Ecole Polytechnique — Promotion 2005
INF431 — Second Contrôle Classant
Corrigé

P. Jacquet, P. Chassignet, J.-M. Steyaert

4 juillet 2007

Problème 1

1 Diffusion sur étoile

On considère dans cette partie un réseau en forme d'étoile et on étudie certains problèmes de transmission liés à cette configuration

1.1 par lien

On suppose que le poste A relaie un paquet de données $p(X,d,s)$ où X est le poste d'origine du paquet, d désigne les données contenues dans le paquet et s est un numéro de séquence qui associé avec l'adresse de X permet d'identifier de façon unique le paquet. Chaque poste B garde un ensemble *données-reçues*(B) qui contient les paires (A,s) des paquets $p(A,d,s)$ reçus. Le poste B reçoit le paquet $p(X,d,s)$ de A (exprimé par réception $= (A,p(X,d,s))$) et fait *données-reçues*$(B) \leftarrow$ *données-reçues*$(B) \cup \{(X,s)\}$ puis renvoie à A un paquet d'acquittement $\text{ack}(X,s)$ (matérialisé par la commande `envoyer`$(A,\text{ack}(X,s))$) en exécutant le programme suivant:

tant que (**vrai**)ACQUITTEMENT(B)
procédure ACQUITTEMENT(B)
si réception $= (A,p(X,d,s))$
alors $\begin{cases} \textbf{si } (X,s) \notin \textit{données-reçues}(B) \textbf{ alors } \textit{données-reçues}(B) \leftarrow \textit{données-reçues}(B) \cup \{(X,s)\} \\ \texttt{envoyer}(A,\text{ack}(X,s)) \end{cases}$

On remarquera que le poste B retransmet un acquittement même quand il a déjà reçu le paquet (le premier acquittement a pu se perdre).

Le poste A retransmet le paquet $p(X,d,s)$ tant qu'il ne reçoit pas l'acquittement $\text{ack}(X,s)$ de B. Il garde la connaissance des paquets acquittés dans l'ensemble *acquittements-reçus*(A) qui contient les triplets (B,X,s) des paquets $p(X,d,s)$ acquittés par B. Le poste A espace ces retransmissions dans le temps de maxdelai en exécutant le programme suivant :

procédure TRANSMISSION$(A,p(X,d,s))$
tant que $(B,X,s) \notin$ *acquittements-reçus*(A)
faire
$\begin{cases} \texttt{retransmettre}(B,p(X,d,s)) \\ \textbf{si } (\text{réception} = (B,\text{ack}(X,s))) \\ \textbf{alors } \textit{acquittements-reçus}(A) \leftarrow \textit{acquittements-reçus}(A) \cup \{(B,X,s)\} \end{cases}$

où `retransmettre` est la procédure qui espace les retransmissions du paquet tous les maxdélai (paramétrée à l'ouverture du socket).

Le canal est aussi fini, c'est à dire que le paquet finit toujours par passer au bout d'un nombre fini de retransmissions de copies. Ou, en d'autres termes, si l'émetteur émet un nombre infini de copies du même paquet, alors ce paquet arrive un nombre infini de fois.

Question 1 *Montrer que sous l'hypothèse du canal fini, le poste B finit par recevoir le paquet* $p(X, d, s)$ *et le poste A par recevoir le paquet ack*(X, s).

Solution 1 Si le poste A ne reçoit pas l'aquittement ack(X, s) alors il va ré-emttre indéfiniment le paquet $p(X, d, s)$, ceui-ci sera donc reçu un nombre infini de fois. Dans ce cas le paquet ack(X, s) sera transmis un nombre infini de fois et reçu un nombre infini de fois.

1.2 par groupe

Dans cette section on suppose que le poste A est un poste de base et qu'il existe autour n postes attachés $B_1, B_2, \ldots, B_n$ qui forment l'ensemble $T(A)$ des postes attachés à la base A. On suppose que $A \notin T(A)$.

On se met aussi dans les conditions des réseaux sans fil, à savoir que les n liens vers les n postes attachés sont en fait un seul lien, par exemple une fréquence radio en diffusion. Lorsque le poste A diffuse une copie du paquet $p(X, d, s)$ sur ce canal, les n postes attachés sont susceptibles de le recevoir. Dans ce cas on remplace la procédure `envoyer`$(B, p(X, d, s))$ par la procédure `émettre`$(p(X, d, s)$ qui ne précise pas les postes destinataires. L'ensemble des postes attachés qui reçoivent effectivement la copie est un sous-ensemble indéterminé de l'ensemble $T(A)$.

Le poste A doit transmettre le paquet $p(X, d, s)$ vers tous ses postes attachés. Les postes attachés gardent le même pseudo-code réception de la section précédente:

procédure ACQUITTEMENT(B)
si (réception $= (A, p(X, d, s))$ **et** $B \in T(A)$)
alors `émettre`(ack(X, s))

Question 2 *En effectuant le suivi de l'ensemble attente*(A, X, s) *des postes attachés qui ont acquitté le paquet* $p(X, d, s)$*, donner le pseudo-code du programme diffusion*$(A, p(X, d, s))$ *suivi par la base A pour transmettre le paquet* $p(X, d, s)$ *à tous ses postes attachés. On utilsera la procédure* `reémettre` *qui espace les retransmissions du paquet tous les maxdélai, paramétré à l'ouverture du socket.*

Solution 2 **procédure** DIFFUSION$(A, p(X.d, s))$
attente$(A, X, s) \leftarrow T(A)$; **tant que** (attente$(A, X, s) \neq \emptyset$)
faire $\begin{cases} \textbf{si } (\text{réception} = (B, \text{ack}(X, s))) \textbf{ alors } \text{attente}(A, X, s) \leftarrow \text{attente}(A, X, s) - \{B\} \\ \texttt{reémettre}(p(X, d, s)); \end{cases}$

Question 3 *On suppose que le canal de communication est instantané, c'est à dire qu'une transmission arrive instantanément sans erreur au récepteur ou n'arrive pas du tout. On ne considère pas les phénomènes de collision : un poste peut recevoir plusieurs paquets simultanément. On suppose que* $T(A) = \{B_1, B_2\}$*.* B_1 *reçoit les copies de* $p(A, d, s)$ *émises aux rangs impairs (1, 3, etc.), et B2 les copies émises aux rangs pairs (2, 4, etc.). Les acquittements ne sont pas perdus. Combien de copies le poste A va-t-il transmettre?*

Solution 3

A émet	A reçoit	B1 émet	B1 reçoit	B2 émet	B2 reçoit
p(A,d,s)			p(A,d,s)		
	ack(B1,s)	ack(B1,s)			
p(A,d,s)					p(A,d,s)
	ack(B2,s)			ack(B2,s)	

Question 4 *Montrer que dans l'hypothèse du canal fini, les postes attachés finissent par recevoir chacun une copie du paquet* $p(A, d, s)$*, et que la base reçoit un acquittement de chacun de ses postes attachés.*

Solution 4 Supposons que le poste A ne reçoit pas l'acquittement a(B,s) d'un de ses postes attachés B. Le poste A va ré-emttre indéfiniment le paquet p(A,d,s), celui-ci sera donc reçu un nombre infini de fois par le poste B qui transmettra un nombre infini de fois le paquet ack(B,s) qui sera reçu un nombre infini de fois par le poste A.

2 Ensemble dominant

2.1 Algorithme d'élection

On considère que les postes sont tous identiques et constituent un graphe (V, E), non orienté, dont les sommets sont les postes et les arêtes sont les liens radio qui existent entre eux. On appelle N(A), le sous-ensemble constitué des voisins du poste A. On suppose que le réseau est auto-organisé et que les postes vont élir parmi eux un ensemble de bases. L'ensemble des bases devra constituer un sous ensemble dominant du graphe.

Définition Un ensemble dominant d'un graphe (V, E) est un sous-ensemble $D \subset V$, tel que pour tout sommet x de V qui ne soit pas dans D il existe un voisin de x dans D.

Question 5 *Montrer que V est un ensemble dominant. Montrer que si un sommet A est voisin de tous les autres alors l'ensemble $\{A\}$ est dominant.*

Solution 5 Pour $D = V$: il n'existe pas de sommet dans V qui ne soit pas dans D. Pour $D = \{A\}$, pour tout $x \neq A$, $A \in N(x)$.

On suppose les postes du réseau numérotés de 1 à n: $x_1, \ldots, x_n$. On appelle $U(x_i)$ l'ensemble des voisins de x_i dont l'indice est strictement supérieur à i. On propose l'algorithme suivant d'élection d'un sous-ensemble dominant. Un sommet x_i est dans l'ensemble dominant D si et seulement si une des deux conditions suivantes est satisfaite:

1. $N(x_i) = \emptyset$: le poste x_i n'a pas de voisin;
2. l'ensemble $U(x_i)$ ne constitue pas un sous-ensemble dominant de $N(x_i)$.

Question 6 *Montrer que $x_n \in D$. Montrer que lorsque $x \notin D$, alors $N(x) \neq \emptyset$. Dans quel cas a-t-on $x_1 \in D$?*

Solution 6 Si $N(x_n) = \emptyset$, alors $x_n \in D$, sinon le poste x_n n'ayant aucun voisin de numéro plus grand que n, ceux-ci ne peuvent dominer $N(x_n)$. Donc $x_n \in D$. $x \notin D$ implique $N(x) \neq \emptyset$ par définition. Par contre tous les voisins de x_1 on un numéro supérieur à 1, donc dominent $N(x_1)$. Donc $x_1 \in D$ si et seulement si $N(x_1) = \emptyset$.

On se propose de montrer que l'ensemble D est dominant. Pour cela on introduit l'ensemble F_i constitué des sommets de numéro supérieur ou égal à i. On a $F_1 = V$, $F_n = \{x_n\}$ et $F_{n+1} = \emptyset$. On va prouver par récurrence que si $F_i \cup D$ est dominant pour $i \leq n$ alors $F_{i+1} \cup D$ est dominant. En faisant $i = n - 1$ on aura prouvé que D est dominant puisque $x_n \in D$.

L'hypothèse de récurrence est vraie pour $i = 1$.

Question 7 *Soit $i < n$, on suppose que $F_i \cup D$ est dominant. Montrer que $F_{i+1} \cup D$ est dominant. On regardera plus particulièrement le cas où $x_i \notin D$.*

Solution 7 Si $x_i \in D$ alors $F_{i+1} \cup D = F_i \cup D$. Si $x_i \notin D$ alors $F_{i+1} \cup D = F_i \cup D - \{x_i\}$. Tous les sommets qui ont un voisin dans $F_i \cup D$, l'auront aussi dans $F_{i+1} \cup D$ sauf peut-être si ce sont des voisins de x_i. Mais comme $x_i \notin D$ alors $F_{i+1} \cap N(x_i)$ est un ensemble dominant de $N(x_i)$, donc tous les voisins de x_i ont un voisin dans F_{i+1}. Il reste le cas de x_i lui-même mais qui a un voisin dans F_{i+1} puisque $F_{i+1} \cap N(x_i) \neq \emptyset$. Donc $F_{i+1} \cup D$ est un ensemble dominant.

Question 8 *Ecrire le pseudo-code de la fonction domine qui retourne vrai si le poste A est dans D et faux si le poste n'y est pas. On suppose que le poste connaît ses voisins $N(A)$ et les voisins de ses voisins.*

Solution 8 **procédure** DOMINE(A)
marqués(A) $= \emptyset$
si $N(A) = \emptyset$ **alors renvoyer vrai**
pour chaque $x \in U(A)$

faire $\{$marqués(A) $\leftarrow$ marqués(A) $\cup \{x\} \cup N(x)$
si $N(A) \subset$ marqués(A) **alors renvoyer faux**
sinon renvoyer vrai

2.2 Règle d'attachement

Tout poste du réseau qui n'est pas une base, (c'est à dire qui ne soit pas dans l'ensemble dominant D) doit s'attacher à un seul voisin qui soit dans D. On propose la règle d'attachement suivante: pour tout $x \in V$ tel que $N(x) \neq \emptyset$, on appelle $m(x)$ le voisin de x qui a le plus grand numéro, si $x \notin D$ alors x s'attache à $m(x)$.

Question 9 *Montrer que si $x_i \notin D$ et $m(x_i) = x_j$, alors $j > i$ et $x_j \in D$.*

Solution 9 Si $x_i \notin D$ alors $N(x_i) \neq \emptyset$ et $F_{i+1} \cap N(x_i) \neq \emptyset$ puisqu'il domine $N(x_i)$. Donc $m(x) \in F_{i+1} \cap N(x_i)$ et $j > i$. Supposons que $x_j \notin D$, donc tous les voisins de x_j de rang strictement plus grand dominent $N(x_j)$. Donc $x_i \in N(x_j)$ a un voisin commun avec x_j de rang strictement plus grand que j, ce qui contredit la définition de $m(x_i)$.

En passant $m(x) \in D$ quand $x \notin D$ prouve que D est dominant.

3 Ensemble dominant connecté

À partir de maintenant on suppose que le graphe (V, E) est connexe.

Dans ce qui précède on suppose que les sources des paquets que doivent transmettre les bases du réseaux leur parviennent par un réseau de distribution distinct, comme un réseau filaire classique.

Dans ce qui suit on va supposer que la source des informations à distribuer est dans le réseau lui-même et que seuls les liens du réseau sont utilisés.

On introduit le concept d'ensemble dominant connecté d'un graphe connexe (V, E). Un ensemble C est dit dominant connecté si et seulement si (i)C domine V, (ii) le graphe restreint à C est connexe.

Question 10 *Montrer que l'existence d'un ensemble dominant connecté implique que le graphe (V, C) est connexe.*

Solution 10 L'union de C avec tous les voisinages de ses membres reste connectée. Comme cette union est V, alors (V,E) est connexe.

3.1 Algorithme d'élection

On propose l'algorithme d'élection suivant. Appartient à l'ensemble C tout poste x tel que ces voisins de plus grands numéro (c'est-à-dire $U(x)$) NE constituent PAS un ensemble dominant connectés de $N(x)$. On constate que $D \subset C$.

Question 11 *En s'inspirant de la preuve de la question 7 montrer que C est un ensemble dominant connecté. On montrera que si $x_i \notin C$ alors x_i n'est pas un point d'articulation de $F_i \cup C$.*

Solution 11 On sait que C est dominant puisque $D \subset C$. Donc il suffit de montrer que si $F_i \cup C$ est connexe alors $F_{i+1} \cup C$ est connexe. Si $x_i \notin C$ alors x_i n'est pas un point d'articulation de $F_i \cup C$. Dans le cas contraire $N(x) \cap (F_i \cup C)$ me serait pas connexe, ce qui n'est pas possible puisque $F_{i+1} \cap N(x)$ est connexe et domine $N(x)$ (donc *a fortiori* $N(x) \cap (F_i \cup C)$).

Question 12 *Soit x un sommet qui ne se trouve pas dans C. Montrer que $m(x)$ appartient à C.*

Solution 12 Comme $x \notin C$ alors $x \notin D$, donc $m(x) \in D \subset C$.

3.2 Diffusion d'information

Dans cette section on voit comment utiliser l'ensemble dominant connecté C. Chaque poste x qui n'appartient pas à C désigne $m(x)$ comme attachement. Chaque poste qui appartient à C désigne tous ces voisins dans C comme attachement.

En conséquence si $x \notin C$, alors $T(x) = \emptyset$ et si $x \in C$ alors $T(x) = m^{-1}(x) \cup (C \cap N(x))$.

Si un poste qui n'appartient pas à C désire transmettre une donnée (d, s) il la transmet vers son poste d'attachement, ensuite chacun des membres de l'ensemble dominant connecté la retransmet vers son ensemble attaché en exécutant le programme relais.

Question 13 *Ecrire le pseudo-code du programme acquittement-relais(B) que chaque poste A de l'ensemble C doit exécuter pour que le paquet $p(X, d, s)$ soit propagé à l'ensemble du réseau.*

Solution 13 **procédure** ACQUITTEMENT-RELAIS(B)
si réception $= (A, p(X, d, s))$

$$\textbf{alors} \begin{cases} \textbf{si } (B \in T(A)) \\ \textbf{alors} \begin{cases} \textbf{si } (X, s) \notin \textit{données-reçues}(B)) \\ \textbf{alors} \begin{cases} \textit{données-reçues}(B) \leftarrow \textit{données-reçues}(B) \cup \{(X, s)\} \\ \text{attente}(B, X, s) \leftarrow T(B) \\ \texttt{émettre}(p(X, d, s)) \end{cases} \\ \texttt{émettre}(\text{ack}(X, s)) \end{cases} \end{cases}$$

si réception $= (A, \text{ack}(X, s))$

$$\textbf{alors} \begin{cases} \textbf{si } (A \in T(B))) \textbf{ et } ((X, s) \in \textit{données-reçues}(B)) \\ \textbf{alors } \text{attente}(B, X, s) \leftarrow \text{attente}(B, X, s) - \{A\} \end{cases}$$

pour chaque $(X, s) \in \textit{données-reçues}(B)$
faire si attente$(B, X, s) \neq \emptyset$ **alors** `reémettre`$(p(X, d, s))$

On peut introduire des améliorations, comme par exemple prendre les réceptions du paquet $p(X, d, s)$ comme des acquittements implicites:
si réception $= (A, p(X, d, s))$

$$\textbf{alors} \begin{cases} \textbf{si } (B \in T(A)) \\ \textbf{alors} \begin{cases} \textbf{si } (X, s) \notin \textit{données-reçues}(B)) \\ \textbf{alors} \begin{cases} \textit{données-reçues}(B) \leftarrow \textit{données-reçues}(B) \cup \{(X, s)\} \\ \text{attente}(B, X, s) \leftarrow T(B) \\ \texttt{émettre}(p(X, d, s)) \end{cases} \\ \texttt{émettre}(\text{ack}(X, s)) \end{cases} \\ \textbf{si } (A \in T(B)) \textbf{ et } (X, s) \in \textit{données-reçues}(B) \textbf{ alors } \text{attente}(X, s) \leftarrow \text{attente}(X, s) - \{A\} \end{cases}$$

Question 14 *Sous l'hypothèse du canal fini, montrer que tous les sommets du réseau reçoivent la donnée* (d, s).

Solution 14 Soit S l'ensemble des sommets de C qui reçoivent la donnée (d,s). Si $S \neq C$ et comme C est connecté, alors il existe un poste x dans S et un poste y dans C-S, tel que y est attaché à x. Comme x execute le programme transmission(d,s), alors tous ces postes attachés reçoivent la donnée (d,s). Donc $y \in S$. De la même manière les postes attachés aux membres de C reçoivent tous la donnée (d,s). Comme C est dominant connecté, cet ensemble est V tout entier.

Problème 2

4 Triangulations optimales

On considère un polygone *convexe*, sans aucune régularité a priori, à $n+1$ sommets, nommés $s_0, s_1, \ldots, s_n$, dont les arêtes sont désignées par $a_{0,1}, a_{1,2}, \ldots, a_{n,0}$, en suivant le sens de parcours des sommets voisins. Dans ce problème on étudie les propriétés des triangulations de ce polygone par des triangles dont les sommets sont ceux du polygone et qui vérifient les propriétés suivantes :
1. deux triangles quelconques sont disjoints et ne peuvent partager qu'une arête au plus ;
2. les arêtes d'une triangulation ne peuvent se couper qu'en un sommet du polygone ;
3. chaque triangulation est maximale et on ne peut pas lui ajouter de triangle (tout le polygone est recouvert).

Un exemple est donné à la fin de l'énoncé.

Question 15 : Dans une triangulation d'un polygone à $n+1$ sommets, il y a $n+1$ arêtes qui sont celles du polygones et qu'on va appeler *arêtes externes*. Il y a aussi des arêtes internes qui relient des sommets du polygone non-adjacents : quel est leur nombre ? Combien y a-t-il de triangles ? Justifiez vos réponses.

Solution 15 Raisonnons par récurrence. Pour $n = 2$ le polygone est un triangle et il n'y a aucune arête interne et un seul triangle dans la triangulation. Nous voulons donc montrer que pour $n+1$ sommets et arêtes externes, il y a n triangles et $n-1$ arêtes internes. De plus pour $n > 2$, au moins deux triangles possèdent deux arêtes externes. Ces propriétés sont vraies pour les quadrilatères qui correspondent à $n = 3$.
On suppose la propriété vraie pour $n \leq N$ et on considère un polygone P_N à $N+1$ sommets. Considérons le sommet de numéro 0. Deux cas se présentent :
— soit la triangulation contient le triangle (0,1,N) dont deux arêtes sont externes ; alors il reste à trianguler un polygone P_{N-1} à N sommets, qui possède donc par récurrence $N-2$ arêtes internes et $N-1$ triangles, dont deux ont au moins deux arêtes externes ; un seul de ces deux triangles, au plus, est construit avec (1,N); donc P_N a $N-1$ arêtes internes et N triangles dont deux au moins ont deux arêtes externes ;
— la triangulation contient une arête interne $(0, k)$, $3 \leq k \leq N-2$, sinon on est ramené au cas précédent en 1 ou N ; alors cette arête coupe P_N en deux polygones à N_1 et N_2 sommets, avec $N_1 + N_2 = N + 2$ qui vérifient les propriétés car ils ont au moins 4 sommets chacun ; par simple addition on vérifie qu'il y a $N_1 - 2 + N_2 - 2 + 1 = N - 1$ arêtes internes, $N_1 - 1 + N_2 - 1 = N$ triangles dont 2 au moins possèdent 2 arêtes externes car sur les 4, 2 au plus s'appuyent sur l'arête $(0, k)$.

Question 16 : Montrer qu'à chaque triangulation on peut associer un arbre binaire unique à n feuilles et $n-1$ nœuds internes,et qui vérifie les propriétés suivantes :
1. les feuilles sont étiquetées par les arêtes externes $a_{0,1}, a_{1,2}, \ldots, a_{n-1,n}$ *dans cet ordre de gauche à droite* ;
2. la racine est étiquetée par la dernière arête externe $a_{n,0} = a_{0,n}$, et les autres nœuds internes

sont étiquetés par les arêtes internes de la triangulation $a_{j,k}$, avec $0 \leq j < k-1 < n-1$;
3. les feuilles du sous-arbre de racine $a_{j,k}$, s'il existe, sont exactement :
$a_{j,j+1}, a_{j+1,j+2}, \ldots, a_{k-1,k}$.
Un exemple d'arbre correspondant à une triangulation est donné à la fin de l'énoncé.
Expliquer comment on peut lire les triangles sur l'arbre ainsi construit.
Donner deux exemples de triangulations sur un polygone avec $n = 6$ et construire les arbres associés. Comment se représente le cas où le triangle $(0, 1, 6)$ est dans la triangulation ?

Solution 16 Sur l'arbre, chaque triangle est déterminé par un nœud interne et ses deux fils. À l'exception de l'arête externe $(0, n)$ toutes les feuilles sont étiquetées par des arêtes externes.
Si le triangle $(0, 1, n)$ est dans la triangulation, alors $(0, 1)$ est une feuille dont le père est $(0, n)$ racine de l'arbre et $(1, n)$ est le nœud interne qui est le fils droit de la racine.

Question 17 : Donner un exemple où l'un des triangles est constitué de trois arêtes internes (avec $n = 6$ ou 7). Dessiner l'arbre associé.

Solution 17 Il suffit d'adapter l'exemple de l'énoncé en conservant la structure du sous-arbre droit et le triangle interne $(4, 6, 8)$.

Question 18 : Écrire une classe Java qui décrit la structure d'arbre binaire représentant une triangulation et une méthode qui vérifie que les étiquettes vérifient les règles des questions 1 et 2. Écrire une méthode qui liste les triangles *internes*.

Solution 18

```
class Noeud {
  Noeud g, d;
  int j, k;

  public Noeud(Noeud g, Noeud d, int j, int k) {
    this.g = g;
    this.d = d;
    this.j = j;
    this.k = k;
  }
}

 static boolean estFeuille(Noeud t) {
    if ( t == null )
      return true;
    else
      return t.g == null;  // suffit si on suppose la structure OK
                           // sinon forme classique
                           // t.g == null && t.d == null
  }

  static void listerTriangles(Noeud t) {
    if ( t == null )
      return;
    listerTriangles(t.g);
    if ( !estFeuille(t) )
      System.out.println(t.g.j+" "+t.g.k+' '+t.d.k);
    listerTriangles(t.d);
  }
```

```
static boolean verifie(Noeud t) {
  if ( t == null )
    return false;
  if ( t.j >= t.k )
    return false;
  if ( t.j == t.k-1 )
    return t.g == null && t.d == null;
  if ( t.g == null || t.d == null )
    return false;
  return verifie(t.g) && verifie(t.d)
      && t.j == t.g.j && t.g.k == t.d.j && t.d.k == t.k;
}

 static void listerTrianglesInternes(Noeud t, boolean racine) {
  if ( t == null )
    return;
  listerTrianglesInternes(t.g, false);
  if ( !racine && !estFeuille(t.g) && !estFeuille(t.d) )
    System.out.println(t.g.j+" "+t.g.k+" "+t.d.k);
  listerTrianglesInternes(t.d, false);
}
```

Par la suite on supposera que les structures d'arbres représentant les triangulations sont correctes.

Question 19 : Chaque triangle a un coût qui est la somme des longueurs de ses arêtes : les longueurs sont des nombres réels. Le coût d'une triangulation est alors la somme des coûts de ses triangles.
On suppose que la classe précédente possède une méthode pour calculer la distance (réelle et symétrique) entre deux sommets, du style `Sommet.distance(int j,int k)`. Prendre en compte ce nouveau paramètre et écrire une méthode qui calcule le coût de la triangulation définie par un arbre.

Solution 19

```
static double cout(Noeud t) {
  if ( estFeuille(t) )
    return 0;
  else // suppose structure OK
    return Sommet.distance(t.j, t.k)
         + Sommet.distance(t.g.j, t.g.k)
         + Sommet.distance(t.d.j, t.d.k)
         + cout(t.g) + cout(t.d);
}
```

Question 20 : En s'appuyant sur la question 2 et sur les exemples de l'énoncé et vos exemples, observer que toute triangulation peut être définie comme un processus dichotomique à partir de la dernière arête $a_{0,n}$ qui étiquette la racine de l'arbre qui consiste à définir le sommet k qui déterminera le triangle $0, k, n$ de la triangulation, puis récursivement sur les deux sous-polygones construits sur les sommets $0, ..., k$ et $k, ..., n$. Montrer sur deux exemples comment on décompose un polygone à partir d'un arbre.

Solution 20 Il (faut et il) suffit de parcourir l'arbre dans un ordre quelconque et de tracer les arêtes internes qui sont définies dans les nœuds internes de l'arbre. C'est la réciproque de la question 2.

Question 21 : On veut maintenant calculer la triangulation de coût minimal, ou du moins donner le coût d'une telle triangulation s'il en existe plusieurs.
Donner un schéma logique qui permet de caractériser une telle configuration et proposer un algorithme récursif qui calcule le coût de cette configuration. Estimer la complexité de cet algorithme. [On pourra s'inspirer de la formule de récurrence $TriangOpt(i,j) = \min_{i<k<j}\{TriangOpt(i,k) + TriangOpt(k,j) + CoutTriangle(i,j,k)\}$.]

Solution 21 Le schéma logique consiste donc à appliquer récursivement la formule donnée dans l'énoncé.

```
static double coutMin(int i, int j) {
  if ( i >= j-1 )
    return 0;
  double min = Double.MAX\_VALUE;
  for ( int k = i+1 ; k < j ; ++k ) {
    double c = Sommet.distance(i, j)
             + Sommet.distance(i, k)
             + Sommet.distance(k, j)
             + coutMin(i, k) + coutMin(k, j);
    if ( min > c )
      min = c;
  }
  return min;
}
```

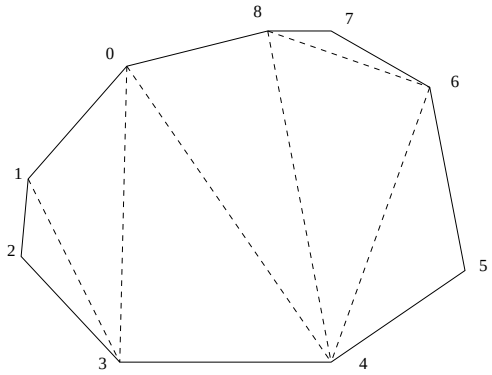
Le coût est exponentiel car il correspond à celui de l'énumération des arbres binaires de taille $N-1$, soit le nombre de Catalan $\frac{1}{N}\binom{2N-2}{N-1}$, qui vaut $O(4^N N^{-3/2})$.

Question 22 : Donner un algorithme polynomial pour résoudre le problème précédent et proposer un code Java pour l'implanter.

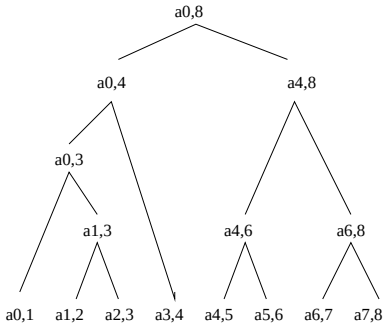
Solution 22

```
static double coutMin(int n) {
  double[][] cMin = new double[n+1][n+1];
  for ( int j = 1 ; j <= n ; ++j )
    for ( int i = j-2 ; i >= 0 ; --i ) {
      cMin[i][j] = Double.MAX\_VALUE;
      for ( int k = i+1 ; k < j ; ++k ) {
        double c = Sommet.distance(i, j)
                 + Sommet.distance(i, k)
                 + Sommet.distance(k, j)
                 + cMin[i][k] + cMin[k][j];
        if ( cMin[i][j] > c )
         cMin[i][j] = c;
      }
    }
  return cMin[0][n];
}
```

Exemple :



Un polygone avec 9 sommets et une de ses triangulations dont les are^tes internes sont en pointille´s



L´arbre correspondant