Exercices : solutions de la série #8

Solution à l'exercice 1.

```
procedure valeurGauche( int gauche, int droite ) returns int r
{ r = gauche; }

procedure diffuser( int v, res int r[*], int n )
{
  r[1] = v;
  calculerPrefixes( r, r, n, valeurGauche );
}
```

Solution à l'exercice 2.

```
procedure estPair( int x ) returns int r
{ if ( x % 2 == 0 ) { r = 1; } else { r = 0; } }

procedure estImpair( int x ) returns int r
{ if ( x % 2 != 0 ) { r = 1; } else { r = 0; } }

procedure transferer( int x, ref int xOut[*],
                      int nbPairsAvant, int nbImpairsAvant,
                      int dernierPair )
{
  if (x % 2 == 0) { xOut[nbPairsAvant] = x; }
              else { xOut[dernierPair + nbImpairsAvant] = x ; }
}

procedure paqueter( int xIn[*], res int xOut[*], int n )
{
  int nbPairsAvant[n], nbImpairsAvant[n];
  int maxs[n];
  int dernierPair;

  co [i = 1 to n]
    nbPairsAvant[i] = estPair(xIn[i]);
  oc
  co [i = 1 to n]
    nbImpairsAvant[i] = estImpair(xIn[i]);
  oc
  calculerPrefixes( nbPairsAvant,   nbPairsAvant,   n, plus2 );
  calculerPrefixes( nbImpairsAvant, nbImpairsAvant, n, plus2 );
  calculerPostfixes( nbPairsAvant, maxs, n, max2 );
  dernierPair = maxs[n];

  co [i = 1 to n]
    transferer( xIn[i], xOut, nbPairsAvant[i], nbImpairsAvant[i], dernierPair );
  oc
}
```

L'algorithme demande n processeurs, pour un temps $\Theta(\lg\ n)$, donc avec un coût $\Theta(n\ \lg\ n)$.

Solution à l'exercice 3.

```
#
# Procedures auxiliaires.
#
procedure inf( int i, int n ) returns int r
{ r = (i-1) * lg(n) + 1; }

procedure sup( int i, int n ) returns int r
{ r = i * lg(n); }

procedure sum2( int x, int y ) returns int r
{ r = x + y; }

# Procedures auxiliaires.
procedure calculerEcartsSeq( int a[*], int moyenne, ref int ecarts[*], int lo, int hi )
{
  for [k = lo to hi] {
    ecarts[k] = a[k] - moyenne;
  }
}

#
# Procedure principale pour l'algorithme.
#

procedure calculerEcarts( int a[*], ref int e[*], int n )
{
  int prefixesSomme[n];
  int moyenne;

  # On calcule la somme des elements.
  # On suppose que ceci se fait par un calcul optimal des prefixes.
  calculerPrefixes( a, prefixesSomme, n, sum2 );

  # On calcule la moyenne.
  moyenne = prefixesSomme[n] / n;

  # On calcule les ecarts a la moyenne.
  co [i = 1 to n/lg(n)]
    calculerEcartsSeq( a, moyenne, e, inf(i, n), sup(i, n) );
  oc
}
```

Analyse :

- Nombre de processeurs : $n/\lg\ n$.
- Temps : $O(\lg\ n)$;
- Coût : $O(n)$.