```
Affichage d'une répartition optimale de H heures de révision sur les n unités
------------------------------------------------------------------------------

1) Rappel des données du problème :
-- n unités,
-- H heures de révision,
-- e(i,h) : estimation de la note à l'unité i pour h heures de révision de cette unité.
Ces estimations sont dans un tableau E[0:n][0:H+1] de terme général E[i][h] = e(i,h),
-- les coefficients des unités dans un tableau C[0:n] de terme général C[i] = c_{i}.

2) L'appel de fonction int[][][] MA = calculerMA(E,C) a retourné deux tableaux M et A
que l'on a récupéré par l'instruction int[][] M = MA[0], A = MA[1] :
• M[0:n+1][O:H+1] est de terme général M[k][t] = m(k,t) = somme des notes des k premières
unités pour t heures de révision réparties au mieux sur les k premières unités.
Il s'agit d'une somme de notes pondérée par les coefficients des notes.
• A[0:n+1][0:H+1] = a(k,t) = arg m(k,t) = le nombre d'heures allouées à la k-ème unité
(unité de numéro k-1) dans la répartition optimale de t heures de révision sur les k
premières unités.

Problème : afficher une répartition optimale RO(n,H) de H heures sur les n unités.
La répartition RO(n,H) est un ensemble de couples "unité <-- nombre d'heures allouées."
Sur l'exemple de l'ébauche de programme nous avons RO(n,H) = { 0 <- 3, 1 <- 2, 2 <- 5 } :
   3 heures allouées à l'unité 0 (l'unité 0 a reçu 3 heures)
   2 heures allouées à l'unité 1 (l'unité 1 a reçu 2 heures)
   5 heures allouées à l'unité 2 (l'unité 2 a reçu 4 heures)

L'affichage de la répartition est plus complet. Pour chaque unité il donne :
la note estimée, le coefficient, la contribution à la somme totale (coefficient x note)
   C[0] * e(0,3) = 4 * 12 = 48
   C[1] * e(1,2) = 3 * 18 = 54
   C[2] * e(2,5) = 4 * 18 = 72

Pour résoudre le problème "afficher une répartition optimale RO(n,H)"
nous résolvons un problème plus général, "afficher une répartition optimale RO(k,t)."
Le problème RO(n,H) à résoudre est le cas particulier RO(k=n,t=H).

Equation de récurrence de la répartition optimale RO(k,t).
---------------------------------------------------------
Il s'agit de la répartition optimale de t heures de révision sur les k premières unités.

1) Base de la récurrence, k = 0. Le sous-ensemble [0:k=0] des k premières unités est vide.
              RO(0,t) = Ø.
2) Hérédité 1 ≤ k < n+1.
Le nombre d'heures de révision allouées à la k-ème unité dans une répartition optimale
de t heures sur les k premières unités est a(k,t).
Il reste donc t - a(k,t) heures. Elles seront réparties de façon optimale sur les k-1
premières unités. Autrement dit :
              RO(k,t) = RO(k-1,t-a(k,t)) union { k-1 <-- a(k,t) }

Le principe d'optimalité de Bellman impose RO(k-1,t-a(k,t)) car si la répartition des
t-a(k,t) heures restantes n'était pas optimale, la répartition des t heures sur les
k premières unités ne serait pas optimale.

La fonction d'affichage se déduit directement de cette équation de récurrence.

static void aro(int[][] A, int[][] E, int[] C, int k, int t){
/* Affichage de la répartition optimale RO(k,t). Appel principal : aro(A, E, C, n, H) */
    if (k==0) return; // RO(0,t) = Ø. Sans rien faire, RO(0,t) a été affichée.
    int akt = A[k][t]; // RO(k,t) = RO(k-1,t-akt) union { k-1<--akt }
    aro(A,E,C,k-1,t-akt); // RO(k-1,t-akt) a été affichée
    System.out.printf("C[%d] * e(%d,%d) = %d * %d = %d\n",
        k-1, k-1, akt, C[k-1], E[k-1][akt], C[k-1]*E[k-1][akt]);
    // les informations sur l'allocation "k-1 <-- akt" ont été affichées
    // RO(k-1,t-akt) a été affichée, "k-1<--akt" a été affichée, donc RO(k,t) a été affichée
}

That's all, folks.
```