Colles analyse et représentations

Exercice 1

On définit la suite de Fibonacci par $F_0 = F_1 = 1$ et $\forall n \geq 2$, $F_n = F_{n-1} + F_{n-2}$.

1. Donner un algorithme récursif naïf permettant de calculer le n-ème terme de cette suite.
2. On note N_n le nombre d'appels à cet algorithme sur l'entier n. Exhiber une relation de récurrence vérifiée par $(N_n)_{n\in\mathbb{N}}$. En déduire que $\forall n \in \mathbb{N}$, $N_n = 2F_n - 1$.
3. En déduire la complexité de cet algorithme. Qu'en penser ?
4. Donner un algorithme de complexité linéaire en n permettant de calculer F_n.
5. a) Rappeler un algorithme permettant de calculer le pgcd de deux nombres a et b.
   b) "Le nombre de divisions euclidiennes dans le calcul du pgcd de a et b est maximal lorsque a et b sont deux termes consécutifs de la suite de Fibonacci." Justifier.
   c) En déduire l'ordre de grandeur d'un majorant pour le nombre de divisions euclidiennes effectuées lors d'un calcul de pgcd.

Exercice 2

On considère la suite suivante : $f_0 = 0$, $f_1 = 1$ et $\forall n \geq 1, f_{2n} = f_n$ et $f_{2n+1} = f_n + f_{n+1}$.

1. Donner un récursif algorithme naïf permettant de calculer f_n.
2. On considère la suite $(v_n)_{n\in\mathbb{N}}$ telle que $\forall n \in \mathbb{N}$, $v_n = (f_{2n}, f_{2n+1})$.
   a) Montrer que l'on peut calculer explicitement v_n uniquement à partir de $v_{n/2}$.
   b) En déduire un nouvel algorithme permettant de calculer f_n.
   c) Quelle est la complexité de cet algorithme ?
3. Quelle était la complexité de votre algorithme naïf ?

Exercice 3

On considère deux polynômes $P = \sum_{i=0}^{n} a_i X^i$ et $Q = \sum_{j=0}^{m} b_j X^j$ et on note $N = \max(m, n) + 1$.

1. Quelle est la complexité d'un algorithme permettant de calculer la somme de P et Q ?
2. On rappelle que le produit des polynômes P et Q est le polynôme $PQ = \sum_{k=0}^{n+m} \left(\sum_{i=0}^{k} a_i b_{k-i} \right) X^k$. Donner un algorithme permettant d'obtenir le produit de deux polynômes. On réfléchira à une façon pertinente de représenter les données en entrée.
3. Quelle est la complexité de votre algorithme ?
4. En notant $M = N/2$ (division entière), on peut toujours écrire P et Q sous la forme

$$P = P_1 + X^M P_2 \quad Q = Q_1 + X^M Q_2$$

avec P_1, P_2, Q_1, Q_2 quatre polynômes de degré inférieur à M.

   a) Montrer que $PQ = R_1 + X^M(R_2 - R_1 - R_3) + X^{2M} R_3$ avec $R_1 = P_1 Q_1$, $R_2 = (P_1 + P_2)(Q_1 + Q_2)$ et $R_3 = P_2 Q_2$. *Le mathématicien russe Karatsuba fut le premier à remarquer cette égalité remarquable.*
   b) On note $C(N)$ le nombre d'opérations effectuées pour multiplier deux polynômes de degrés n, m tels que $\max(n, m) + 1 = N$ avec la méthode précédente. Monter que $C(N) = 3C(N/2) + O(N)$.
   c) En utilisant un arbre récursif similaire à celui utilisé pour déterminer la complexité du tri fusion, montrer que $C(N) = O(N^\alpha)$ où on précisera la valeur exacte de α. On pourra commencer par traiter le cas où N est une puissance de deux. Conclure.

Exercice 6

On considère un tableau A de taille n. On pourra dans un premier temps supposer que n est une puissance de deux. Un élément x de A est dit majoritaire si et seulement si A contient strictement plus de $n/2$ occurrences de x.

1. Ecrire un algorithme occurrences permettant de calculer le nombre d'occurrences de x dans un tableau A entre les indices i et j. Etudier sa complexité.
2. Supposons qu'on divise A en deux parties égales et qu'on soit en mesure de savoir sur chacune des moitiés si elle comprend un élément majoritaire et si oui quelle est sa valeur. Expliquer comment calculer la valeur majoritaire de A si elle existe à partir de cette connaissance.
3. Ecrire un algorithme majoritaire prenant en entrée un tableau A et deux indices dans ce tableau i, j et qui renvoit le couple (Vrai, x) si x est majoritaire dans $A[i, j]$ et (Faux, -1) si $A[i, j]$ ne contient pas d'élément majoritaire en utilisant la question précédente.
4. a) Si on note $C(n)$ le nombre d'opérations effectuées sur l'entrée $(A, 1, n)$, montrer que

$$C(n) = 2C(n/2) + \Theta(n)$$

   b) En déduire la complexité de l'algorithme majoritaire.

Exercice 8

1. Concevoir un algorithme permettant de compter le nombre d'occurrences dans une liste et déterminer sa complexité.
2. La suite de Golomb est l'unique suite croissante définie comme suit : $g_1 = 1$, $g_2 = 2$ et pour tout $n \geq 1$, g_n est le nombre de fois que n apparaît dans la suite $(g_n)_{n\in\mathbb{N}^\star}$.
   a) Donner les 20 premiers termes de la suite de Golomb.
   b) Déterminer un algorithme renvoyant le n-ème terme de la suite de Golomb.
   c) Etudier sa complexité.

Exercice 9

On considère un tableau A de taille n. On pourra dans un premier temps supposer que n est une puissance de deux. Un élément x de A est dit majoritaire si et seulement si A contient strictement plus de $n/2$ occurrences de x. On considère l'algorithme suivant :

```
pseudomajoritaire(A, i, j) =
  Si i = j
    Renvoyer (Vrai, A[i], 1)
  (r_x, x, c_x) ← pseudomajoritaire(A, i, (i+j-1)/2)
  (r_y, y, c_y) ← pseudomajoritaire(A, (i+j+1)/2, j)
  Si r_x = Faux et r_y = Faux
    Renvoyer (Faux, −1, −1)
  Si r_x = Vrai et r_y = Faux
    Renvoyer (Vrai, x, c_x + (j−i+1)/4)
  Si r_x = Faux et r_y = Vrai
    Renvoyer (Vrai, y, c_y + (j−i+1)/4)
  Si r_x = Vrai et r_y = Vrai
    Si x = y
      Renvoyer (Vrai, x, c_x + c_y)
    Sinon
      Si c_x = c_y
        Renvoyer (Faux, −1, −1)
      Sinon
        Si c_x > c_y
          Renvoyer (Vrai, x, (j−i+1)/2 + c_x − c_y)
        Sinon
          Renvoyer (Vrai, y, (j−i+1)/2 + c_y − c_x)
```

1. Montrer l'assertion suivante : si pseudomajoritaire(A, i, j) renvoit (Faux, -1, -1) alors A n'a pas d'élément majoritaire. Dans les autres cas, pseudomajoritaire(A, i, j) renvoit (Vrai, x, c_x) tel que $c_x > n/2$ avec n la longeur de $A[i, j]$ et x apparaît au plus c_x fois dans A et tout autre élément de A apparaît au plus $n - c_x$ fois dans A.

2. En notant $C(n)$ le nombre d'opérations effectuées si la taille de $A[i, j]$ vaut n, donner une relation de récurrence sur $C(n)$.

3. En déduire la complexité de pseudomajoritaire.

4. A partir de l'algorithme pseudomajoritaire, déduire un algorithme majoritaire qui vérifie si un tableau A contient un élément majoritaire et déterminer sa complexité.

Exercice 10

On considère le problème suivant : n personnes sont disposées en cercle, numérotées de 1 à n. Elles sont éliminées du cercle selon la règle suivante : le joueur en position numéro 2 est éliminé, puis celui deux positions plus loin, puis celui deux positions plus loin et ce jusqu'à ce qu'il ne reste plus qu'une personne dans le cercle, qui est alors le gagnant. Pour tout $n \geq 1$, on note $f(n)$ la position du gagnant si le cercle compte n personnes initialement. Ainsi, $f(4) = 1$ et $f(5) = 3$.

1. a) Déterminer $f(1)$ et $f(7)$.
   b) Montrer que $\forall j \geq 1$, $f(2j) = 2f(j) - 1$ et $f(2j + 1) = 2f(j) + 1$.
   c) En déduire un algorithme permettant de déterminer la position du gagnant en fonction de n.
   d) Déterminer sa complexité.

2. a) Déterminer $f(j)$ pour $j \in [\![1, 16]\!]$.
   b) En extrapoler une formule explicite pour $f(n)$ et montrer qu'elle est vérifiée.
   c) En déduire un nouvel algorithme pour calculer $f(n)$. Déterminer sa complexité et la comparer avec celle de la question précédente.

Exercice 11

On considère l'algorithme suivant, prenant en entrée deux entiers naturels non nuls :

```
mult(m,n) =
  Si n = 1
    Renvoyer m
  Sinon
    Renvoyer mult(m,n−1) + m
```

1. Etudier la terminaison, la correction et la complexité de cet algorithme.

On considère à présent l'algorithme suivant ; m, n sont toujours des entiers naturels non nuls :

```
russe(m,n) =
  res ← 0
  Tant que n ≠ 0
    Si n est impair
      res ← res+m
      n ← n−1
    n ← n/2
    m ← m × 2
  Renvoyer res
```

2. a) Que fait cet algorithme ?
   b) Montrer sa terminaison puis étudier sa complexité et la comparer avec celle obtenue précédemment.
   c) Montrer la correction de cet algorithme à l'aide d'un invariant judicieux.