Section 5.3 et 5.4

1. (a)

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 1 \\ f(3) &= 1 \\ f(4) &= 1 \\ &\vdots \end{aligned}$$

(b)

$$\begin{aligned} f(1) &= 20 \\ f(2) &= 10 \\ f(3) &= 5 \\ f(4) &= 16 \\ f(5) &= 8 \\ f(6) &= 4 \\ f(7) &= 2 \\ f(8) &= 1 \\ f(9) &= 4 \\ &\vdots \end{aligned}$$

(c)

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 2 \\ f(3) &= 6 \\ f(4) &= 24 \\ &\vdots \end{aligned}$$

(d)

$$\begin{aligned} f(1) &= 1 \\ f(2) &= 11 \\ f(3) &= 121 \\ f(4) &= 1331 \\ &\vdots \end{aligned}$$

(e)

$$
\begin{aligned}
f(1) &= 1 \\
f(2) &= 2 \\
f(3) &= \frac{1}{2} \\
f(4) &= 4 \\
f(5) &= \frac{1}{8} \\
f(6) &= 32 \\
f(7) &= \frac{1}{256} \\
&\vdots
\end{aligned}
$$

3. (a) Soit S l'ensemble défini récursivement par
   - $2 \in S$
   - Si $x \in S$ et $y \in S$ alors $x + y \in S$

   (b) Soit S l'ensemble défini récursivement par
   - $5 \in S$
   - Si $x \in S$ et $y \in S$, alors $x + y \in S$

   (c) Soit S l'ensemble défini récursivement par
   - $1 \in S$
   - Si $x \in S$ alors $3 \cdot x \in S$

   (d) Soit S l'ensemble défini récursivement par
   - $1 \in S$
   - Si $x \in S$ alors $(\sqrt{x} + 1)^2 \in S$

   On pourrait aussi écrire

   Soit S l'ensemble défini récursivement par
   - $1 \in S$
   - Si $x \in S$ alors $x + 2\sqrt{x} + 1 \in S$

5. L'ensemble S est l'ensemble de paires (x, y) où $x, y \in \mathbb{N}$ et x est un nombre pair et y est un nombre impair ou x est un nombre impair et y est un nombre pair.
7. Ici, $A[1...n]$ est un tableau de nombres triés en ordre croissant !

Algo BinarySearch($A[1..n], i, j, x$)

if $i = j$ **then**
  if $A[i] = x$ **then**
    return i
  else
    return nil
  end if
else
  if $x \leq A[m]$ **then**
    $index =$ BinarySearch($A[1..n], i, m, x$)
  else
    $index =$ BinarySearch($A[1..n], m + 1, j, x$)
  end if
  return index
end if

Il s'agit d'appeler BinarySearch($A[1..n], 1, n, x$).

On pourrait décrire cet algorithme un peu plus judicieusement. En effet, on pourrait d'abord vérifier si $A[m] = x$ et retourner m immédiatement si c'est le cas. Ici, l'objectif est de suivre le framework général présenté en classe.

— Le cas de base est lorsque $i = j$, donc quand le sous-tableau est de taille 1.
— On coupe le tableau en deux sous-tableaux de tailles égales.
— On fait un seul appel récursif.
— Le merge step ne fait rien.

9. Algo NbUns($S[1..n]$)

if $n = 1$ **then**
  return $S[1]$
else
  $m = \lfloor \frac{m}{2} \rfloor$
  $\ell =$ NbUns($S[1..m]$)
  $r =$ NbUns($S[(m + 1)..n]$)
  $total = \ell + r$
  return total
end if

— Le cas de base est lorsque $n = 1$, donc quand le sous-tableau est de taille 1.

— On coupe le tableau en deux sous-tableaux de tailles égales.

— On fait deux appels récursifs (un pour chaque sous-problème).

— Le merge step fait $total = \ell + r$.

---

11. Algo LengthAlt($S[1..n]$)

if $n = 1$ **then**
  return 1
else
  $m = \lfloor \frac{n}{2} \rfloor$
  $\ell_1 =$ LengthAlt($S[1..m]$)
  $\ell_2 =$ LengthAlt($S[(m+1)..n]$)
  $\ell_3 = 0$
  if $S[m] \neq S[m+1]$ **then**
    • Parcourir S de $S[m+1]$ vers $S[n]$ pour trouver la longueur de la plus longue suite alternée
      qui commence à $S[m+1]$ et mettre ℓ_3 à jour.
    • Parcourir S de $S[m]$ vers $S[1]$ pour trouver la longueur de la plus longue suite alternée
      qui se termine à $S[m]$ et mettre ℓ_3 à jour.
  end if
  $\ell = \max\{\ell_1, \ell_2, \ell_3\}$
  return ℓ
end if

Aux deux endroits endroits on on dit de « parcourir » le tableau pour la longueur de la plus longue suite alternée, il faut écrire une boucle « for ». Dans cette boucle for, on incrémente ℓ_3 de 1 chaque fois que le caractère suivant (ou précédent) dans S est différent. Dès qu'on trouve deux éléments consécutifs identiques, on doit sortir de la boucle for.

— Le cas de base est lorsque $n = 1$, donc quand le sous-tableau est de taille 1. Par défaut, une suite de longueur 1 est considérée alternée.

— On coupe le tableau en deux sous-tableaux de tailles égales.

— On fait deux appels récursifs (un pour chaque sous-problème).

— Le merge step cherche la plus longue suite alternée qui commence dans la première moitié et qui se termine dans la deuxième moitié de S. Ensuite, cette suite est comparée aux deux suites trouvées dans les appels récursifs.

13. Ici, on suppose que le tableau $A[1..n]$ donné en input est tel que $n \geq 2$. L'algorithme retourne deux informations : le plus petit élément et le deuxième plus petit élément de $A[1..n]$.

Algo Deux($A[1..n]$)

if $n = 2$ **then**
    $n_1 = \min\{A[1], A[2]\}$
    $n_2 = \max\{A[1], A[2]\}$
    return (n_1, n_2)
else if $n = 3$ **then**
    $n_1 = \min\{A[1], A[2], A[3]\}$
    Soit n_2 le deuxième plus petit élément de $\{A[1], A[2], A[3]\}$.
    return (n_1, n_2)
else
    $m = \lfloor \frac{n}{2} \rfloor$
    $(n_{1,left}, n_{2,left}) = \text{Deux}(A[1..m])$
    $(n_{1,right}, n_{2,right}) = \text{Deux}(A[(m+1)..n])$
    $n_1 = \min\{n_{1,left}, n_{2,left}, n_{1,right}, n_{2,right}\}$
    Soit n_2 le deuxième plus petit élément de $\{n_{1,left}, n_{2,left}, n_{1,right}, n_{2,right}\}$
    return (n_1, n_2)
end if

Voyez-vous pourquoi on a besoin de retourner ces deux informations ?

- Le cas de base est lorsque $n \leq 3$, donc quand le sous-tableau est de taille au plus 3. Si $n = 2$, c'est facile de trouver le plus petit élément et le deuxième plus petit élément. Si $n = 3$, il est facile de trouver le plus petit élément. Pour le deuxième plus petit élément, il faut comparer les 3 nombres pour trouver le deuxième plus petit. On a besoin d'au plus 3 comparaisons. Voyez-vous pourquoi le cas de base est $n \leq 3$ et non pas $n = 2$?
- On coupe le tableau en deux sous-tableaux de tailles égales.
- On fait deux appels récursifs (un pour chaque sous-problème).
- Le merge step calcule le plus petit élément et le deuxième plus petit élément parmi les 4 candidats $n_{1,left}$, $n_{2,left}$, $n_{1,right}$ et $n_{2,right}$. On peut y arriver en faisant au plus 5 comparaisons.

15. Étant donné deux nombres a et b, on pourrait d'abord calculer $\text{pgcd}(a, b)$ et ensuite retourner $\frac{a\,b}{\text{pgcd}(a,b)}$. À l'exercice précédent, on a vu comment calculer $\text{pgcd}(a, b)$ récursivement.

    Essayons de calculer le ppcm sans passer par le pgcd. Voyons d'abord comment calculer le ppcm directement sans utiliser la récursivité. Les multiples positifs de a sont

$\{a, 2a, 3a, 4a, 5a, ...\}$ et les multiples positifs de b sont $\{b, 2b, 3b, 4b, 5b, ...\}$. Donc le ppcm de a et b est le plus petit nombre qui se trouve dans

$$\{a, 2a, 3a, 4a, 5a, ...\} \cap \{b, 2b, 3b, 4b, 5b, ...\}.$$

On peut donc calculer le ppcm de la façon suivante.

Algo PPCM(a,b)

$x = a$
$y = b$
while $x \neq y$ **do**
    if $x < y$ **then**
        $x = x + a$
    else
        $y = y + b$
    end if
end while
return x

Voyons maintenant comment l'écrire de façon récursive. Si l'input était un tableau de n nombres, ce serait facile de diviser l'input en deux. Ici, l'input ne contient que deux nombres. Le plus simple est de traduire la boucle « While » précédente en un algorithme récursif. On obtient l'algorithme suivant.

Algo PPCMRecur(a,b,x,y)

if $x = y$ **then**
    return x
else
    if $x < y$ **then**
        $answer$ = PPCMRecur(a,b,$x + a$,y)
    else
        $answer$ = PPCMRecur(a,b,x,$y + b$)
    end if
    return $answer$
end if

Il s'agit d'appeler PPCMRecur(a,b,a,b).

— Le cas de base est lorsque $x = y$. Dans ce cas, ça signifie qu'on a trouvé le ppcm de a et b.

— Cet algorithme parcourt les ensembles $\{a, 2a, 3a, 4a, 5a, ...\}$ et $\{b, 2b, 3b, 4b, 5b, ...\}$ pour trouver le plus petit nombre qui appartient à leur intersection. À chaque appel récursif, on exclut un candidat. Il n'y a pas vraiment de merge step.

---

17. Algo Somme($A[1..n]$)

if $n = 1$ **then**
    return $A[1]$
else
    $m = \lfloor \frac{n}{2} \rfloor$
    $left =$ Somme($A[1..m]$)
    $right =$ Somme($A[(m+1)..n]$)
    $total = left + right$
    return $total$
end if

— Le cas de base est lorsque $n = 1$. Il s'agit de retourner le seul élément présent dans le tableau.

— On coupe le tableau en deux sous-tableaux de tailles égales.

— On fait deux appels récursifs (un pour chaque sous-problème).

— Le merge step additionne la somme des nombres dans le sous-tableau de gauche avec la somme des nombres dans le sous-tableau de droite.