DM 2 :

1 Exemles de complexité

1.1 Un premier exemple

```
let rec min_et_max liste =
  match liste with
  | [] -> failwith "liste vide"
  | tete :: [] -> (tete, tete)
  | tete :: queue ->
    (min tete (fst (min_et_max queue)),
     max tete (snd (min_et_max queue)))
;;
```

Soit $n \in \mathbb{N}^*$.

1. queue est une liste de taille $n-1$ donc min_et_max queue a une comlexité valant $T(n-1)$.
   min, max, fst et snd sont en $\Theta(1)$.
   Ainsi : $\boxed{T(n) = 2T(n-1) + \Theta(1)}$

2. On calcule cette somme géométrique de raison $\frac{1}{2}$.
   $\sum_{k=1}^{n} \frac{1}{2^k} = 1 \cdot \frac{1 - 2^{-n-1}}{1 - 2^{-1}} = 2 - \frac{1}{2^n} \longrightarrow 2.$
   Toute suite convergente étant bornée : $\boxed{\sum_{k=1}^{n} \frac{1}{2^k} = \Theta(1)}$

3. Soit $k \in \mathbb{N}$: $u_k = \frac{T(k)}{2^k}$.
   Soit $n \in \mathbb{N}^*$, on sait que : $T(k) = 2T(k-1) + \Theta(1)$
   En divisant pat $2^k > 0$: $u_k = u_{k-1} + \Theta\left(\frac{1}{2^k}\right)$.
   Ainsi : $\boxed{u_k - u_{k-1} = \Theta\left(\frac{1}{2^k}\right).}$

4. En sommant la relation précédente : $\sum_{k=1}^{n} u_k - u_{k-1} = \sum_{k=1}^{n} \Theta\left(\frac{1}{2^k}\right)$.
   Par télescopage et avec la question 2 : $u_n - u_0 = \Theta(1)$ donc $u_n = \Theta(1)$ puis $\boxed{T(n) = \Theta(2^n)}$

5. Avec min_et_max [1; 2; 3; 4] :
   On appelle :
   min_et_max [1; 2; 3; 4]
   min_et_max [2; 3; 4]
   min_et_max [3; 4]
   min_et_max [4]
   min_et_max [4]
   min_et_max [3; 4]
   min_et_max [4]
   min_et_max [4]
   min_et_max [2; 3; 4]
   min_et_max [3; 4]
   min_et_max [4]
   min_et_max [4]
   min_et_max [3; 4]
   min_et_max [4]
   min_et_max [4]
   15 appels.
   On obtient en fait un arbre d'appels récursifs, **complet**, de profondeur 3. Chaque noeud interne possède

deux fils identiques. **Le nombre de noeuds est $2^4 - 1 = 15$.**
On obtient une complexité exponentielle puisqu'à chaque étape le nombre d'appels récursifs double.

6.
```
let rec min_et_max liste =
  match liste with
  | [] -> failwith "liste vide"
  | tete :: [] -> (tete, tete)
  | tete :: queue ->
     let min_q, max_q = min_et_max queue in
     (min tete min_q),
     (max tete max_q)
;;
```

L'équation de récurrence devient alors : $C(n) = C(n?1) + \Theta(1)$ dont la solution est bien une complexité linéaire.

1.2 Complexité du tri fusion

1.
```
let rec division liste =
  match liste with
  | [] ->[] ,[]
  | tete :: [] -> liste, []
  | tete1 :: tete2 :: queue ->
   let liste1, liste2 = division queue in
  tete1 :: liste1, tete2 ::liste2
 ;;
division [5;6;9;5;3;8;9];;

let rec fusion liste1 liste2 =
  match liste1,liste2 with
  | _, [] -> liste1
  | [], _ -> liste2
  | tete1 :: queue1, tete2 :: queue2 ->
    if tete1 < tete2 then tete1 :: fusion queue1 liste2
    else tete2 :: fusion liste1 queue2
;;
fusion [1; 4; 7; 8; 9] [2; 3; 5; 8];;

let rec tri_fusion liste =
  match liste with
  | [] -> []
  | tete :: [] -> liste
  | tete :: queue ->
   let liste1, liste2 = division liste in
  fusion (tri_fusion liste1) (tri_fusion liste2)
  ;;
tri_fusion [4; 7; 4; 9; 6; 2; 7; 9; 18];;
```

2. La compléxité de division est en $\Theta(n)$.
La compléxité de fusion est en $\Theta(n_1 + n_2)$.

3. Pour $n \in \mathbb{N}$. Notons $c_n = C(division(n)) + C(fusion(n))$ le coût dans le pire cas cumulé de la division et de la recombinaison d'une entrée de taille n. On a bien $c_n \geqslant 0$ et on a vu que $c_n = \Theta(n)$.
On montre sans difficulté que $(C(division(n))_{n\in\mathbb{N}^*}$ et $(C(fusion(n))_{n\in\mathbb{N}^*}$ sont croissantes et donc$(c_n)_{n\in\mathbb{N}^*}$ également. Pour une liste de taille $n \geqslant 2$, on entre dans le troisième cas du filtrage de la fonction tri_fusion, dont le coût au pire est celui de la division, de la fusion et de deux appels récursifs sur deux listes de tailles $\lceil \frac{n}{2} \rceil$ et $\lfloor \frac{n}{2} \rfloor$.
On a alors bien une équation de récurrence de la forme : Soit $n \in \mathbb{N}$ tq $n > 1$.

$T(n) = \underbrace{c_n}_{\text{division et fusion}} + 1 \quad \underbrace{T\left(\left\lceil \frac{n}{2} \right\rceil\right)}_{trifusion\ liste1} + 1 \quad \underbrace{T\left(\left\lfloor \frac{n}{2} \right\rfloor\right)}_{trifusion\ liste2}$

On a bien aussi $a = 1 \in \mathbb{N}$, $b = 1 \in \mathbb{N}$ et $a + b \geqslant 1$

4. Soit $n = 2^k \in \mathbb{N}$. $u_k = T(2^k)$ et $v_k = \frac{u_k}{2^k}$.
   Il vient : $u_k = \Theta(2^k) + 2u_{k-1}$ puis en divisant par $2^k > 0$: $v_k = \Theta(1) + v_{k-1}$.
   Par télescopage de $v_k = \sum_{j=1}^{k} (v_j - v_{j-1}) + v_0$ on obtient : $v_k = \Theta(k)$

5. On a $k = \log_2(n)$ et $T(n) = T(2^k) = u_k = 2^k v_k = n\Theta(\log_2(n))$ Donc : $T(n) = \Theta(n \ln(n))$

6. $T(1) \geqslant T(0)$
   Supposons pour $n > 1$ fixé : $T(n-1) \geqslant T(n-2 \geqslant \ldots \geqslant T(1) \geqslant T(0)$
   $\left\lceil \frac{n}{2} \right\rceil \in [\![0, n-1]\!]$ et $\left\lfloor \frac{n}{2} \right\rfloor \in [\![0, n-1]\!]$
   Par hypothèse de récurrence : $T\left(\left\lceil \frac{n}{2} \right\rceil\right) \geqslant T\left(\left\lceil \frac{n-1}{2} \right\rceil\right)$ et $T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \geqslant T\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right)$
   Ainsi, en multipliant par $a = 1$ et $b = 1$ positifs et avec (c_n) croissante :
   $aT\left(\left\lceil \frac{n}{2} \right\rceil\right) + bT\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c_n \geqslant aT\left(\left\lceil \frac{n-1}{2} \right\rceil\right) + bT\left(\left\lfloor \frac{n-1}{2} \right\rfloor\right) + c_{n-1}$
   ie : $T(n) \geqslant T(n-1)$. donc **la suite $(T(n))$ est croissante.**

7. Il existe $k \in \mathbb{N}$ unique ie tel que : $2^k \leqslant n < 2^{k+1}$.
   $k = \lfloor \log_2(n) \rfloor$. Donc $k = \Theta(\ln n)$ et $k + 1 = \Theta(\ln n)$.
   De plus $2^k = \Theta(n)$ et $2^{k+1} = \Theta(n)$
   Par croissance de T :
   $T(2^k) \leqslant T(n) \leqslant T(2^{k+1})$.
   Avec : $T(2^k) \leqslant T(n)$, on obtient que $T(n) = \Omega(T(2^k)) = \Omega(n \ln(n))$
   Avec : $T(2^{k+1}) \geqslant T(n)$, on obtient que $T(n) = O(T(2^{k+1})) = O(n \ln(n))$
   Ainsi : $T(n) = \Theta(n \ln(n))$

8. Le tri fusion est plus performant car les tris par insertion et par sélection ont une complexité en $\Theta(n^2)$. Remarquons que cette complexité est bien, bien meilleure. Par exemple, pour une liste de taille 10^6 sur un ordinateur à un milliard d'opérations par seconde on passe de 1/4 h (pour le tri par sélection ou insertion) à 10ms(pour le tri fusion) !

2 Problème : Représentation d'images par des arbres quaternaires