Partiel MIM1 – Algorithmique

Yves Robert

Mardi 24 Octobre 2000, 10h15–12h15

1 Diviser pour régner (facile)

On donne la procédure de tri suivante :

```
procedure tri_bizarre (var A : array[1..n] of integer; i,j : integer);
var k : integer;
begin
  if A[i] > A[j] then échange A[i] ↔ A[j];
  if i + 1 ≥ j then return;
  k := (j − i + 1) div 3;
   tri_bizarre(A, i, j − k);   { les deux premiers tiers }
   tri_bizarre(A, i + k, j);   { les deux derniers tiers }
   tri_bizarre(A, i, j − k);   { les deux premiers tiers à nouveau }
end;
```

1. Montrer que l'appel tri_bizarre(A,1,n) trie bien le tableau A
2. Etablir la complexité au pire de tri_bizarre.

2 Glouton (facile)

On considère n événements sportifs devant se dérouler le même jour dans des gymnases de la bonne ville de Lyon. L'événement i couvre l'intervalle de temps $[d_i, f_i[$: il débute à l'heure d_i et se termine à l'heure f_i.

Proposer un algorithme (glouton) pour déterminer le nombre minimal de gymnases qu'il faut réserver pour organiser tous les événements :

1. Justifier soigneusement la correction de votre algorithme
2. Donner sa complexité

3 Programmation dynamique (moyen)

On dote n réels positifs $s_1, s_2, \ldots, s_n$ de somme 1. On veut partitionner le carré unité en n rectangles de surfaces $s_1, s_2, \ldots, s_n$. On cherche un partitionnement en colonnes comme sur la Figure 1. La fonction objective à minimiser est la somme des traits dessinés, i.e. (à une constante près) la somme des demi-périmètres des n rectangles.

1. Résoudre le problème dans le cas $n = 4$ et $s_1 = s_2 = s_3 = s_4 = 0.25$.
2. Résoudre le problème dans le cas $n = 6$, $s_1 = 0.5$ et $s_2 = s_3 = s_4 = s_5 = s_6 = 0.1$

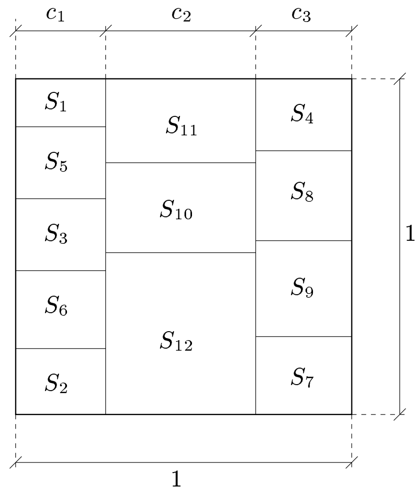

FIG. 1 – Partitionnement du carré unité en colonnes de rectangles.

3. Dans le cas général, on ne connaît pas le nombre de colonnes qu'on va utiliser. Mais :
   (a) Montrer qu'on peut se ramener à placer les s_i dans l'ordre croissant
   (b) Résoudre le problème par programmation dynamique

4 Algorithmique (difficile)

On donne un tableau $A[1..n]$ de n entiers dans $\mathbb{Z}$. Une coupe de A est une suite $A[i..j[$, $i \leq j$, d'éléments consécutifs $A[i], A[i+1], \ldots, A[j-1]$ de A (si $i = j$ la coupe est vide).

1. Montrer que le code suivant calcule la valeur maximale de la somme des éléments d'une coupe de A :

```
maxSum ← 0; thisSum ← 0;
for i = 1 to n do
  thisSum ← thisSum + A[i];
  if (thisSum > maxSum) then
    maxSum ← thisSum
  else
    if (thisSum < 0) then thisSum ← 0;
```

   Quel est le coût ?

2. Comment calculer la valeur maximale de la somme des éléments d'une coupe de A, en se restreignant aux sommes de valeur strictement négative ? Quel est le coût de votre solution ?
3. Comment calculer la valeur maximale du produit des éléments d'une coupe de A ? Quel est le coût de votre solution ?