Exercice 1. Implémentation de l'algorithme de Kruskal

Question 1 – On va utiliser un tableau d'entiers de taille `n`. Chaque composante connexe sera identifiée par un entier compris entre `0` et `n-1`. Pour tout sommet `v`, l'entier `cc.(v)` est le numéro de la composante connexe contenant `v`. Pour fusionner les composantes connexes de numéro `i` et `j`, il suffit de parcourir le tableau `cc` et d'y remplacer toutes les occurrences de `j` par `i`. Voici l'évolution du tableau lorsqu'on applique l'exemple de l'énoncé :

Après ligne 1 : `[|0; 1; 2; 3|]`
Après ligne 2 : `[|0; 1; 2; 0|]`
Après ligne 3 : `[|0; 2; 2; 0|]`
Après ligne 4 : `[|2; 2; 2; 2|]`

```
type compo_connexe = int array;;

let creer (n: int): compo_connexe =
  let f i = i in
  Array.init n f;;

let meme_compo (cc: compo_connexe) (u: int) (v: int) : bool =
  cc.(u) = cc.(v);;

(* Définir la variable i avant la boucle est obligatoire car cc.(v)
   va être modifiée lors de l'exécution de la boucle *)
let fusion (cc: compo_connexe) (u: int) (v: int): unit =
  if meme_compo cc u v then failwith "fusion: même composante connexe";
  let i = cc.(u) in
  let j = cc.(v) in
  for w = 0 to Array.length cc - 1 do
    if cc.(w) = j then cc.(w) <- i
  done;;
```

Question 2.a –

```
let liste_aretes (g: graphe): arete list =
  let li = ref [] in
  let n = Array.length g in
  for s1 = 0 to n-1 do
    for s2 = s1 to n-1 do
      if g.(s1).(s2) <> 0 then begin
          let a = {s1 = s1; s2 = s2; poids = g.(s1).(s2)} in
          li := a :: !li;
        end;
    done;
  done;
  !li;;
```

Question 2.b –

```
let comp (a1: arete) (a2: arete): int = a1.poids - a2.poids;;
```

Question 2.c –

```
let kruskal (g: graphe): graphe =
  let n = Array.length g in
  let t = Array.make_matrix n n 0 in
  let cc = creer n in
  let li = liste_aretes g in
  let li = List.sort comp li in

  let rec aux: arete list ->  unit = function
    | [] -> ()
    | a :: q ->
       if not (meme_compo cc a.s1 a.s2) then begin
           fusion cc a.s1 a.s2;
           t.(a.s1).(a.s2) <- a.poids;
           t.(a.s2).(a.s1) <- a.poids;
         end;
       aux q;
  in

  aux li;
  t;;
```