1.

1. Écrire un algorithme simple à l'aide de deux boucles permettant d'afficher le plus grand élément suivant chacun des éléments d'un tableau. Étudier sa complexité.
2. On cherche maintenant un algorithme en $\Theta(n)$ pour résoudre le porblème.
   (a) Supposons que les i premiers éléments du tableau forment une suite décroissante et que $T[i] < T[i+1]$. De quel(s) élément(s) $T[i+1]$ est-il le plus grand élément suivant ?
   (b) Supposons maintenant que les i derniers éléments du tableau forment une suite décroissante. Quel est leur plus grand élément suivant à chacun ?
   (c) Écrire un algorithme qui affiche le plus grand élément suivant de chaque élément du tableau, pas nécessairement dans l'ordre, de complexité $\Theta(n)$.

Exercice 62

(Examen 2018)
On considère une pile `P` contenant des entiers tous différents.

1. Écrire un algorithme `DepilerMax(P)` qui permet de supprimer le plus grand entier de la pile `P` sans changer l'ordre des autres éléments. L'algorithme ne doit utiliser qu'une deuxième pile et un entier comme variables auxillaires.
2. Écrire un algorithme `TriPile(P)` qui tri la pile `P` dans l'ordre croissant (le plus grand élément au sommet de la pile). L'algorithme ne doit utiliser qu'une deuxième pile et un entier comme variables auxillaires.

Exercice 63

(Examen 2019)
On rappelle les procédures standards de manipulation des piles : `Empiler(P,x)`, `Depiler(P)`, `Vide(P)`, `Lire(P)`. On considère une pile d'entiers **tous différents**.

Écrire un algorithme `DepilerMinMax(P)` qui permet de supprimer le plus petit entier et le plus grand entier de la pile `P` sans changer l'ordre des autres éléments. L'algorithme ne doit utiliser qu'une deuxième pile et au plus trois entiers comme variables auxillaires.

Exercice 64

Les étudiants du cours d'algorithmique décident de se retrouver au Barathym pour noyer leur désespoir dans la bière. Monsieur M. responsable de ce cours a dit qu'il passerait peut-être observer la resistance à l'alcool de ses chers étudiants. Celui-ci a la propriété d'être connu de tous les étudiants mais ne connait lui aucun d'entre eux.

Arrivé en retard après un apéro corsé, vous distinguez mal les visages des n participants. Vous avez oubliez les noms de tout le monde et n'êtes plus capable que de poser des questions du type "Est-ce que A connais B ?". Le but du jeu et de trouver monsieur M. ou de s'assurer qu'il n'est pas venu en un minimum de questions.

(On pourra modéliser l'entrée du problème comme étant un tableau de nombres représentant les différents participants et la question comme une fonction `Connais(A,B)` prenant deux élements du tableau en paramètre et renvoyant `VRAI` si `A` connais `B` et `FAUX` sinon.)

1. En fonction du resultat de la fonction `Connais(A,B)` que peut en déduire sur `A` ou `B`.
2. Montrer que pour une soirée à 3 personnes, le problème peut être résolu en 5 questions.
3. À l'aide d'une structure de pile, écrire un algorithme en $\Theta(n)$ qui résoud le problème.

Exercice 65

Énumération Écrire un algorithme qui prend comme entrée un entier n et affiche l'écriture ternaire (en base 3) de tous les nombres entre 1 et $3^n - 1$.

Exercice 66

Propagation On considère une grille 2D remplie de nombres entiers positifs (l'origine est en haut à gauche). Écrire un algorithme `Propagation(x,y)` qui prend en paramètre les coordonnées `x` et `y` d'une case soustrait 1 à toutes les cases adjacentes, puis aux cases à distance 2, 3 etc jusqu'à rencontrer une case contenant un 0.

Par exemple :

```
 ---------------                                  ---------------
|0  0  0  1  2  0  0|                            |0  0  0  0  1  0  0|
|0  0  2  3  2  1  0|                            |0  0  1  2  1  0  0|
|1  0  1  2  2  2  2| --Propagation(4, 1)-->     |1  0  0  1  1  1  1|
|0  0  0  1  0  0  1|                            |0  0  0  0  0  0  0|
|0  0  0  0  0  0  0|                            |0  0  0  0  0  0  0|
|1  1  0  3  1  2  0|                            |1  1  0  3  1  2  0|
 ---------------                                  ---------------
```