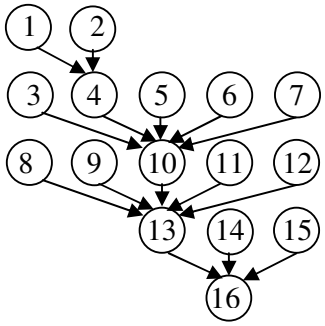
Devoir Surveillé N°1

Matière	**:** Algorithmique & Architectures Parallèles	**Date**	**:** 18/11/2008
Enseignant	**:** Wahid NASRI	**Durée**	**:** 1h30
Sections	**:** MI4 GL & RI	**Nb. Pages**	**:** 2
Documents	**:** Non autorisés	**Barème**	**:** 3+2+3+4+8

N.B. * Toutes les réponses doivent être clairement expliquées et justifiées.
**** Dans tous les exercices, on néglige les coûts de communications entre processeurs.***

Exercice 1

Soit le graphe ci-dessous représentant les dépendances entre tâches d'un calcul particulier, où toutes les tâches sont supposées UET :



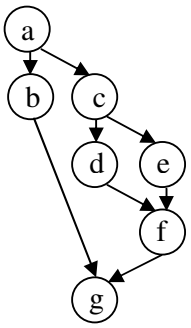
1- Déterminer le degré de parallélisme maximum correspondant au calcul en question.

2- Déterminer P_{opt}.

3- Proposer un ordonnancement du graphe sur un tel nombre de processeurs.

Exercice 2

Soit le graphe de tâches ci-dessous. Les coûts des différentes tâches, exprimés en unités de temps, sont comme suit : a=g=1, d=e=f=2 et b=c=3.



Proposer un ordonnancement du graphe sur une machine parallèle composée de 3 processeurs tout en essayant d'optimiser le temps d'exécution parallèle et les ressources matérielles utilisées.

Exercice 3

Soit l'algorithme suivant :

```
Pour i = 1 , n
    Tâche Ti,i : X(i) = B(i) / A(i,i)
    Pour j = i+1 , n
        Tâche Ti,j : B(j) = B(j) – A(j,i) * X(i)
    FinPour
FinPour
```

1. Représenter le graphe de tâches correspondant à la décomposition en tâches proposée.
2. Déterminer le temps d'exécution de l'algorithme parallèle en supposant un modèle PRAM.

Exercice 4

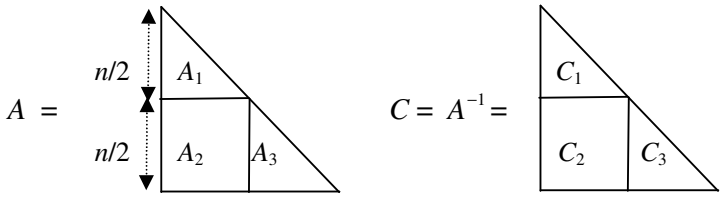
Soit l'algorithme suivant, où A, B et C sont des matrices carrées de taille n :

```
Pour i = 1 , n
    Pour j = 1 , n
        Pour k = 1 , n
            Cij = Cij + Aik * Bkj
        FinPour
    FinPour
FinPour
```

1. Proposer une décomposition en tâches de granularité grossière.
2. Représenter le graphe de tâches correspondant à la décomposition proposée.
3. Proposer un ordonnancement du graphe sur un nombre p de processeurs tel que $p \geq n/2$ (n est supposé pair). En déduire l'accélération. Discuter.

Exercice 5

On se propose de paralléliser un algorithme séquentiel pour inverser une matrice triangulaire inférieure (TI) A de taille $n=2^k$ (k est un entier strictement positif et n est supposé très grand). L'inverse de A, notée C, peut être déterminée comme suit :



avec : $C_1 = A_1^{-1}, \quad C_3 = A_3^{-1}, \quad C_2 = -A_3^{-1} A_2 A_1^{-1}$

C'est Heller qui a proposé cet algorithme en 1973. Ainsi, inverser A, par la décomposition précédente, revient à inverser deux sous-matrices TI de taille $n/2$ et à multiplier trois matrices ayant la même taille, la première et la troisième étant TI et la seconde pleine. A_1 et A_3 étant elles-mêmes TI, un algorithme d'inversion de A de type « Diviser pour régner » consiste à appliquer la même décomposition sur A_1 et A_3 et ainsi de suite jusqu'à ce qu'on atteigne un seuil (taille des sous-matrices) donné.

1- Proposer une décomposition en tâches de l'algorithme séquentiel.

2- En déduire le graphe de tâches correspondant à un algorithme de profondeur 1 et à un autre de profondeur 3.

3- Donner l'ordonnancement associé au graphe de tâches ainsi obtenu (en appliquant une seule fois la décomposition) en disposant de $p=2$ processeurs.

4- En déduire le temps d'exécution de l'algorithme parallèle.

5- Déterminer l'efficacité et l'accélération de l'algorithme parallèle.

6- Qu'en remarquez-vous ? Proposer alors une amélioration et refaire toute l'étude.

7- Décrire un algorithme performant pouvant s'exécuter sur $p=2^r$ processeurs (r est un entier >1).