Corrigé de la Composition Hors Classement

Jean-Jacques Lévy Yves Robert

20 décembre 2000 – 2h

A. Décompositions d'entiers

Question 1

```
class ListeDecomp {
  int val;
  ListeDecomp suivant;
  ListeDecomp (int v, ListeDecomp x) { val = v; suivant = x; }

  static boolean estDecomp (ListeDecomp x, int n) {
    int somme = 0;
    for (ListeDecomp y = x; y != null; y = y.suivant)
      if (y.val <= 0) return false;
      else somme = somme + y.val;
    return n == somme;
  }
}
```

Question 2 (1,1,1,1,1) < (1,1,1,2) < (1,1,2,1) < (1,1,3) < (1,2,1,1) < (1,2,2) < (1,3,1) < (1,4) < (2,1,1,1) < (2,1,2) < (2,2,1) < (2,3) < (3,1,1) < (3,2) < (4,1) < (5). On modifie 2 chiffres (et on ajoute éventuellement des 1) dans toute décomposition pour passer à la suivante.

Question 3 Bien penser au cas $n = 0$ dans ce test.

```
  static boolean estMax (ListeDecomp x, int n) {
    if (x == null) return n == 0;
    else return x.val == n && x.suivant == null;
  }
```

Question 4 Si les deux derniers chiffres sont i et j, on passe au suivant en les changeant par $i + 1$ suivi par $j - 1$ fois 1.

```
  static ListeDecomp suivante (ListeDecomp x) {
    if (x.suivant.suivant == null) {
      ListeDecomp r = null;
      for (int i = 0; i < x.suivant.val - 1; ++i)
        r = new ListeDecomp (1, r);
      return new ListeDecomp (x.val + 1, r);
    } else return new ListeDecomp (x.val, suivante (x.suivant));
  }
```

B. Partage équilibré de périmètre

Question 5 Le coût est clairement linéaire pour une coupe donnée. Comme il y a un nombre quadratique de coupes, le coût total est en $O(n^3)$.

Question 6 a)

```
static int perimetre (int[] a) {
  int p = 0;
  for (int k = 0; k < a.length; ++k)
    p = p + a[k];
  return p;
}
```

b) Soit p le périmètre. Pour un indice i donné, posons $s_k = a[i] + a[i+1] + \cdots a[k]$ et $s'_k = p - s_k$. On cherche à minimiser $|s'_k - s_k| = |p - 2s_k|$. Clairement le minimum est autour de $p - 2s_k = 0$. Et donc pour $s_k \in \{\lfloor p/2 \rfloor, \lfloor p/2 \rfloor + 1\}$.

c) On en déduit la fonction cherchée. Pour chaque valeur de i, la recherche de j est linéaire, d'où un coût total en $O(n^2)$.

```
static void partage1(int[] a) {
  int n = a.length, j, iMin = 0, jMin = 0;
  int p = perimetre (a), dMin = p; int d;
  for (int i = 0; i < n; ++i) {
    int k = i; int s = a[i];
    while (2*s <= p) { k = (k + 1 ) % n; s = s + a[k]; }
    int delta = Math.abs(p - 2*(s-a[k]));
    int delta1 = Math.abs(p - 2*s);
    if (delta < delta1) { j = k; d = delta; }
    else { j = (k+1) % n ; d = delta; }
    if (d < dMin) {
      iMin = i; jMin = j; dMin = d;
    }
  }
  System.out.println("i= " + iMin + ", j = " + jMin + ", d = " + dMin);
}
```

Question 7 On réutilise la valeur courante de s en passant d'une valeur de i à la suivante: k ne peut qu'augmenter et ne peut d'ailleurs faire plus de deux tours au fil de l'exécution.

```
static void partage2(int[] a) {
  int n = a.length, j, iMin = 0, jMin = 0;
  int p = perimetre (a), dMin = p, d; int s = 0; int k = 0;
  for (int i = 0; i < n; ++i) {
    s = (i == 0) ? a[0] : s - a[i-1] ;
    while (2*s <= p) { k = (k + 1 ) % n; s = s + a[k]; }
    int delta = Math.abs(p - 2*(s-a[k]));
    int delta1 = Math.abs(p - 2*s);
    if (delta < delta1) { j = k; d = delta; }
    else { j = (k+1) % n ; d = delta; }
    if (d < dMin) {
      iMin = i; jMin = j; dMin = d;
    }
  }
  System.out.println("i= " + iMin + ", j = " + jMin + ", d = " + dMin);
}
```