Master 1 STL 4I505 CPA – Examen rattrapage Juin 2017

Les documents distribués dans le cadre du cours et les notes personnelles sont autorisés.

A Programmation dynamique et structures de données

A.1 Piano (3 points)

Jean-Claude vient d'apprendre à jouer au piano. Pour l'instant il ne sait jouer qu'une seule note, la plus à gauche du piano. Il active un métronome et à chaque tic il joue ou ne joue pas la note. Pour que la musique produite soit plus intéressante il ne joue jamais la note pendant deux tics consécutifs. Maintenant son côté informaticien reprend le dessus, et il se demande, étant donné n tics, de combien de différentes manières peut-on jouer avec la contrainte mentionnée. Appelons $f(n)$ cette valeur. Par exemple si on note o l'absence d'une note et p une note jouée on a les valeurs suivantes :

n	$f(n)$	explication
0	1	par convention
1	2	o, p
2	3	oo, op, po mais pas pp
3	5	ooo, opo, poo, oop, pop

Donnez un programme dynamique de complexité $O(n)$ pour calculer $f(n)$.

A.2 Sucré-Acide (5 points)

Un fabriquant de bonbons dispose d'un bâton comestible composé de segments d'un centimètre chacun. Un segment peut être sucré ou acide. Les segments sont numérotés de 0 à $n-1$. Le bâton est décrit par un tableau $x \in \{0,1\}^n$, où $x[j] = 1$ si et seulement si le j-ème segment est sucré. Le fabriquant peut découper le bâton en plusieurs morceaux, chacun étant constitué d'un nombre entier de segments. Les enfants achètent un morceau seulement s'il contient strictement plus de segments sucrés que de segments acides. On veut calculer la longueur totale des morceaux vendables que le fabriquant peut produire à partir d'un bâton décrit par x ? Donnez un programme dynamique qui résout ce problème en temps $O(n^2)$. Des points partiels seront données pour un algorithme correct de plus grande complexité.

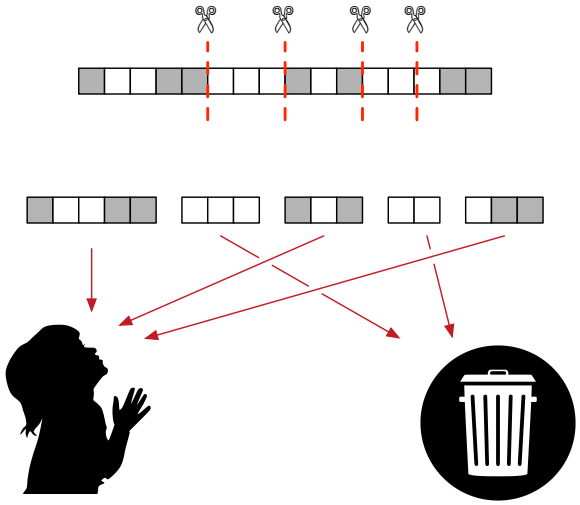

Figure 1: Le problème Sucré-Acide illustré sur l'exemple $x = 1001100010100011$. Les segments sucrés sont montrés en gris. La réponse optimale est 11, ce qui correspond au total des morceaux de longueur $5, 3$ et 3.

B Éléments de correction

B.1 Piano

Soit F_n l'ensemble des chaînes $x \in \{0,1\}^2$ ne contenant pas deux 1 consécutifs. F_0 est composé seul du mot vide ε et $F_1 = \{o, p\}$. Pour $n \geq 2$, soit $x \in F_n$ une chaîne arbitraire. La chaîne x termine par o ou par p. Dans le premier cas x est de la forme $x = yo$ pour une chaîne arbitraire $y \in F_{n-1}$. Dans le deuxième cas, l'avant dernier symbole est forcément o, et dans ce cas $x = yop$ pour une chaîne arbitraire $y \in F_{n-2}$. Nous avons donc la récursion suivante.

Cas de base $f(0) = 1, f(1) = 2$ et pour tout $n \geq 2$ $f(n) = f(n-1) + f(n-2)$. Pour information, il s'agit du nombre de Fibonacci. Pour calculer $f(n)$, il suffit de remplir un tableau avec les valeurs $f(0), f(1), \ldots, f(n)$ dans l'ordre des indices croissant, chaque valeur se calculant en temps constant. Ceci montre une complexité $O(n)$ pour calculer $f(n)$.

B.2 Sucré-Acide

Pour les indices $0 \leq i \leq j \leq n-1$ appelons $A[i,j]$ le *score* du morceau composé des segments i à j. Donc $A[i,j] = j + 1 - i$ s'il y a plus de segments sucrés dans le morceaux qu'acide et $A[i,j] = 0$ sinon. Pour calculer A nous utilisons un tableau $C[i,j]$ qui contient le nombre de segments sucrés entre les segments i et j. Pour $0 \leq i \leq n-1$ nous avons $C[i,i] = x_i$ et pour $i < j \leq n-1$ nous avons $C[i,j] = C[i,j-1] + x_j$. Puis

$$A[i,j] = \begin{cases} j - i + 1 & \text{si } C[i,j] > (j+1-i)/2, \\ 0 & \text{sinon.} \end{cases}$$

Les matrices A et C se calculent en temps $O(n^2)$, car il y a $O(n^2)$ valeurs et chacune se calcule en temps constant.

Appelons $B[j]$ le score maximum qu'on peut obtenir à partir de segments 0 à j avec un découpage optimal. Pour un découpage optimal fixé soit $0 \leq k \leq j$ le début du dernier morceau. Par additivité des scores, si $k > 0$, alors le découpage du reste composé des segments de 0 à $k-1$ doit à son tour être optimal. Donc $B[0] = A[0,0]$ et pour $j > 0$ on a

$$B[j] = \max \left\{ A[0,j], \max_{1 \leq k \leq j} (B[k-1] + A[k,j]) \right\}.$$

Pour le calcul de B il faut évaluer n valeurs, chacun étant la minimisation sur $O(n)$ expressions, ce qui donne une complexité en $O(n^2)$.