Révisions

1 Communication dans les réseaux

Un Réseau Échange-Mélange (REM) avec $p = 2^r$ processeurs est défini comme suit :
(i) numéroter les processeurs de 0 à $p-1$ et les écrire en binaire sur r bits :

$$q = b_{r-1}b_{r-2}\dots b_2b_1b_0 \text{ avec } b_i \in \{0,1\} \text{ pour } 0 \leq i \leq r-1$$

(ii) pour $q = b_{r-1}b_{r-2}\dots b_2b_1b_0$, définir :

$$\begin{aligned} rot(q) &= b_0b_{r-1}b_{r-2}\dots b_2b_1 \\ exch(q) &= b_{r-1}b_{r-2}\dots b_2b_1(1-b_0) \end{aligned}$$

(iii) pour tout q, $0 \leq q \leq p-1$, relier q à $rot(q)$ et à $exch(q)$ par des arcs orientés.

1. Dessiner un REM à 16 processeurs. Quel est le degré maximal d'un REM à 2^r processeurs ?
2. Proposer un algorithme de routage d'un processeur à un autre dans un REM.
3. Proposer un algorithme de diffusion d'un processeur à tous les autres dans un REM, et donner son temps d'exécution.

2 Ordonnancement sans communication

On veut ordonnancer un ensemble de n tâches indépendantes $T_1, T_2, \dots, T_n$. La durée de T_i est p_i. On suppose que l'ordonnancement débute au temps $t = 0$, et on note C_i la date de la fin de l'exécution de la tâche T_i.

Question 1 On dispose d'un seul processeur dans cette question.

1. Quel est l'ordre d'exécution optimal si l'objectif est de minimiser la somme des dates de fin $\sum_{i=1}^{n} C_i$?
2. Quel est l'ordre d'exécution optimal si l'objectif est de minimiser la somme pondérée des dates de fin, i.e. $\sum_{i=1}^{n} w_i.C_i$, où w_i est un poids positif associé à la tâche T_i ?

Question 2 *(difficile)* On dispose de $p \geq 2$ processeurs identiques dans cette question. Quel est l'ordre d'exécution optimal si l'objectif est de minimiser la somme des dates de fin $\sum_{i=1}^{n} C_i$?

3 Permutation de boucles

On considère un nid de boucle parfait de profondeur n, qui comprend m dépendances uniformes. Soit D la matrice de taille $n \times m$, obtenue à partir du signe des composantes des vecteurs de dépendance. Si d_i est le i-ème vecteur de dépendance, on stocke le signe $+$, 0 ou $-$ de ses composantes dans la i-ème colonne de D. Par exemple si $n = 3$ et $d_1 = (2, -2, 0)^t$, on stockera $(+, -, 0)^t$ dans la première colonne de D.

Question 1 Donner la matrice D_1 pour le nid suivant :

```
for i = 1 to n do
  for j = 1 to n do
    for k = 1 to n do
      S1 : a(i+1, j, k) = a(i, j, k) + 2
      S2 : b(i, j, k+1) = b(i, j, k) - 1
      S3 : c(i+1, j+1, k+1) = c(i, j, k) + 1
    endfor
  endfor
endfor
```

Question 2 Donner la matrice D_2 pour le nid suivant :

```
for i = 1 to n do
  for j = 1 to n do
    for k = 1 to n do
      S1 : a(i, j, k + 1) = a(i, j − 1, k) + a(i − 1, j, k + 2) + 2
    endfor
  endfor
endfor
```

Question 3 Donner un nid dont la matrice D_3 soit

$$D_3 = \begin{pmatrix} + & + & + & 0 & 0 & 0 \\ + & 0 & 0 & + & 0 & 0 \\ 0 & + & 0 & 0 & + & 0 \\ 0 & 0 & + & 0 & 0 & + \end{pmatrix}$$

Question 4 L'algorithme de parallélisation fonctionne ainsi :

- Si une ligne de la matrice D (disons la i-ème) ne contient que des 0, alors on place cette boucle à l'extérieur du nid et on et on l'exécute en parallèle : on permute les boucles, la i-ème boucle devient la première boucle, et on remplace le for par un $forall$.
- Si aucune ligne de la matrice D ne contient que des 0, on sélectionne la ligne qui contient le plus de +, on place la boucle correspondante à l'extérieur et on l'exécute en séquentiel.

On exécute alors récursivement l'algorithme sur les boucles et les dépendances restantes, i.e. qui n'ont pas été satisfaites par la séquentialisation de la boucle.

Par exemple pour D_1, on choisit de séquentialiser soit la première soit la troisième boucle. Si on choisit la première, pas besoin de permuter, la première et troisième dépendances sont satisfaites. Il reste deux boucles à l'intérieur, avec la matrice réduite $\overline{D}_1 = \begin{pmatrix} 0 \\ + \end{pmatrix}$. On en déduit le code parallèle :

```
forseq i
  forpar j
    forseq k
      ...
```

Bien sûr, on aurait pu choisir d'abord la boucle sur k et obtenir :

```
forseq k
  forpar j
    forseq i
      ...
```

1. Faire tourner l'algorithme sur l'exemple avec D_2.
2. Faire tourner l'algorithme sur l'exemple avec D_3.
3. Prouver la correction de l'algorithme. Pour cela, prouver qu'on a le droit de déplacer à l'extérieur et paralléliser une boucle correspondant à une ligne nulle, et préciser quelles sont les dépendances qui sont satisfaites après le choix d'une boucle, son déplacement à l'extérieur et sa séquentialisation.
4. Pensez-vous que cet algorithme conduise à un nombre maximal de boucles parallèles ? (utiliser l'exemple avec D_3)
5. *(difficile)* Montrer que trouver le nombre maximal de boucles parallèles est NP-complet, par réduction à partir de $SET - COVER$: étant donnés un ensemble X à n éléments, K sous-ensembles de X et une borne B, peut-on trouver B sous-ensembles parmi les K tels que tout élément de X appartienne à au moins l'un de ces B sous-ensembles ?