Sorbonne Université

3I003 : Algorithmique

Projet :

Confitures

Ahmed Boukerram
Angelo Ortiz

Licence d'Informatique
Année 2018/2019

Table des matières

1 Introduction

Une séquence d'ADN (acide Desoxyribonucleique) est une suite de nucléotides, chacun comprenant trois éléments

— une molécule de base : l'adénine A, la guanine G, la cytosine C ou la thymine T,
— un sucre, le désoxyribose,
— et du phosphate.

Chaque nucléotide est désigné par la première lettre de sa molécule de base. Une séquence d'ADN est ainsi un mot sur l'alphabet $\mathcal{A} = \{A, C, G, T\}$. Par exemple, $a = ATCGGCTGCATTTCGA$ représente une séquence d'ADN.

Bien que l'on considère la structure d'une séquence d'ADN comme une suite de nucléotides, sa représentation physique est plus complexe : en effet, les nucléotides se regroupent par paires, l'adénine A avec la thymine T et la guanine G avec la cytosine C.

La structure secondaire S d'une séquence d'ADN $a = a_1 \cdots a_n$ de taille n est définie comme un ensemble de couples d'indices (i, j), avec $1 \leq i < j \leq n$ vérifiant les propriétés suivantes :

1. S définit un couplage : pour tout indice $i \in \{1, \ldots, n\}$, il existe au plus un couple de S contenant i.
2. On suppose que il n'y a pas de nœud : il n'existe pas de couple (i_1, j_1) et (i_2, j_2) de S avec $i_1 < i_2 < j_1 < j_2$.
3. Tout couple $(i, j) \in S$ vérifie $(a_i, a_j) \in \{(A, T), (T, A), (G, C), (C, G)\}$.

Pour tout couple $(i, j) \in S$, on dira que i est couplé avec j et j est couplé avec i

Le problème que l'on se pose revient à déterminer une structure secondaire S de a dont le nombre de couples est maximum. Par exemple, pour la séquence $a = TCGGCTGCATTTCGA$, une structure secondaire de taille maximale est donnée par $S = \{(1, 15), (2, 14), (3, 13), (4, 8), (5, 7), (9, 10)\}$.

2 Définition et évaluation des algorithmes à programmer

Soit une séquence d'ADN $a = a_1 \cdots a_n$ de taille n. Pour tout couple (i, j) d'indices tels que $1 \leq i \leq j \leq n$ on note $a_{i,j}$ la sous-séquence $a_i \cdots a_j$ de a, $S_{i,j}$ une structure secondaire de $a_{i,j}$ dont le nombre de couples est maximum et $E_{i,j} = |S_{i,j}|$ le nombre d'éléments de $S_{i,j}$.

Pour l'exemple précédent, $a_{3,13} = GGCTGCATTTC$, $S_{3,13} = \{(3, 13), (4, 8), (5, 7), (9, 10)\}$ et $E_{3,13} = 4$.

2.1 Question 1

Pour tout $i \in \{1, \ldots, n\}$, $a_{i,i}$ est la sous-séquence a_i de a. Donc, on ne peut pas former de couple d'indices distincts dans $a_{i,i}$. De ce fait, $S_{i,i} = \emptyset$ et $E_{i,i} = 0$.

2.2 Question 2

1. Supposons que ni i ni j ne soient couplés dans $S_{i,j}$.
   D'un côté, on sait que $T_{i+1,j-1} \subseteq S_{i,j}$. Par hypothèse, $S_{i,j} = T_{i+1,j-1}$.
   De l'autre côté, $S_{i+1,j-1}$ représente la séquence secondaire de la sous-séquence $a_{i+1,j-1}$. Or, $T_{i+1,j-1}$ est l'ensemble des couples de la séquence $a_{i,j}$ à valeurs dans l'intervalle $\{i+1, \ldots, j-1\}$. En considérant l'hypothèse, on en déduit que $T_{i+1,j-1} = S_{i+1,j-1}$ et plus particulièrement $|T_{i+1,j-1}| = |S_{i+1,j-1}|$.
   On obtient donc par transitivité que $E_{i,j} = E_{i+1,j-1}$.
2. Supposons que j ne soit pas couplé dans $S_{i,j}$.
   D'une part, on sait que $T_{i,j-1} \subseteq S_{i,j}$. Par hypothèse, $S_{i,j} = T_{i,j-1}$.
   D'autre part, $S_{i,j-1}$ représente la séquence secondaire de la sous-séquence $a_{i,j-1}$. Or, $T_{i,j-1}$ est l'ensemble des couples de la séquence $a_{i,j}$ à valeurs dans l'intervalle $\{i, \ldots, j-1\}$. En considérant l'hypothèse, on en déduit que $T_{i,j-1} = S_{i,j-1}$ et plus particulièrement $|T_{i,j-1}| = |S_{i,j-1}|$.
   On obtient donc par transitivité que $E_{i,j} = E_{i,j-1}$.
3. Supposons que le couple $(i, j) \in S_{i,j}$.
   D'abord, on sait que $T_{i+1,j-1} \subseteq S_{i,j}$. Par hypothèse, $S_{i,j} = \{(i, j)\} \cup T_{i+1,j-1}$. De plus, puisque $\{(i, j)\} \cap T_{i+1,j-1} = \emptyset$, $|S_{i,j}| = 1 + |T_{i+1,j-1}|$.
   Par ailleurs, $S_{i+1,j-1}$ représente la séquence secondaire de la sous-séquence $a_{i+1,j-1}$. Or, $T_{i+1,j-1}$ est l'ensemble des couples de la séquence $a_{i,j}$ à valeurs dans l'intervalle $\{i+1, \ldots, j-1\}$. En considérant l'hypothèse, on en déduit que $T_{i+1,j-1} = S_{i+1,j-1}$ et plus particulièrement $|T_{i+1,j-1}| = |S_{i+1,j-1}|$.
   On obtient donc par transitivité que $E_{i,j} = 1 + E_{i+1,j-1}$.
4. Supposons que le couple $(k, j) \in S_{i,j}$, avec $k \in \{i+1, \ldots, j-1\}$.
   Premièrement, on sait que $T_{i,k-1}, T_{k,j} \subseteq S_{i,j}$. Par hypothèse et par le fait qu'il ne peut pas y avoir de nœuds, $S_{i,j} = T_{i,k-1} \cup T_{k,j}$. De plus, puisque $T_{i,k-1} \cap T_{k,j} = \emptyset$, $|S_{i,j}| = |T_{i,k-1}| + |T_{k,j}|$.

Deuxièmement, $S_{i,k-1}$ représente la séquence secondaire de la sous-séquence $a_{i,k-1}$. Or, $T_{i,k-1}$ est l'ensemble des couples de la séquence $a_{i,j}$ à valeurs dans l'intervalle $\{i, \ldots, k-1\}$. En considérant l'hypothèse, on en déduit que $T_{i,k-1} = S_{i,k-1}$ et plus particulièrement $|T_{i,k-1}| = |S_{i,k-1}|$.
Réciproquement, on déduit $|T_{k,j}| = |S_{k,j}|$.
On obtient donc par transitivité que $E_{i,j} = E_{i,k-1} + E_{k,j}$.

2.3 Question 3

1. Soit la propriété suivante P :

$$E_{i,j} = \max(E_{i+1,j-1} + e(i,j), E_{i,j-1}, \max_{i<k<j} E_{i,k-1} + E_{k,j}),$$
$$\forall i \in \{1, \ldots, n-1\}, \forall j \in \{i+1, \ldots, n\}.$$

Montrons que P est vraie.
Vérifions d'abord que cette propriété est vraie pour tout couple $(i, i+1)$ pour $i \in \{1, \ldots, n-1\}$. On traite séparément chaque argument de la fonction `max`.
Le premier argument est alors $E_{i+1,i} + e(i, i+1)$. Or, d'après l'énoncé, $E_{i+1,i} = 0$. Donc le premier terme est réduit à $e(i, i+1)$.
De plus, d'après la question 1, le deuxième argument est $E_{i,i} = 0$.
Finalement, $\nexists k \in \{1, \ldots, n\}$ tel que $i < k < i+1$. D'où, on peut négliger le dernier argument $\max_{i<k<i+1} E_{i,k-1} + E_{k,i+1}$.
Donc, $\max(E_{i+1,i} + e(i, i+1), E_{i,i}, \max_{i<k<i+1} E_{i,k-1} + E_{k,i+1}) = e(i, i+1)$. Or, $E_{i,i+1} = e(i, i+1)$. D'où, on obtient la'égalité cherchée. ■
Vérifions maintenant que cette propriété est vraie pour tout couple (i, j) pour $i \in \{1, \ldots, n-1\}$ et $j \in \{i+2, \ldots, n-1\}$.
D'après la question précédente, quatre cas sont possibles selon le couplage des bornes :

(a) Aucune des deux bornes de $S_{i,j}$ n'est couplée.
Donc, $E_{i,j} = E_{i+1,j-1}$. De plus, $e(i,j) = 0$.
D'où, on peut poser $E_{i,j} = E_{i+1,j-1} + e(i,j)$.

(b) L'une des deux bornes de $S_{i,j}$ n'est pas couplée.
Donc, $E_{i,j} = \max(E_{i+1,j}, E_{i,j-1})$.

(c) Les deux bornes de $S_{i,j}$ sont couplées, c'est-à-dire $(i,j) \in S_{i,j}$.
Donc, $E_{i,j} = 1 + E_{i+1,j-1}$. De plus, $e(i,j) = 1$.
D'où, on peut poser $E_{i,j} = E_{i+1,j-1} + e(i,j)$.

(d) L'une des deux bornes de $S_{i,j}$ est couplée avec un élément $k \in \{i+1, \ldots, j-1\}$.
Donc, $E_{i,j}$ est obtenu pour une partition en deux sous-séquences de la forme $a_i \cdots a_{k-1}$ et $a_k \cdots a_j$. D'où, $E_{i,j} = \max_{i+2 \leq k \leq j-1} (E_{i,k-1} + E_{k,j})$.

On sait que pour chaque cas l'expression donnée pour $E_{i,j}$ correspond au cardinal d'une structure secondaire $S_{i,j}$ de taille maximale. On en déduit que

$$E_{i,j} = \max(E_{i+1,j-1} + e(i,j), \max(E_{i+1,j}, E_{i,j-1}), \max_{i+2 \leq k \leq j-1}(E_{i,k-1} + E_{k,j})). \tag{1}$$

Étant donné qu'il y a des fonctions `max` imbriquées dans (1), on peut réordonner les arguments et passer ainsi le premier de la deuxième fonction `max` à la troisième avec $k = i + 1$. Donc,

$$E_{i,j} = \max(E_{i+1,j-1} + e(i,j), E_{i,j-1}, \max_{i+1 \leq k \leq j-1}(E_{i,k-1} + E_{k,j})). \tag{2}$$

Autrement dit,

$$E_{i,j} = \max(E_{i+1,j-1} + e(i,j), E_{i,j-1}, \max_{i<k<j}(E_{i,k-1} + E_{k,j})). \qquad \blacksquare$$

2. De la même manière que pour réordonner des arguments dans la question précédente, on passe le deuxième argument à la fonction `max` imbriquée avec $k = j$. Donc,

$$E_{i,j} = \max(E_{i+1,j-1} + e(i,j), \max_{i<k\leq j}(E_{i,k-1} + E_{k,j})),$$
$$\forall i \in \{1, \ldots, n-1\}, \forall j \in \{i+1, \ldots, n\}. \qquad \square$$

2.4 Question 4

On déduit du paragraphe précédent une fonction récursive `tailleMaxRec` qui pour une séquence a de taille n et deux entiers $(i,j) \in \{1, \ldots, n\}$ retourne $E_{i,j}$. En voici le pseudocode :

```
1: function E(a,i,j)
2:     couples ← [('A','T'),('T','A'),('G','C')('C','G')]
3:     if (a[i − 1], a[j − 1]) in couples then return 1
4:     return 0
```

```
1: function TAILLEMAXREC(a,i,j)
2:     if j − i < 1 then return 0
3:     return max(tailleMaxRec(a, i + 1, j − 1)+e(i, j),
         max_{i<k≤j} (tailleMaxRec(a, i, k − 1)+tailleMaxRec(a, k, j)))
```

2.5 Question 5

Soient $i \in \{1, \ldots, n\}, j \in \{i, \ldots, n\}$. Posons $p = j - i \in \{0, \ldots, n-1\}$. Soit la propriété $P(p)$: `tailleMaxRec` se termine et est valide.

Montrons-la par récurrence forte sur p.

N.B. On utilisera dans la suite des questions la sigle *IE* pour instruction élémentaire.

Cas négatif : Pour $j < i$, `tailleMaxRec` est valide et se termine car on a posé $E_{i,j} = 0$. Ainsi, on rentre dans le branchement conditionnel et renvoie bien 0. De plus, on n'a réalisé que des *IE* en nombre fini.

Base : Pour $p = 0$, $j = i$. On rentre dans le branchement conditionnel. Alors, on renvoie 0. Or, d'après la question 1, $E_{i,i} = 0$. Donc, la fonction est valide.

De la même manière que pour prouver que la fonction se termine lorsque $j < i$, on montre qu'elle finit dans ce cas-là.

De ces faits, $P(0)$ est vraie.

Induction : Supposons $P(p)$ vraie pour tout $p < p_0$, pour un $p_0 > 0$ fixé. Montrons $P(p_0)$ vraie.

Comme $p_0 > 0$, on ne rentre pas dans le branchement conditionnel. Donc, on renvoie le maximum de deux expressions :

- `tailleMaxRec`$(a, i+1, j-1) +$ `e`(i, j). Par hypothèse de récurrence, $P(p)$ est vraie pour $0 \leq p = j - i - 2 < p_0$. De plus, d'après la remarque du cas négatif, la fonction se termine et est valide lorsque $j - i - 2 < 0$. Quant à `e`(i,j), cette fonction ne contient que des *IE* et donc se termine. De plus, elle est bien valide car la valeur de retour est 0 ou 1 selon la possibilité de couplage entre i et j.
- $\max_{i<k\leq j}$ `tailleMaxRec`$(a, i, k-1) +$ `tailleMaxRec`(a, k, j). Pour tout $k \in \{i+1, \ldots, j\}$,
  - `tailleMaxRec`$(a, i, k-1)$ finit et est valide par hypothèse de récurrence : $0 \leq k - 1 - i < p_0$.
  - `tailleMaxRec`(a, k, j) finit et est valide par hypothèse de récurrence : $0 \leq j - k < p_0$.

  Par ailleurs, la fonction `max` est finie et renvoie bien le maximum des $j - i$ valeurs de la somme.

Dans la deuxième partie de la question 3, on a montré une formule calculant $E_{i,j}$, c'est-à-dire la taille maximale d'une séquence de taille $j - i + 1$ avec $i < j$, ce qui est bien le cas ici. On sait que cette formule est transcrite dans le code de la fonction `tailleMaxRec`. De plus, on vient de montrer que la valeur de retour de la fonction est finie et correspond au maximum des deux arguments décrits ci-dessus. De ces faits, $P(p_0)$ est vraie.

Conclusion :

$$\left.\begin{array}{l} P(0) \text{ vraie} \\ \forall p_0 \in \{1, \ldots, n-1\}, [(\forall p < p_0, P(p)) \implies P(p_0)] \end{array}\right\} \begin{array}{l} \forall p \in \{0, \ldots, n-1\}, \\ \texttt{tailleMaxRec} \text{ se termine} \\ \text{et est valide.} \end{array}$$

2.6 Question 6

Soit u_p le nombre d'appels de la fonction `tailleMaxRec` effectués pour $p = j - i$.

1. $u_0 = 1$ car on rentre dans le branchement conditionnel.
   $u_1 = 4$ car on ne rentre pas dans le branchement conditionnel et fait 1 appel pour le premier argument de la fonction `max` et 2 appels pour le deuxième argument.

2. Soit $p \geq 2$. Soit la propriété $Q(p) : u_p = u_{p-2} + 1 + 2\sum_{i=0}^{p-1} u_i$.
   Montrons-la par démonstration directe.
   Soit $p \geq 2$. Comme $p > 1$, on ne rentre pas dans le branchement conditionnel de la fonction `tailleMaxRec`. Donc, on renvoie le maximum de deux expressions :
   — `tailleMaxRec`$(a, i+1, j-1)$+`e`(i, j). Ici, le nombre d'appels à la fonction est u_{p-2} car $j - 1 - (i + 1) = j - i - 2 = p - 2$.
   — $\max_{i<k\leq j}$ `tailleMaxRec`$(a, i, k-1)$ + `tailleMaxRec`(a, k, j). Pour tout $k \in \{i + 1, \ldots, j\}$,
      — `tailleMaxRec`$(a, i, k-1)$ réalise u_{k-1-i} appels à la fonction.
      — `tailleMaxRec`(a, k, j) réalise u_{j-k} appels à la fonction.

   Donc, on a $u_p = u_{p-2} + 1 + \sum_{k=i+1}^{j} u_{k-1-i} + u_{j-k}$.
   D'abord, on ré-indexe la somme : $u_p = u_{p-2} + 1 + \sum_{k=0}^{j-i-1} u_k + u_{j-k-i-1}$.
   Or, $p = j - i$. Donc, on sépare les termes :
   $u_p = u_{p-2} + 1 + \sum_{k=0}^{p-1} u_k + \sum_{k=0}^{p-1} u_{p-1-k}$.
   Puis, on inverse l'indice de la deuxième somme :
   $u_p = u_{p-2} + 1 + \sum_{k=0}^{p-1} u_k + \sum_{k=0}^{p-1} u_k$.
   Finalement, on peut changer la variable des sommes et les regrouper :

$$u_p = u_{p-2} + 1 + 2\sum_{i=0}^{p-1} u_i. \qquad \square$$

3. Soit la propriété $R(p) : 2^p \leq u_p \leq 4^p$.
   Montrons-la par récurrence forte sur p.
   Base : Pour $p = 0$, d'après la sous-question 1, $2^0 = 1 \leq u_0 = 1 \leq 1 = 4^0$. Donc, $R(0)$ est vraie.
   Pour $p = 1$, d'après la sous-question 1, $2^1 = 2 \leq u_1 = 4 \leq 4 = 4^1$. Donc, $R(1)$ est vraie.
   Induction : Supposons $R(p)$ vraie pour tout $p < p_0$, pour un $p_0 > 1$ fixé. Montrons $R(p_0)$ vraie.
   D'après la sous-question 2, $u_{p_0} = u_{p_0-2} + 1 + 2\sum_{i=0}^{p_0-1} u_i$.
   Or, par hypothèse de récurrence, pour tout $i \in \{0, \ldots, p_0 - 1\}, Q(i)$ est vraie. Donc, $2^{p_0-2} + 1 + 2\sum_{i=0}^{p_0-1} 2^i \leq u_{p_0} \leq 4^{p_0-2} + 1 + 2\sum_{i=0}^{p_0-1} 4^i$. D'où,
   $2^{p_0-2} + 1 + 2\dfrac{2^{p_0} - 1}{2 - 1} \leq u_{p_0} \leq 4^{p_0-2} + 1 + 2\dfrac{4^{p_0} - 1}{4 - 1}$, c'est-à-dire
   $2^{p_0} \leq 2^{p_0-2} + 2^{p_0+1} - 1 \leq u_{p_0} \leq 4^{p_0-2} + 2\dfrac{4^{p_0}}{3} + \dfrac{1}{3} \leq 4^{p_0}$.

Conclusion :

$$\left.\begin{array}{l} R(0), R(1) \text{ vraies} \\ \forall p_0 \in \{2, \ldots, n-1\}, [(\forall p < p_0, R(p)) \implies R(p_0)] \end{array}\right\} \begin{array}{l} \forall p \in \{0, \ldots, n-1\}, \\ 2^p \leq u_p \leq 4^p \end{array}$$

4. D'une part, d'après le paragraphe précédent, $R(n-1)$ est vraie, c'est-à-dire $2^{n-1} \leq u_{n-1} \leq 4^{n-1}$, où $n = p+1 = j-i+1$ est la longueur de la séquence.

   D'autre part, chaque appel à `tailleMaxRec` ne contient que des *IE* à part ses appels récursifs (on considère que la fonction `max` est optimale et fait donc les comparaisons *in situ*).

   Donc, on peut exprimer la complexité de la fonction en nombre d'appels récursifs effectués. De ce fait, la complexité de `tailleMaxRec` est en $\Omega(2^n)$ et $\mathcal{O}(4^n)$.

2.7 Question 7

On utilisera un tableau à deux dimensions pour stocker les valeurs de $E_{i,j}$. De plus, on considèrera que l'accès à une case du tableau et l'appel à `e()` sont des instructions élémentaires compte tenu de leur complexité : $\Theta(1)$.

Tout d'abord, on initialise la première case de la première ligne à 0. Donc, la complexité de cette partie est en $\Theta(1)$.

Puis, la première boucle `for` est effectuée $n-1$ fois et à chaque tour de boucle on ne réalise que des *IE*. Donc, la complexité de cette partie est en $\Theta(n)$.

Finalement, on effectue deux boucles l'une à l'intérieur de l'autre. La boucle imbriquée consiste en le calcul du maximum de deux expressions. Les opérations arithmétiques n'étant pas d'instructions représentatives, la première d'entre elles comprend deux *IE*. En ce qui concerne le deuxième argument, le nombre d'*IE* effectuées est $2p$. Donc, le nombre d'*IE* réalisées dans cette partie est $c = \sum_{p=1}^{n-1} \sum_{i=1}^{n-p} 2p+2 = \sum_{p=1}^{n-1} (2p+2)(n-p) = 2\sum_{p=1}^{n-1} [p(n-1)+n-p^2]$. D'où, $c = (n-1)^2 n + 2(n-1)n - \dfrac{(n-1)n(2n-1)}{3} = \dfrac{(n-1)n(n+4)}{3}$. Donc, la complexité de cette partie est en $\Theta(n^3)$.

De ces trois faits, on en déduit que la complexité de l'algorithme est en $\Theta(n^3)$.

3 Mesure expérimentale de la complexité

Le but de ce second exercice est de mesurer de manière expérimentale la complexité de la fonction `tailleMaxRec` et de l'algorithme décrit dans la question 5 de l'exercice précédent. Le choix du langage de programmation est libre. La complexité expérimentale de l'exécution d'une fonction correspond ici au temps qu'il faut pour que la fonction s'exécute.

Dans cet exercice, n désigne la taille de la séquence initiale.

3.1 Question 1

Voici le code des fonctions `e`, `maximum`, `tailleMaxRec` et `tailleMaxIter`.

```
def e(a,i,j):
    """ str x int x int -> bool
    rend True si i et j peuvent Ãªtre couplÃ©s dans a, False
       sinon
    """
    # couples : list[tuple[str]]
    couples = [('A','T'),('A','T'),('G','C'),('C','G')]
    if (a[i-1],a[j-1]) in couples:
        return True
    return False

def maximum(a,i,j):
    """ str x int x int -> int
    rend le nombre de couples de la structure secondaire de
       taille maximale obtenue pour l'un des partitionnements
       de la sous-sÃ©quence a_{i,j}
    """
    # maxi : int
    maxi = 0
    # k : int
    for k in range(i,j):
        # val : int
        val = tailleMaxRec(a,i-1,k-1)+tailleMaxRec(a,k,j-1)
        if val > maxi:
            maxi = val
    return maxi

def tailleMaxRec(a,i,j):
    """ str x int x int -> int
    rend le nombre de couples de la structure secondaire de
       taille maximale de la sous-sÃ©quence a_{i,j}
    """
    if j-i < 1:
        return 0
    return max(tailleMaxRec(a,i+1,j-1) + e(a,i,j), \
                            maximum(a,i+1,j+1))
```

```
def tailleMaxIter(a,i,j):
    """ str x int x int -> list[list[int]]
    rend un tableau Ã  2 dimensions contenant le nombre de
       couples de la structure secondaire de taille maximale de
        chaque sous-sequence a_{k,l} avec i≤k,l≤j et k≤l
    """
    # n : int
    n = len(a)
    # E : list[list[int]]
    E = [[0]]
    # i : int
    for i in range(2,n+1,1):
        E.append([0]*i)
    # p : int
    for p in range(1,n):
        for i in range(1,n-p+1):
            E[i-1].append(max(E[i][i+p-2]+e(a,i,i+p), \
                    max(E[i-1][k-2]+E[k-1][i+p-1] \
                        for k in range(i+1,i+p+1))))
    return E[i-1][j-1]
```

Voici un jeu d'essai.

3.2 Question 2

Voici le code de la fonction `SeqAleatoire(n)` qui renvoie une séquence d'ADN aléatoire de taille n.

```
import random

def SeqAleatoire(n):
    """ int -> str
    rend une sÃ©quence alÃ©atoire d'ADN de taille n
    """
    seq = ''
    for i in range(n):
        seq += random.choice('AGCT')
    return seq
```

3.3 Question 3

La plus grande valeur de n que l'on peut traiter sans problème de mémoire ou de temps d'exécution de quelques minutes pour la fonction `tailleMaxRec` est 18.

Quant à `tailleMaxIter`, la valeur de n correspondante est 1600.

3.4 Question 4

On a mis dans le tableau ci-dessous le temps d'exécution de plusieurs appels à la fonction `tailleMaxRec` selon les valeurs de n.

n	$CRec(n)$ (en s.)	$\log CRec(n)$
11	0.0915572	-1.0383075
12	0.3087260	-0.5104268
13	0.9356748	-0.0288751
14	3.0208589	0.4801304
15	9.7992974	0.9911949
16	30.8693360	1.4895273
17	100.5011405	2.0021710
18	322.5870859	2.5086470

On en déduit la droite de régression de $\log CRec(n)$ en fonction de n : $\Delta : y = 0.505452x - 6.592292$. On remarque que la pente de la droite est 0.505452, ce qui était attendu car la valeur est comprise entre $\log 2 = 0.301030$ et $\log 4 = 0.602060$.

3.5 Question 5

On a mis dans le tableau ci-dessous le temps d'exécution de plusieurs appels à la fonction `tailleMaxIter` selon les valeurs de n.

n	$CIter(n)$ (en s.)	$\frac{CIter(n)}{n^3}$
200	0.5192115	$6.490\,143\,4 \times 10^{-8}$
400	4.1114257	$6.424\,102\,6 \times 10^{-8}$
600	14.2562266	$6.600\,104\,9 \times 10^{-8}$
800	34.8566699	$6.807\,943\,3 \times 10^{-8}$
1000	71.8084174	$7.180\,841\,7 \times 10^{-8}$
1200	124.3441816	$7.195\,843\,8 \times 10^{-8}$
1400	200.7089246	$7.314\,465\,2 \times 10^{-8}$
1600	314.0727653	$7.667\,792\,1 \times 10^{-8}$

On en déduit la droite de régression de $\frac{CIter(n)}{n^3}$ en fonction de n : $\Delta : y = 8.842\,541 \times 10^{-12}x - 6.164\,326 \times 10^{-8}$. On remarque l'ordre de grandeur de la pente de la droite : $8.842\,541 \times 10^{-12} \ll 1$. Ceci implique que la fonction est bien constante de valeur $6.960\,154\,6 \times 10^{-8}$.