ÉLÉMENT MAJORITAIRE

Dans une liste de longueur n, un élément est dit majoritaire si et seulement si son occurrence (*i.e.* son nombre d'apparitions) est strictement supérieur à $n/2$. On se propose d'écrire une algorithme récursif utilisant la méthode « diviser pour régner » qui détermine l'élément majoritaire d'une liste ainsi que son occurrence, s'il existe.

Question 1) Écrivez une fonction `occurrence l x` de type `'a list -> 'a -> int` qui donne l'occurrence dans une liste `l` d'un objet `x`.

Question 2)

a) Justifiez brièvement l'unicité d'un élément majoritaire.

b) Écrivez une fonction naïve qui répond à la question et précisez sa complexité en fonction de n.

Question 3) Étant donné une liste ℓ de longueur n, on la coupe en deux sous-listes ℓ_1, ℓ_2 de longueurs à peu près égales à $n/2$ et on suppose connaître dans chacune d'eux un élément majoritaire a_1, a_2 et sa fréquence f_1, f_2 respectivement.

a) Montrez que si ℓ a un élément majoritaire, alors ce ne peut être que a_1 ou a_2.

b) Que peut-on conclure si $a_1 = a_2$?

c) Dans le cas contraire, comment savoir si l'un des deux est majoritaire dans ℓ ?

Question 4) Et si l'une des deux listes ℓ_1, ℓ_2 a un élément majoritaire et l'autre non ? Et si aucun des deux n'a d'élément majoritaire ?

Question 5) Écrivez une fonctions récursive `partage l`, qui partage la liste `l` en deux listes de longueurs égales à 1 prés.

Question 6) Écrivez une fonction récursive `majoritaire l` de type `'a list -> 'a * int`, qui donne comme résultat un couple `(?,0)` si la liste n'a pas d'élément majoritaire (remarque ; `?` symbolise un objet arbitraire que vous choisirez) ou le couple `(x,f)` où `x` est l'élément majoritaire et `f` sa fréquence.

Question 7) Quelle est la complexité d'un tel algorithme ?

Corrigé

Question 1)

```
let frequence t x =
        let n = vect_length t and c = ref 0 in
        for i = 0 to (n - 1) do
                if t.(i) = x then c := !c + 1
        done;
        !c;;
```

Sans difficulté, cet algorithme est de complexité linéaire (avec comme paramètre de taille la longueur du tableau, bien sûr).

Question 2)

a) S'il existe deux éléments majoritaires, alors la somme de leurs fréquences dépasse la capacité du tableau, autrement dit le tableau contient plus d'éléments que de cases disponibles : c'est contradictoire.

b) On parcourt le tableau jusqu'à trouver un élément majoritaire : à chaque étape, on parcourt tout le tableau pour calculer la fréquence de l'élément en cours en comptant son nombre d'apparitions avec une variable accumulateur. On parcourt donc au plus n fois le tableau et chaque étape est un parcours complet du tableau avec des opérations élémentaires, donc la complexité est quadratique.

Question 3)

a) On suppose que t_1 est de longueur $n/2$ et t_2 de longueur $(n+1)/2$ (notation CAML pour la partie entière de $\frac{n}{2}$ et $\frac{n+1}{2}$).

Si t a un élément majoritaire a qui n'est ni a_1, ni a_2, alors sa fréquence f vérifie $f > n/2$ donc $f \geqslant n/2\ +1$. Or comme a_1 est majoritaire dans t_1, on a $f_1 > (n/2)/2$ donc $f_1 \geqslant (n/2)/2\ +1$ et de même $f_2 \geqslant ((n+1)/2)/2\ +1$.

Donc à eux trois, les éléments a_1, a_2, a apparaissent $g = f + f_1 + f_2$ fois dans le tableau t, sachant que $g \geqslant n/2\ + (n/2)/2\ + ((n+1)/2)/2\ + 3$.

En notant $n = 4k + r$ avec $r \in \{0, 1, 2, 3\}$, on a les quatre cas possibles :

r	$n/2$	$(n/2)2$	$(n+1)/2$	$((n+1)/2)/2$	$g \geqslant ...$
0	$2k$	k	$2k$	k	$4k+3$
1	$2k$	k	$2k+1$	k	$4k+4$
2	$2k+1$	k	$2k+1$	k	$4k+4$
3	$2k+1$	k	$2k+2$	$k+1$	$4k+5$

Dans tous les cas, on constate que $g > n$, ce qui est contradictoire.

Donc s'il y a un élément majoritaire dans t, ce ne peut être que a_1 ou a_2.

b) Si $a_1 = a_2$, alors la fréquence f de a_1 dans t est $f = f_1 + f_2$,
et on sait que $f \geqslant (n/2)/2\ + ((n+1)/2)/2\ + 2$: on vérifie comme dans la question précédente que $f > n/2$, donc a_1 est majoritaire dans t.

c) Si $a_1 \neq a_2$, alors on calcule la fréquence g_1 de a_1 dans t_2 et celle g_2 de a_2 dans t_1 : si $f_1 + g_1$ ou $f_2 + g_2$ est strictement supérieur à $n/2$, alors l'élément correspondant est majoritaire dans t, sinon t n'a pas d'élément majoritaire.

Question 4) Si t_1 a un élément majoritaire a_1 et que t_2 n'en a pas, alors seul a_1 peut être majoritaire dans t (même raisonnement que précédemment). On calcule alors sa fréquence g_1 dans t_2 et on regarde si $f_1 + g_1 > n/2$.

De même, bien sûr, si t_2 a un élément majoritaire et l'autre non.

Enfin, si aucun des deux tableaux t_1 et t_2 n'a d'élément majoritaire, alors t n'a aucun élément majoritaire.

Question 5)

Question 6)

```
let coupe t =
        let n = vect_length t in
        let k = n / 2 in
        sub_vect t 0 k, sub_vect t k (n - k);;

let rec majoritaire t =
  let n = vect_length t in
    (* cas de base : n = 1 ou n = 2 *)
  if n = 1 then t.(0),1
  else if n = 2 then (if t.(0) = t.(1) then t.(0),2 else t.(0),0)
    (* cas g\'en\'eral r\'ecursif *)
  else  begin
      (* division : partage en deux sous-tableaux *)
    let debut, fin = coupe t in
      (* r\'egne *)
    let maj_d, freq_d = majoritaire debut
            and maj_f, freq_f = majoritaire fin in
      (* fusion *)
      (* cas : pas de majoritaire dans les sous-tableaux *)
    if freq_d = 0 && freq_f = 0 then t.(0),0
      (* cas : m\^eme majoritaire dans les sous-tableaux *)
    else if maj_d = maj_f && freq_d > 0 && freq_f > 0
         then maj_d, (freq_d + freq_f)
      (* cas : un majoritaire diff\'erent dans chaque sous-tableau
      (freq_d et freq_f non nulles) ou un majoritaire dans l'un
      et pas dans l'autre (freq_d >0 et freq_f = 0, ou le contraire)
      dans ce cas, on calcule les fr\'equences crois\'ees *)
    else
      (* un majoritaire dans le d\'ebut : on calcule sa fr\'equence dans la fin
      et on en d\'eduit sa fr\'equence totale *)
    if freq_d >= freq_f then
    begin
      let occ_totale_maj_d = freq_d + frequence fin maj_d in
      if occ_totale_maj_d > n / 2 then maj_d, occ_totale_maj_d else t.(0),0
    end
    else
      (* m\^eme chose avec un majoritaire dans la fin *)
    begin
      let occ_totale_maj_f = freq_f + frequence debut maj_f in
      if occ_totale_maj_f > n / 2 then maj_f, occ_totale_maj_f else t.(0),0
    end
end;;
```

Les commentaires expliquent le fonctionnement de l'algorithme, dont la preuve est l'ensemble des réponses aux questions précédentes.

Quant à la terminaison, elle est assurée par le fait que la longueur de chaque tableau dans chaque sous-appel récursif est strictement inférieur à celle du tableau de l'appel, et que les cas de base sont bien présents.

Question 7) Cet algorithme fonctionne selon le principe « diviser pour régner ». En notant n la longueur du tableau paramètre, l'étape de division consiste à le partager en deux sous-tableaux, ce qui se fait en complexité linéaire (simple recopie de n valeurs). L'étape de fusion consiste en quelques tests, quelques additions et éventuellement un appel à la fonction frequence sur un tableau de longueur inférieure à n, donc elle est de complexité linéaire.

On peut appliquer le th. du cours. En notant $C(n)$ la complexité de la fonction majoritaire, on a la relation $C(n) = qC(n/2)+O(n^p)$, où $q = 2 = 2^\alpha$, $\alpha = 1$ et $p = 1$. Comme $\alpha = p$, on a $C(n) = O(n^\alpha \log_2 n) = O(n \log_2 n)$.