Exercices INF7440 : série #7

1. Produit de deux matrices $n \times n$

```
procedure produitScalaire( int a[*], int b[*], int n ) returns int ps
{
  ps = 0;
  for [k = 1 to n] {
    ps += a[k] * b[k]
  }
}

# Multiplications de matrices.
procedure multMatrices( int a[*,*], int b[*,*], ref int c[*,*], int n )
{
  co [i = 1 to n, j = 1 to n]
    c[i,j] = produitScalaire( a[i,*], b[*,j], n );
  oc
}
```

Algorithme 1: Multiplication de matrices $n \times n$: première version

L'Algorithme 1 est un algorithme parallèle pour effectuer le produit de deux matrices $n \times n$. Déterminez les caractéristiques suivantes de l'algorithme :

a. Temps d'exécution (complexité asymptotique).

b. Nombre (exact) de processeurs requis.

c. Coût de l'algorithme (complexité asymptotique).

d. Travail effectué par l'algorithme (complexité asymptotique).

2. Produit de deux matrices $n \times n$ (deuxième version)

```
procedure produit( int x, int y ) returns int z
{ z = x * y; }

procedure somme( int x, int y ) returns int z
{ z = x + y; }

procedure sommeVect( int t[*], int n ) returns int s
{
  for [i = 1 to lg(n)] {
    co [j = 1 to n/(2**i)]
      t[j] = somme( t[2*j-1], t[2*j] );
    oc
  }
  s = t[1];
}

# Multiplications de matrices.
procedure multMatrices( int a[*,*], int b[*,*], ref int c[*,*], int n )
{
  int t[n,n,n];

  co [i = 1 to n, k = 1 to n, j = 1 to n]
    t[i,k,j] = produit( a[i,k], b[k,j] );
  oc
  co [i = 1 to n, j = 1 to n]
    c[i,j] = sommeVect( t[i,*,j], n );
  oc
}
```

Algorithme 2: Multiplication de matrices $n \times n$: deuxième version

Une deuxième façon de calculer le produit de deux matrices $n \times n$ est présentée à l'Algorithme 2. Comme pour l'exercice précédent, déterminez les caractéristiques suivantes de l'algorithme :

a. Temps d'exécution (complexité asymptotique).

b. Nombre (exact) de processeurs requis.

c. Coût de l'algorithme (complexité asymptotique).

d. Travail effectué par l'algorithme (complexité asymptotique).

3. Produit de deux matrices $n \times n$ (version améliorée)

Est-il est possible de modifier l'algorithme précédent pour obtenir un coût qui soit du même ordre que le coût de l'algorithme séquentiel *standard* tout en conservant le même temps (asymptotique) d'exécution? Si oui, expliquez brièvement comment et donnez l'algorithme modifié. Si non, expliquez pourquoi.