Exercice 1. Suite de Fibonacci

Question 1.a – On fixe $q \in \mathbb{N}$. Montrons la formule par récurrence forte sur $p \in \mathbb{N}$:

→ Si $p = 0$: $f_{q+2} = f_{q+1} + f_q = f_1 f_{q+1} + f_0 f_q$

→ Si $p = 1$: $f_{q+3} = f_{q+2} + f_{q+1} = 2f_{q+1} + f_q = f_2 f_{q+1} + f_1 f_q$

→ Sinon pour $p \geqslant 2$, on suppose la formule vraie jusqu'au rang $p-1$. On a alors :

$$\begin{aligned} f_{p+q+2} &= f_{p+q+1} + f_{p+q} \\ &= (f_p f_{q+1} + f_{p-1} f_q) + (f_{p-1} f_{q+1} + f_{p-2} f_q) \\ &= (f_p + f_{p-1}) f_{q+1} + (f_{p-1} + f_{p-2}) f_q \\ &= f_{p+1} f_{q+1} + f_p f_q \end{aligned}$$

Question 1.b – Pour f_{2k+2}, on pose $p = q = k$ dans la formule de la question 1.a :

$$f_{2k+2} = f_{k+1}^2 + f_k^2$$

Pour f_{2k+1}, on pose $p = k-1$ et $q = k$ dans la formule de la question 1.a :

$$f_{2k+1} = f_k f_{k+1} + f_k f_{k-1} = f_k f_{k+1} + f_k(f_{k+1} - f_k) = 2f_k f_{k+1} - f_k^2$$

Pour f_{2k}, on pose $p = q = k-1$ dans la formule de la question 1.a :

$$f_{2k} = f_k^2 + f_{k-1}^2 = f_k^2 + (f_{k+1} - f_k)^2$$

Question 2 – On s'intéresse à A_n l'arbre des appels récursifs lors de l'appel à `fibo_lent1` sur l'entrée $n \in \mathbb{N}$.

→ Au vu des deux cas du filtrage, chaque noeud de A_n a 0 ou 2 fils. A_n est donc un arbre binaire strict.

→ Si on note m_n le nombre de feuilles dans l'arbre des appels récursifs, alors :

$$\begin{cases} m_0 = m_1 = 1 \\ \forall n \in \mathbb{N} : m_{n+2} = m_{n+1} + m_n \end{cases}$$

Donc $m_n = f_n$.

→ Comme A_n est binaire strict, son nombre de noeuds internes est $f_n - 1$ et son nombre de noeuds est $2f_n - 1$.

En conclusion, lors de l'exécution de « `fibo_lent1 n` », le nombre d'appels à `fibo_lent1` est $2f_n - 1$. De plus, chaque appel s'exécute en temps constant (sans compter les appels récursifs). Ainsi :

La fonction `fibo_lent1` s'exécute en temps $\Theta(f_n) = \Theta(\phi^n)$.

Pour `fibo_lent2`, on note :

→ $T(k)$ le temps d'exécution de `aux`.

→ $f(k)$ le temps d'exécution de `aux` sans les appels récursifs.

Alors :

$$\begin{cases} T(0) = f(0) \\ \forall k \geqslant 1 : T(k) = T(k-1) + f(k) \\ f(k) = \Theta(1) \end{cases}$$

Donc :

$$\forall k \in \mathbb{N} : T(k) = \sum_{i=0}^{k} f(i) = \Theta(n)$$

Ainsi :

La fonction `fibo_lent2` s'exécute en temps $\Theta(n)$.

Pour `fibo_lent3`, on note :

→ $T(n)$ le temps d'exécution de `fibo_lent3`.

→ $f(n)$ le temps d'exécution de `fibo_lent3` sans les appels récursifs.

Alors :

$$\begin{cases} T(0) = f(0), T(1) = f(1), T(2) = f(2) \text{ sont des constantes.} \\ \forall n \geqslant 3 : T(n) = T(\lfloor n/2 \rfloor) + T(\lfloor n/2 \rfloor + 1) + f(n) \\ f(n) = \Theta(1) \end{cases}$$

On va utiliser le master theorem. Soit $A = 2$, $K = 2$, $\alpha = \log_2(2) = 1$ et $\beta = 0$. Puisque $\alpha > \beta$, d'après le master theorem, on a $T(n) = \Theta(n^\alpha) = \Theta(n)$.

La fonction `fibo_lent3` s'exécute en temps $\Theta(n)$.

Question 3 – On utilise les formules de la question 1.b et on écrit une fonction auxiliaire qui prend en entrée $n \in \mathbb{N}$ et renvoie (f_n, f_{n+1}).

```
(* 'aux n' renvoie (f_n, f_{n+1}) *)
let fibo n =
  let rec aux n = match n with
    | 0 -> 1, 1
    | n -> let fk, fkp1 = aux (n/2) in
           if n mod 2 = 0
           then fk*fk + (fkp1 - fk)*(fkp1 - fk), 2*fk*fkp1 - fk*fk
           else 2*fk*fkp1 - fk*fk, fkp1*fkp1 + fk*fk in
  fst(aux n);;
```

La complexité de cette fonction vérifie :

$$\begin{cases} T(0) \text{ est fixé} \\ \forall n \geqslant 1 : T(n) = T(\lfloor n/2 \rfloor) + f(n) \quad \text{avec } f(n) = \Theta(1) \end{cases}$$

On va utiliser le master theorem. Soit $A = 1$, $K = 2$, $\alpha = \log_2(1) = 0$ et $\beta = 0$. Puisque $\alpha = \beta$, d'après le master theorem, on a $T(n) = \Theta(n^\alpha \log(n)) = \Theta(\log(n))$.

Exercice 2. Tri rapide

Question 1 –

```
let rec diviser pivot li = match li with
  | [] -> [], []
  | e::q -> let li1, li2 = diviser pivot q in
            if e <= pivot
            then e::li1, li2
            else li1, e::li2;;
```

```
let rec tri_rapide li = match li with
  | [] -> []
  | pivot::q -> let li1, li2 = diviser pivot q in
                (tri_rapide li1) @ (pivot::tri_rapide li2);;
```

Question 2 – On note n la taille de la liste en entrée. La complexité pire cas correspond au cas où le pivot est à chaque fois l'élément minimal (ou maximal) de la liste (par exemple si la liste est triée). Dans ce cas, la liste `li1` est vide et la liste `li2` est de taille $(n-1)$. On note :

→ $T(n)$ le temps d'exécution de `tri_rapide` pour une liste triée de taille n.

→ $f(n)$ le temps d'exécution de `tri_rapide` pour une liste triée de taille n sans compter les appels récursifs.

Alors :

$$\begin{cases} T(0) = f(0) \\ \forall n \geqslant 1 : T(n) = T(0) + T(n-1) + f(n) \\ f(n) = \Theta(n) \end{cases}$$

Donc $T(n) = (n+1) \times f(0) + \sum_{k=1}^{n} f(k) = \Theta(n^2)$

Question 3 – On note n la taille de la liste en entrée. La complexité meilleur cas correspond au cas où tous les éléments de la liste sont différents et le pivot est à chaque fois la médiane de la liste. Dans ce cas, les listes `li1` et `li2` sont de taille $\lfloor n/2 \rfloor$ et $\lfloor (n-1)/2 \rfloor$. Si on note $T(n)$ la complexité meilleur cas, on a :

$$\begin{cases} T(0) \text{ est fixé} \\ \forall n \geqslant 1 : T(n) = T(\lfloor n/2 \rfloor) + T(\lfloor (n-1)/2 \rfloor) + f(n) \quad \text{avec } f(n) = \Theta(n) \end{cases}$$

On utilise le master theorem pour trouver l'ordre de grandeur de $T(n)$. Soit $A = 2$, $K = 2$, $\alpha = \log_2(2) = 1$ et $\beta = 1$. On a $\alpha = \beta$ donc d'après le master theorem, $T(n) = \Theta(n \log(n))$.

Question 4 – Les complexités pire cas et meilleur cas du tri fusion sont en $\Theta(n \log n)$. On obtient le même ordre de grandeur dans le meilleur cas mais le tri fusion a un meilleur ordre de grandeur dans le pire cas.

Exercice 3. Nombre d'inversions dans une liste

Question 1 – Les 7 inversions pour la liste `[3; 4; 0; 1; 5; 2]` sont :

`(0,2), (0,3), (0,5), (1,2), (1,3), (1,5), (4,5).`

Question 2 – On a $\texttt{n}_1 = \lfloor \texttt{(n + 1)/2} \rfloor$ et $\texttt{n}_2 = \lfloor \texttt{n/2} \rfloor$.

Question 3 – On a $\texttt{m} = \texttt{m}_1 + \texttt{m}_2 + \texttt{m}'$.

Question 4 –

```
let rec fusion (t1: int list) (t2: int list) (n1: int) =
  match t1, t2 with
  | [], _ -> 0,t2
  | _,[]  -> 0,t1
  | e1::q1, e2::q2 when e1 < e2 ->
     let m_prime, t = fusion q1 t2 (n1-1) in
     m_prime, e1::t
  | t1, e::q2 ->
     let m_prime, t = fusion t1 q2 n1 in
     m_prime + n1, e::t;;
```

Question 5 – La fonction `couper` renvoie deux listes : la première contient les `n` premiers éléments de `li` et la deuxième contient les autres éléments.

```
(* Hypothèse: 0 <= n <= List.length li *)
let rec couper li n = match li,n with
  | li, 0 -> [], li
  | e::q, n -> let li1, li2 = couper q (n-1) in
               e::li1, li2
  | _ -> failwith "Ne devrait pas arriver";;
```

La fonction `aux` prend en argument une liste ainsi que sa taille et renvoie le nombre d'inversions et la liste triée.

```
let nb_inv li0 =
  let rec aux n li = match li with
    | [] -> 0, []
    | [e] -> 0, [e]
    | li ->
       let n1 = (n+1)/2 and
           n2 = n/2 in
       let li1, li2 = couper li n1 in
       let m1, t1 = aux n1 li1 in
       let m2, t2 = aux n2 li2 in
       let m_prime, t = fusion t1 t2 n1 in
       m1 + m2 + m_prime, t
  in
  fst (aux (List.length li0) li0);;
```

Question 6 – Soit $T(n)$ la complexité de la fonction `nb_inv`. On a :

$$\begin{cases} T(0), T(1) \text{ sont fixés} \\ \forall n \geqslant 2 : T(n) = T(\lfloor (n+1)/2 \rfloor) + T(\lfloor n/2 \rfloor) + f(n) \quad \text{avec } f(n) = \Theta(n) \end{cases}$$

D'après le master theorem, $T(n) = \Theta(n \log n)$.

Exercice 4. Tri rapide en place

Question 1 –

```
let swap tab i j =
  let tmp = tab.(i) in
  tab.(i) <- tab.(j);
  tab.(j) <- tmp;;
```

```
let partition tab i j =
  let f = ref (i-1) in
  for k = i to j-1 do
    if tab.(k) <= tab.(j) then begin
        incr f;
        swap tab !f k;
      end;
  done;
  incr f;
  swap tab !f j;
  !f;;
```

Question 2 –

```
let tri_rapide_en_place tab =
  let rec aux i j =
    if j > i then
      let m = partition tab i j in
      aux i (m-1);
      aux (m+1) j;
  in
  aux 0 (Array.length tab - 1);;
```

Exercice 5. Preuve du Master Theorem

Question 1.a – On le montre par récurrence sur p :

- Si $p = 0$ alors $m = n$ et la double inégalité est vérifiée.
- Sinon, on suppose la propriété vraie au rang p et la montre au rang $p+1$. Soit m l'entrée d'un appel récursif de profondeur $p+1$ et m' l'entrée de l'appel récursif précédent de profondeur p. Il existe donc $i \in [\![1; A]\!]$ tel que :

$$m = \frac{m'}{K} + \varphi_i(m') \leqslant \frac{n}{K^{p+1}} + \left[\sum_{k=0}^{p-1} \frac{M}{K^{k+1}}\right] + M = \frac{n}{K^{p+1}} + \left[\sum_{k=0}^{p} \frac{M}{K^k}\right]$$

  Comme $\varphi_i(m) \geqslant -M$, on obtient l'autre inégalité avec le même raisonnement.

Question 1.b – On a :

$$\sum_{k=0}^{p-1} \frac{M}{K^k} \leqslant \sum_{k=0}^{+\infty} \frac{M}{K^k} = \frac{MK}{K-1}.$$

Donc d'après la question précédente, l'entrée m d'un appel récursif à profondeur p vérifie $m \in \left[\frac{n}{K^p} - c_0, \frac{n}{K^p} + c_0\right]$ avec $c_0 = \frac{MK}{K-1}$.

Question 2.a – Soit m l'entrée d'un appel récursif à profondeur p_n alors d'après la question 1.b :

$$m \geqslant \frac{n}{K^{p_n}} - c_0 \geqslant \frac{n}{\frac{n}{c_0+n_0}} - c_0 = n_0$$

De plus, lors d'un appel récursif, l'entrée de `f_rec` est strictement décroissante. Donc à profondeur $p < p_n$, l'entrée est strictement supérieure à n_0.

Question 2.b – Soit m l'entrée d'un appel récursif à profondeur p_n, alors d'après la question 1.b :

$$m \leqslant \frac{n}{K^{p_n}} + c_0 \leqslant \frac{Kn}{\frac{n}{c_0+n_0}} + c_0 = Kn_0 + (K+1)c_0$$

Donc $m_0 = Kn_0 + (K+1)c_0$ convient.

Question 3 – Par hypothèse, il existe $c \geqslant 0$ tel que $f(m) \leqslant cm^\alpha$. On a donc :

$$\begin{aligned} T_1(n) &\leqslant \sum_{p=0}^{p_n-1} cA^p \left(\frac{n}{K^p} + c_0\right)^\beta \\ &\leqslant cn^\beta \sum_{p=0}^{p_n-1} \left(\frac{A}{K^\beta}\right)^p \left(1 + \frac{K^p c_0}{n}\right)^\beta \\ &\leqslant cn^\beta \sum_{p=0}^{p_n-1} K^{p(\alpha-\beta)} \left(1 + \frac{K^{p_n} c_0}{n}\right)^\beta \\ &\leqslant cn^\beta \sum_{p=0}^{p_n-1} K^{p(\alpha-\beta)} \left(1 + \frac{c_0}{c_0+n_0}\right)^\beta \end{aligned}$$

Donc $c_1 = c\left(1 + \dfrac{c_0}{c_0+n_0}\right)^\beta$ convient.

Question 4 – On a :

$$T_2(n) \leqslant A^{p_n} t_0 = K^{\alpha p_n} t_0 \leqslant n^\alpha \frac{t_0}{(c_0+n_0)^\alpha}$$

Donc $c_2 = \dfrac{t_0}{(c_0+n_0)^\alpha}$ convient.

Question 5 – Si on est dans le cas (ii), c'est à dire $\alpha = \beta$ alors :

$$T(n) = T_1(n) + T_2(n) \leqslant c_1 n^\beta p_n + c_2 n^\alpha = \mathcal{O}(n^\alpha \log n)$$

Sinon, lorsqu'on n'est pas dans le cas (ii), on a une somme géométrique et on obtient :

$$T_1(n) \leqslant c_1 n^\beta \frac{K^{p_n(\alpha-\beta)} - 1}{K^{\alpha-\beta} - 1}$$

Si on est dans le cas (i), c'est à dire $\alpha > \beta$ alors $K^{\alpha-\beta} > 1$ donc :

$$T_1(n) \leqslant \frac{c_1}{K^{\alpha-\beta}-1} n^\beta K^{p_n(\alpha-\beta)} \leqslant \frac{c_1}{(K^{\alpha-\beta}-1)(n_0+c_0)^{\alpha-\beta}} n^\beta n^{\alpha-\beta} = \mathcal{O}(n^\alpha)$$

Finalement, $T(n) = T_1(n) + T_2(n) = \mathcal{O}(n^\alpha)$.
Si on est dans le cas (iii), c'est à dire $\alpha < \beta$ alors $K^{\alpha-\beta} < 1$ donc :

$$\frac{K^{p_n(\alpha-\beta)} - 1}{K^{\alpha-\beta} - 1} \leqslant \frac{1}{1 - K^{\alpha-\beta}}$$

Donc :

$$T_1(n) \leqslant \frac{c_1}{1 - K^{\alpha-\beta}} n^\beta = \mathcal{O}(n^\beta)$$

Donc $T(n) = T_1(n) + T_2(n) = \mathcal{O}(n^\beta) + \mathcal{O}(n^\alpha) = \mathcal{O}(n^\beta)$