Université Lille1 Sciences et Technologies

Info 204 - ASD

2012–2013
Licence STS - Semestre 4

TD3 : programmation dynamique

Exercice 1 : Nombres de Catalan

Les nombres de Catalan sont définis de manière récursive ainsi :

$$\begin{cases} \text{catalan}(0) & = & 1 \\ \text{catalan}(1) & = & 1 \\ \text{catalan}(n) & = & \sum_{k=0}^{n-1} \text{catalan}(n-k-1) \times \text{catalan}(k) \end{cases}$$

Question 1.1 Dessinnez l'arbre des appels récursifs lors du calcul de catalan(4).

Question 1.2 Indiquez quelles sont les valeurs de catalan$(k), k < n$ dont il faut disposer pour calculer catalan(n).

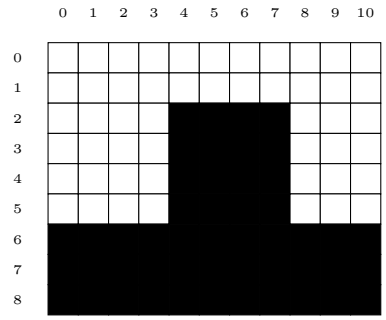
Question 1.3 Déduisez-en la taille du tableau nécessaire pour stocker les calculs intermédiaire lors du calcul de catalan(n). Comment devra être initialisé ce tableau ?

Question 1.4 Ecrivez en OCaml un programme qui calcule catalan(n) par programmation dynamique.

Question 1.5 Donnez la complexité en temps et en espace de votre fonction ainsi qu'une borne asymtotiquement approchée.

Exercice 2 : Extrait du DS de 2012

On s'intéresse dans cet exercice à un problème d'analyse d'image. On considère des images en noir et blanc et le problème qu'on veut traiter est la détection du plus grand carré de cases noires dans l'image. Par exemple, pour l'image suivante :



l'algorithme indiquera que le plus grand carré noir se situe en case (4,2)[1] et est de longueur 4.

On propose la récursion suivante pour résoudre le problème. On note $\ell(i,j)$ la longueur du plus grand carré noir se terminant en position $(i-1, j-1)$ dans l'image. On a alors :

$$\ell(i,j) = \begin{cases} 0 & \text{si la case de l'image en position } (i-1,j-1) \text{ est blanche} \\ 1 + \min(\ell(i-1,j-1), \ell(i-1,j), \ell(i,j-1)) & \text{sinon} \end{cases}$$

et

$$\ell(i,0) = \ell(0,j) = 0 \text{ pour } 0 \leq i \leq n, 0 \leq j \leq m$$

L'obtention du résultat, c'est-à-dire la plus grande longueur L, consiste à sélectionner le couple (i,j) maximisant $l(i,j)$:

$$L = \max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} \ell(i,j)$$

si on considère une image de taille $n \times m$.

Nous allons donc mettre en œuvre une résolution par programmation dynamique. Pour simplifier, on supposera qu'une image est donnée par un tableau à deux dimensions dont les cases contiennent soit 0 soit 1, 0 indiquant les cases blanches, et 1 les cases noires. On manipulera trois variables déclarées comme ci-dessous :

[1] D'autres coordonnées sont possibles – (4,3), 4,4), (4,5) – , c'est simplement celles que mon algorithme calcule

```
let img =
  [|
    [| 0;0;0;0;0;0;1;1;1 |];
    [| 0;0;0;0;0;0;1;1;1 |];
    [| 0;0;0;0;0;0;1;1;1 |];
    [| 0;0;0;0;0;0;1;1;1 |];
    [| 0;0;1;1;1;1;1;1;1 |];
    [| 0;0;1;1;1;1;1;1;1 |];
    [| 0;0;1;1;1;1;1;1;1 |];
    [| 0;0;1;1;1;1;1;1;1 |];
    [| 0;0;0;0;0;0;1;1;1 |];
    [| 0;0;0;0;0;0;1;1;1 |];
    [| 0;0;0;0;0;0;1;1;1 |]
  |]
;;
let n = Array.length img;;
let m = (Array.length img.(0));;
```

Question 2.1 Quelle est la taille de la table de programmation dynamique ?

Question 2.2 Donnez le code CAML qui réalise l'initialisation de la table.

Question 2.3 Donnez le code CAML qui réalise le remplissage de la table.

Question 2.4 Donnez le code CAML qui extrait le résultat.

Question 2.5 Quel est le comportement asymptotique en temps ? Justifez clairement.

<u>Exercice 3</u> : Le problème du sac à dos - ou le dilemme du voleur

Le fameux problème du sac à dos est le suivant. Etant donné un ensemble d'objets de poids et de valeur différentes, comment remplir un sac à dos de capacité donnée tel que la somme des valeurs des objets soit maximale.

Formalisons un peu une variante qu'on pourra résoudre par programmation dynamique.

On suppose qu'on dispose de n objets et on note v_i et w_i respectivement la valeur et le poids de l'objet i.

On note c la capacité du sac à dos.

Il s'agit donc de trouver les $x_i \in 0, 1$ de sorte que

$$\sum_{i=1}^{n} x_i w_i \leq c \text{ et } \sum_{i=1}^{n} x_i v_i \text{ soit maximal}$$

Notons $T(i, j)$ la valeur maximale pouvant être atteinte avec les i premiers objets pour un poids maximal j. $T(i, j)$ donne ainsi la résolution du problème où on ne considèrerait que les i premiers objets et un sac à dos de capacité j.

Question 3.1 Trouvez la meilleure solution si on considère $c = 4$, $w_1 = 1$, $w_2 = 3$, $w_3 = 2$ et $v_1 = 1$, $v_2 = 4$, $v_3 = 3$.

Question 3.2 Que vaut $T(1, j)$?

Question 3.3 Si $w_i > j$, peut-on inclure l'objet i ? Que vaut alors $T(i, j)$?

Question 3.4 Si $w_i \leq j$, que vaut $T(i, j)$?

On aboutit ainsi à une équation de récurrence pour résoudre le problème qui se prête bien à une résolution par programmation dynamique.

Question 3.5 Dessinnez et remplissez la table pour l'exemple donné ci-dessus.

Question 3.6 Ecrivez le code OCaml permettant la résolution du problème.

On écrira une fonction `sac_a_dos : int array -> int array -> int -> int` qui calcule la valeur maximale en fonction de deux tableaux de valeurs et de poids et d'une capacité.

Par exemple, `sac_a_dos [|1;4;3|] [|1;3;2|] 4` fournira la valeur trouvée à la question 1.