Feuille de TD N^o 2

Rappelons le principe du parcours en profondeur avec les calculs des "dates de découverte" et des "dates de fin de traitement" :

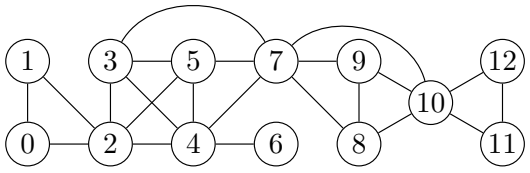
```
visiter_pp(g,s) :
  si dec[s] = -1 :
   dec[s]<-k; k++;
   pour tous les voisin x de s :
      visiter_pp(g,x);
   fin[s]<-k; k++;

visiter(g) :
   k<-0;
   pour chaque sommet s :
      dec[s]<- -1; fin[s]<- -1;
   pour chaque sommet s :
      si dec[s]= -1 :
         visiter_pp(g,s);
```

1 Sommets coupants

Soit $G = (S, A)$ un graphe non orienté connexe. On dit que $s \in S$ est un sommet coupant si le graphe induit par $S \setminus \{s\}$ n'est pas connexe.

1. Déterminer les sommets coupants du graphe suivant :



   Corrigé : 2,4,7,10

2. Décrire un algorithme linéaire qui détermine, étant donné G et s, si s est un sommet coupant ou non.
   Corrigé : on enlève le sommet s de G (se fait en temps linéaire en parcourant chaque liste d'adjacence) puis à l'aide d'un parcours, on teste la connexité. Complexité obtenue : $O(|S| + |A|)$.

3. En déduire un algorithme naïf qui détermine tous les sommets coupants d'un graphe G donné et déterminer sa complexité.
   Corrigé : On applique l'algo de la question précédente à chque sommet : $O(|S|(|S| + |A|))$.

4. Soit T l'arborescence d'un parcours en profondeur de G depuis un sommet s quelconque.
   (a) Montrer que s est un sommet coupant de G si et seulement si le degré de s dans T est de degré au moins 2.
   Corrigé :
   Si $deg_T(s) \leq 1$ alors l'arbre T dans lequel on retire s est connexe et donc G dans lequel on retire s le sera aussi et le sommet n'est clairement pas coupant.
   Si $deg_T(s) \geq 2$ alors soient x et y deux fils de s dans t, montrons que tout chemin dans G entre x et y passe nécessairement par s. Soit $x = s_0 \to s_1 \to \ldots \to s_n = y$ un tel chemin avec $x \in a$ et $y \notin a$ où a est le sous arbre de T enraciné en x. Soit i le plus petit indice tel que $s_i \notin a$ $(1 \leq i \leq n)$, on a $s_{i-1} \in a$ par définition et donc l'arête (s_{i-1}, s_i) relie un sommet de a avec un sommet en dehors de a et comme le parcours en profondeur n'a pas pu donner lieu à une arête transverse (cf lemme de la partie 1) alors la seule possibilité est que $s_i = s$.

   (b) Montrer qu'un sommet $t \in S, t \neq s$ est un sommet coupant de G si et seulement si t possède un fils u tel qu'aucun descendant (dans T) de u n'est voisin (dans G) d'un ancêtre strict (dans T) de t.
   Corrigé : Si t n'admet aucun fils dans T alors T rpivé de t est connexe et donc G privé de t aussi et t n'est pas coupant.
   Si pour chaque fils de t dans T, il existe une arête dans G entre un de ses descendants dans T et un ancêtre strict de t dans T alors on peut montrer que t n'est pas coupant dans G. En effet, si on prend deux sommets x et y dans G alors on peut trouver un chemin dans G entre x et y qui ne passe pas par t.
   Premier cas : si x et y sont tous les deux des ancêtres stricts de t ou dans un même sous arbre "de t" alors clairement un tel chemin existe dans T privé de t et donc dans G privé de t.
   Deuxième cas : si x est dans un sous arbre, nommé a, entraciné en u fils de t et y est un ancêtre strict de t dans T

alors soit (s_1, s_2) une arête avec $s_1 \in a$ et s_2 ancêtre strict de t dont l'existence est garantie par notre hypothèse. On a alors $x \to s_1 \to s_2 \to y$ est un chemin qui convient.
Troisième cas : si x est dans un sous arbre a_1 et y dans un autre sous-arbre a_2 alors en exploitant les deux arêtes reliant chacun de ces sous arbres à un ancêtre strict de t, on pourra de manière analogue construire un chemin dans G entre x et y.

Réciproquement, supposons qu'il existe un fils u de t pour lequel il n'existe aucune arête dans G entre un descendant de u et un ancêtre strict de t. Soit p le père de t dans T qui existe car $t \neq s$. On va montrer que tout chemin dans G entre u et p doit nécessairement passer par t et donc que t est coupant. Notons, a le sous-arbre de T enraciné en u. Soit $u = s_0 \to s_1 \to \ldots \to s_n = p$ un chemin dans G et soit i le plus petit indice tel que $s_i \notin a$, le fait qu'il n'existe pas d'arête transverse ni d'arête vers un ancêtre strict de t depuis un sommet de a, garantit que $s_i = t$.

(c) En déduire un algorithme linéaire qui détermine tous les sommets coupants d'un graphe G donné. (on pourra calculer, pour chaque sommet u, la date de découverte du sommet accessible depuis l'arbre enraciné en u qui a été découvert en premier)
Corrigé :
on fait un pp et on enregistre l'arborescence. On aura besoin de calculer au passage pour chaque sommet : sa date de découverte et son père dans l'arborescence ce qui se fait de manière usuelle mais aussi une quantité $anc[x]$ qui correspond à la plus petite date de fin de découverte d'un sommet y pour lequel il existe un chemin depuis x qui suit les branches de l'arborescence vers le bas et au maximum un arc arrière.
On obtient par exemple le code suivant :

```
(*on suppose que le graphe est connexe donc on se contente de la partie recursive*)

let detecte_articulations g =
    let n = Array.length g in
    let vu = Array.make n false in
    let dec = Array.make n (-1) in
    let pere = Array.make n (-1) in
    let anc = Array.make n (-1) in
    let time = ref 0 in
    let coupants = Array.make n false in
    let rec visiter g s =
        if (not vu.(s)) then
            begin
            vu.(s)<-true;
            anc.(s)<-(!time); dec.(s)<-(!time);
            time:=(!time)+1;
            let rec parcours l = match l with
            |[]->()
            |a::suite when vu.(a)-> if (a<>pere.(s)) then anc.(s)<- min anc.(s) dec.(a); parcours su
            |a::suite-> pere.(a)<- s; visiter g a; anc.(s)<- min anc.(s) anc.(a);
                        if (pere.(s)<>(-1) && anc.(a)>=dec.(s)) then coupants.(s)<-true;
                        parcours suite;



            in parcours g.(s);
            end
    in visiter g 0;
    let deg0 = ref 0 in
    for i = 0 to n-1 do
        if (pere.(i)=0) then
            incr deg0;
    done;
    if (!deg0 >1) then coupants.(0)<-true;
    coupants;;
```