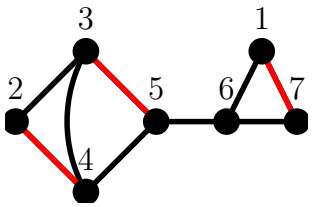
Algorithmes distribuées auto-stabilisants.

Exercice : couplage maximal dans un graphe.

Un `couplage` d'un graphe est un ensemble d'arêtes non-adjacentes, c'est-à-dire telles que chaque sommet du graphe appartient à au plus une arête (voir figure 2). Il est `maximal` s'il ne peut être augmenté comme celui représenté en figure 1.



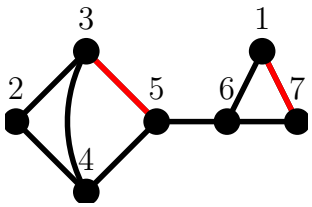


Figure 1 et Figure 2
Les arêtes rouges forment un couplage maximal du graphe.

Question 1 : Trouver un autre couplage maximal du graphe de la figure 1.

Le modèle à lecture d'état est considéré. Chaque processus p possède une variable locale `pref(`p`)` dont la valeur est $\emptyset \cup \Gamma(p)$ avec $\Gamma(p)$ qui est l'ensemble des voisins de p. La valeur `pref(`p`)` désigne avec quel voisin il est associé dans le couplage.

Question 2 : Calculer la valeur `pref` des sommets du graphe de la figure 1.

Nousnous allons définir un état pour chaque processus p.

- Si le processus p a choisi le processus q alors
  - p `attend` si q n'a pas encore choisi
  - p est `apparié` si q a choisi p
  - p est `chaînant` si q a choisi un autre que p
- Si le processus p n'a pas choisi le processus q alors
  - p est `mort` si tous ses vousins sont appariés q n'a pas encore choisi
  - p est `libre` dans le cas contraire

Question 3 : Calculer les états des sommets du graphe G représenté en figure 2.

Considérons la spécification ϕ suivante pour une configuration γ :
$\phi = \forall p\ (etat(p) =$ `apparié` $\vee etat(p) =$ `mort`$)$

Question 4 : Montrer que pour une configuration γ si $\phi(\gamma)$ est vraie alors l'ensemble des arêtes $M = \{(p, pref(p)) : pref(p) \neq \emptyset\}$ est un couplage maximal.

Considérons l'algorithme suivant :

$\mathtt{M_p}$: $\{pref(p) = \emptyset \wedge pref(q) = p\} \rightarrow pref(p) := q$

$\mathtt{S_p}$: $\{pref(p) = \emptyset \wedge (\forall r \in \Gamma(p), pref(r) \neq p) \wedge pref(q) = \emptyset\} \rightarrow pref(p) := q$

$\mathtt{U_p}$: $\{pref(p) = q \wedge pref(q) \neq p \wedge pref(q) = \emptyset\} \rightarrow pref(p) := \emptyset$

Question 5 : Montrer que la configuration γ est terminale[1] si et seulement si $\phi(\gamma)$ est vraie.

Question 6 : L'algorithme atteint une configuration terminale en plus en $O(n^2)$ pas avec n le nombre de processus. *Indication :* Etudier la fonction F définit de la façon suivante : soit γ une configuration possible $F(\gamma) = (c + f = w, 2c + f)$ *avec*

- c le nombre de processus chaînants
- f le nombre de processus libres
- w le nombre de processus en attente

Exercice : élection de leader dans un réseau en général

On suppose ici que chaque processeur a une identité unique allant de 1 à N où N est un majorant du nombre de processeurs noté n (i.e $N \geq n$). L'objectif de cet exercice est d'élire un processeur parmi tous et de propager le résultat de cette élection.

Considérons l'algorithme suivant décrit de façon informelle. Chaque processeur P_i est candidat pour être le leader. Au début, il se désigne comme tel. De façon itérative, P_i communique à ces voisins son identité x du leader. Quand P_i reçoit une identité y d'un autre candidat par ces voisins, si $x > y$, alors P_i choisit y comme son identité du nouveau leader.

Question 1 : Cet algorithme est-il auto-stabilisant ?

Maintenant considérons l'algorithme suivant en supposant que chaque processeur P_i connaît la valeur N. De plus, le processeur P_i définit le leader en fonction de deux variables

- $leader_i$ l'identité du leader
- $distance_i$ la valeur de distance entre lui et le leader.

De façon itérative, P_i lit les variables liées au leader de ces voisins. En fonction de cette lecture, il choisit le leader ayant la plus petite identité et tel que la distance entre lui et le leader soit inférieur à N.

Voici sa description formelle pour le noeud courant P_i :

$true \rightarrow$

1. $\langle candidat, distance\rangle := \langle ID, 0\rangle$
2. pour chaque $P_j \in \Gamma(P_i)$ faire
   (a) $\langle leader[j], distance[j]\rangle := lire(\langle leader_j, distance_j\rangle$
   (b) Si $(distance[j] < N)$ et si $(leader[j] < candidat)$ alors
   $\langle candidat, distance\rangle := \langle leader[j], distance[j] + 1\rangle$
3. écrire $\langle leader_j, distance_j\rangle = \langle candidat, distance\rangle$

Question 2 Prouver que pour chaque exécution équitable qui commence à partir de n'importe quelle configuration, il existe une configuration c à partir de laquelle tous les candidats leaders de chaque processeur correspondent bien une identité d'un processeur.

Question 3 Prouver que pour chaque exécution équitable qui commence à partir de n'importe quelle configuration c, il existe une configuration c à partir de laquelle toutes les configurations suivantes satisfont le prédicat suivant :

`P` : tous les processeurs sélectionnent le processeur ayant la plus petite identité comme leader.

[1] aucune commande dans cette configureation peut être déclenchée