Exercice - 2 : ENS Algo 2015 Charlie Brown

Les députés du parlement de Jersey ont **chacun 3 ennemis au plus** dans ladite assemblée. Montrer qu'il existe une partition, en deux blocs, dudit parlement telle que, dans chacune des deux nouvelles structures, chaque député ait **au plus un** ennemi !

Solution.

— *Méthode 2 (M.Guelfi) :*
On peut aussi préférer procéder de la sorte : pour chaque partition en deux blocs $\mathrm{B} \sqcup \mathrm{C}$ de A, on appelle "mesure d'ennemis" de B (resp. C) un nombre qui est initialisé à 0, et qui, pour tout couple (x, y) de députés de B (resp. de C), est incrémenté de 1 si y est un ennemi de x.

La "mesure totale d'ennemis" de $\mathrm{B} \sqcup \mathrm{C}$ est la "mesure d'ennemis" de A + la "mesure d'ennemis" de B.

— Montrons qu'une partition $\mathrm{B}_0 \sqcup \mathrm{C}_0$ dont la "mesure totale d'ennemis" est minimale convient :
En effet : S'il existait un député de[1] B_0 ayant au moins deux ennemis dans B_0, alors il aurait au maximum 1 ennemi dans C_0 (puisqu'il a au plus 3 ennemis dans toute l'assemblée).
En déplaçant ce député dans C_0, la mesure totale d'ennemis de $\mathrm{B}_0 \sqcup \mathrm{C}_0$ diminuerait strictement, ce qui contredirait la minimalité de la mesure d'ennemis de $\mathrm{B}_0 \sqcup \mathrm{C}_0$.

Notons qu'une telle partition existe, puisque l'ensemble des mesures totales d'ennemis des parties de A est fini (puisque de cardinal inférieur à 2^n) et non vide (puisque A est non vide) : il admet donc un minimum.

Reprendre exo le raisonnement, avec une partition : $\{\mathrm{B}_0, \mathrm{B}_0^c\}$, qui minimise la « mesure d'ennemis » de B_0 seule. Est-ce que ça marche ? Qu'est-ce qui change ?

□

Simulation CamL

ENTRÉE :

```
let minimisation_ennemis ennemis =
    let n = vect_length ennemis and
     nombre_de_k k = (it_list(fun x y -> if y=k then x + 1 else x) 0)
    in
  (
    let mesure_ennemis partie1 =
        let m1 = ref 0 and m2 = ref 0 in
        let partie1_ennemis = ref [] and partie2_ennemis = ref [] in
        (
        for k=0 to (n-1) do
            if ((partie1 lsr k) land 1 = 1) then
                partie1_ennemis := ennemis.(k)@(!partie1_ennemis)
            else
                partie2_ennemis := ennemis.(k)@(!partie2_ennemis)
        done;
        for k=0 to (n-1) do
            if ((partie1 lsr k) land 1) = 1 then
                let enplus = (nombre_de_k k (!partie1_ennemis))
                        in m1 := (!m1) + enplus
            else
                let enplus = (nombre_de_k k (!partie2_ennemis))
```

1. sans perte de généralité

```
                        in m2 := (!m2) + enplus
        done;
        (!m1)+(!m2);
        ) in

    let mesure_min = ref (mesure_ennemis 1) and partie = ref 1
        and limite_sup = int_of_float(2.**.(float_of_int n)) in
    for i=2 to limite_sup do
        let m = mesure_ennemis i in
        if m < !mesure_min then
            ( mesure_min := m;
              partie := i;
            )
        else ()
    done;
    let liste_assemblee x =
      let rec aux y liste1 liste2 compteur = match y with
        | 0 -> liste1, liste2
        | _ -> let depute = ((y mod 2)*compteur) in
                if depute>0 then
                        aux (y/2) (depute::liste1) liste2 (compteur+1)
                else    aux (y/2) liste1 (compteur::liste2) (compteur+1)
      in
      (aux x [] [] 1);
       in
           (!mesure_min),(liste_assemblee (!partie))
  );;

let assemblee_test =
[|[7; 2]; [2]; [10]; []; [1]; [6]; [3; 5; 3]; [2; 10; 9]; [5; 8; 10]; [1; 8]|];;

let test = minimisation_ennemis assemblee_test;;
```

SORTIE :

```
#minimisation_ennemis : int list vect -> int * (int list * int list) = <fun>
#assemblee_test : int list vect =
 [|[7; 2]; [2]; [10]; []; [1]; [6]; [3; 5; 3]; [2; 10; 9]; [5; 8; 10];
   [1; 8]|]
#test : int * (int list * int list) = 2, ([9; 8; 7; 2], [6; 5; 4; 3; 1])
```