SALAMAT Mohamed
BEN ELHADJ Ali
KEMALI Ismail

DM 2
Algorithme répartie

<u>Question1:</u> Par induction sur k prouvez que si $k=d(q,p)$ **alors** $q \in i_p^{d(q,p)}$

Soit $\varphi : si \ \ k=d(q,p) \ \ alors \ \ q \in i_p^{d(q,p)}$

Montrons par induction que φ est vrai

<u>cas k=0:</u>

On a:

$$i_p^0 = \{(p,p)\} \, et \, d(p,p)=0=k$$

$$d'ou \ \ k=d(q,p)=0 \rightarrow q \in i_p^{d(q,p)} avec (q=p)$$

$$donc \ \ \varphi \ \ vrai \ \ pour \ \ k=0$$

<u>cas k= n+1 (n>0):</u>

On suppose qu'il existe $p' \in \mathrm{In}_p \ \ tel\,que \ \ d(q,p')=n \ \rightarrow \ q \in i_{p'}^{d(q,p')}$ (HR)

À la phase n, et à la ligne 3 de l'algorithme , p' transmet le message (q, u) à p car $p' \in \mathrm{In}_p \ (p \in Out_{p'})$ et pour cela qu'on en déduit que:

$$d(q,p)=d(q,p')+1=n+1 \qquad (1)$$

Ensuit à la phase (n+1) , p a reçu le message (q, u) alors $q \in i_p^{k+1}$ (2)

par conséquence et d'après (1) et (2) on a prouvé que φ est vrai pour k = n+1

Donc nous avons prouvé que φ est vrai pour tout k, p et q.

<u>Question 2:</u> Par induction sur k montrez que si d(q,p) > k alors $q \notin i_p^k$

Soit $\varphi \ : si \ \ d(q,p)>k \ \ alors \ \ q \notin i_p^k$

montrons par induction que φ est vrai

<u>cas k=0:</u>

On a $i_p^0 = \{(p,p)\}$

Soit $q \in \Pi$, on a trivialement d(q, p) > 0 alors d(q, p) > k=0

On voit bien que $q \notin i_p^0 \, et\,que\, d(q,p)>0 \, alors \, \varphi \, est\,vrai\,pour\, k=0$

<u>cas k= n+1 (n>0) :</u>

On suppose qu'il existe $p' \in \mathrm{In}_p \ \ tel\,que\,si \ \ d(q,p')>n \ \rightarrow \ q \notin i_{p'}^n$ (HR)

Comme $p' \in \mathrm{In}_p$ alors $d(q,p)=d(q,p')+1$ d'ou $d(q,p)>n+1$, et cela veut dire que la phase $d(q,p)$ se passe bien après la phase $(n+1)$, donc on en deduit que à la phase $(n+1)$, $q \notin i_p^{n+1}$

Par conséquence φ est vrai pour k = n+1

Donc nous avons démontré que φ est vrai pour tout k, p et q.

<u>Question 3</u>: En déduire que p n'écrit $T_p[q]$ **qu'une seule fois et ce dans la phase égale à d(q, p)**

On sait que $si \ \ k=d(q,p) \ \ alors \ \ q \in i_p d(q,p)$ (1)

et que $si \ \ d(q',p)>k \ \ alors \ \ q' \notin i_p^k$ $(avec\, k=d(q,p))$ (2)

→ on constate d'après (2) que $q' \notin i_p^k$ tant que $d(q',p)>k$, donc p n'écrit pas $T_p[q']$ tant

que le numéro de la phase est inférieure à k.
→ on constate d'après (1) et (2) que p écrit $T_p[q]$ exactement à la phase k pour la première fois, (puisqu'on sait qu'il ne l'a fait durant ces phases précédentes).

→ il reste à prouver que p n'écrit pas $T_p[q]$ dans les phases supérieurs à d(q, p):
Dans l'algorithme à la ligne 9, $\forall (q,x) \in i_p^{d(q,p)}$ | $(q,x) \notin nouveau_p$ donc p n'écrit pas $T_p[q]$ durant les supérieurs à d(q, p).

D'après les trois derniers points on a prouvé que p n'écrit $T_p[q]$ qu'une seule fois et ce dans la phase d(q, p).

<u>Question 4:</u> Montrer par une induction sur d(q, p) que si p écrit $T_p[q]$ = j alors j est la première étape d'un chemin de q à p.

Soit φ : si p écrit $T_p[q]$ = j alors j est la premiere étape d'un chemin de q à p

Montrons par induction que φ est vrai:

<u>cas d(q, p) = 0:</u>

Si d(q, p) = 0 alors la longueur du plus court chemin de p à q égale à 0, d'où p=q .

Si $T_p[q]$ = p alors p est la première étape d'un chemin de p à p ce qui est trivialement vrai.

<u>Cas d(q, p)+1:</u>

On suppose qu'on a d(q, p) tel que si p écrit $T_p[q]$ = j alors j est la première étape d'un chemin de q à p (HR).

Soit $p' \in Out_p$ donc $d(q,p') = d(q,p)+1$, p' écrit $T_{p'}[q]$ = j seulement et exactement à la phase d(q, p') (d'après la question 3), donc puisque p a écrit $T_p[q]$ = j, d'après HR**,** alors j est la première étape d'un chemin de q à p

d(p, p')=1 (car $p' \in Out_p$) d'où j est aussi le premier pas d'un chemin de q à p'.

On a donc démontré φ pour d(q, p)+1

Nous avons démontré par ce qui précède que φ est vrai pour p, q et d(q, p).

<u>Question 5:</u> Montrer que l'algorithme termine (on pourra utiliser les résultats prouvés dans le devoir 1), en déduire que chaque processus p termine avec une table T_p telle que $T_p[q]$ est la première étape d'une route de langueur minimale de q vers p.

On a vu dans le devoir-1 que l'algorithme de phase termine quand chaque processus atteint la phase maximale qui est égale à D. Et comme l'algorithme de routage minimal est un algorithme de phase qui construit une table de routage alors l'algorithme de routage termine aussi.

(1) Dans la question-3 on a montré que p n'écrit $T_p[q]$ qu'une seule fois et ceci à la phase d(q, p).
(2) Dans la question-4 on a montré que à la phase d(q, p) , si écrit $T_p[q]$ = j alors j est la première étape d'un chemin de q à p.

→ Donc d'après (1) et (2), p n'écrit qu'une seule fois $T_p[q]$ et ce à la phase d(q, p) avec $T_p[q]$ la première étape d'un chemin de q à p, mais comme cela se passe à la $d(q,p)^{eme}$ phase alors de q on atteint p en d(q, p) phases et par définition d(q, p) est la langueur du plus court chemin de q à p, d'où $T_p[q]$ est la première étape d'une route de langueur minimale de q vers p.

Le graphe est fortement connexe, chaque processus termine à la phase D et $|D| \geq |\{d(q,p) \mid \forall q,p \in \Pi\}|$ donc chaque processus p termine avec une table T_p telle que $T_p[q]$ est la première étape d'une route de langueur minimale de q vers p

<u>Question 6:</u> Évaluez la complexité en nombre de messages de l'algorithme. En supposant que les identités des processus sont des entiers consécutifs 1...n. Montrer que le nombre de bits échangé dans l'algorithme est en O(e.n.log(n)), (e étant le nombre de liens de communication).

Les messages sont sous une forme de couple de (p, q) avec $p, q \in \{1...n\}$ donc log(n) est la complexité en nombre de bit d'un message.

Nous avons (e.n) messages échangés par un processus d'où une complexité de (e.n) pour le nombre de messages échangés et O(e.n.log(n)) pour le nombre de bite échangés.

<u>Question 7</u>:

1) $i_p := \{\} ; R_p[p] = p ;$
2) $phase_p := 1 ;$
3) **forall** $j \in Out_p\, do$
4)   **forall** $q \in \Pi\, do$
5)   send({(p, Tp[q],q)}) to j;
6) $i_p := i_p U \{(p, T_p[q], q)\} ;$
7) **while** true **do**
8)   **if** tous les liens entrants continent de messages **then**
9) $ancien_p := i_p ;$
10)   consommer un message sur chaque lien entrant
11) $i_p := i_p U \{x \mid x$ lu dans l'instruction ligne 10 }
12) $nouveau_p := \{(x, y, z) \in i_p \mid \nexists (x, y, z) \in ancien_p\}$
13)   **forall** $(x, y, z) \in nouveau_p$ **do**
14)   **if** z = p **then**
15) $R_p[x] := y ; nouveau_p := nouveau_p / \{(x, y, z)\} ;$
16) $phase_p := phase_p + 1 ;$
17)   **if** $phase_p \leq D$ **then**
18)   **forall** $j \in Out_p$ do $send(nouveau_p)$ to j
19)   **else** *termine*

Les message échangé sont de la forme (p, s, q) ce qu'il signifie que si p souhaite envoyer un message à q, il doit le transmettre à s (premier pas).

Nous avons (e.n) messages échangés par processus donc (e.n.log(n)) bit échangés par processus (cf Question 6), donc la complexité en nombre de messages envoyés est de $O(2.\text{e.n}) \simeq O(e.n)$ et la complexité en nombre de bits est $O(2.\text{e.n.log}(n)) \simeq O(e.n.log(n))$

Remarque:

Les deux algorithmes ont le même ordre de grandeur du nombre de messages (et de bits) échangés.