Master 1 STL 4I505 CPA – Examen Rattrapage juin 2018

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

A Programmation dynamique

Les questions suivantes sont toutes à résoudre par la programmation dynamique. Voici une correction type pour que vous voyez quel genre de réponse est attendue.

Exemple (0 points)

Un mot w sur l'alphabet à deux symboles $\{(,)\}$ est bien parenthésé si on peut associer chaque parenthèse ouvrante x à une parenthèse fermante à droite de x et que les associations soient *sans collision* et *ne se croisent pas*. Voir Figure 1.

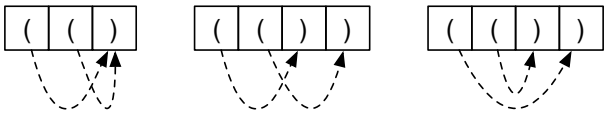

Figure 1: à gauche une association avec collision, au centre une avec croisement, et à droite une sans collision ni croisement.

Formellement on associe à un mot $w \in \{(,)\}^*$ un compteur $p[w]$ défini comme le nombre de parenthèses ouvrantes dans w moins le nombre de parenthèses fermantes dans w. On dit que $w \in \{(,)\}^n$ est bien parenthésé si $p[w] = 0$ et que pour tout $i = 1, \ldots, n-1$, on ait $p[w_{1,\ldots,i}] \geq 0$, où $w_{1,\ldots,i}$ est le mot constitué des i premières lettres de w.

bien parenthésé	mal parenthésé
ε	(
()	)
()()	)(
(())	(()
(())()	())(()

Il est facile de déterminer si un mot est bien parenthésé, en effectuant un unique parcours pendant lequel on maintien un compteur représentant $p[w_{1,\ldots,i}]$. Mais si un mot n'est pas bien parenthésé, on peut toujours le rendre parenthésé en supprimant des parenthèses à différents endroits de la chaîne. Étant donnée un mot $w \in \{(,)\}^n$, déterminer le nombre minimal de parenthèses à supprimer pour rendre w bien parenthésé. Une complexité en $O(n^3)$ est attendue.

Réponse type : Soit $A_{i,j}$ le nombre minimal de parenthèses à supprimer pour rendre le mot $w_{i,\ldots,j}$ bien parenthésé, avec $1 \leq i \leq j \leq n$. Nous étendons la notation à $i = j + 1$ et définissons $A_{j+1,j} = 0$, correspondant à la solution optimale pour le mot vide.

La récursion est sur la valeur $j - i$. Le cas de base est $A_{j+1,j} = 0$ pour le mot vide. Pour la récursion avec $j \geq i$, considérons la première lettre w_i du mot $w_{i,\ldots,j}$ et considérons une solution optimale. Soit w_i est supprimée dans la solution optimale, et dans ce cas $A_{i,j} = 1 + A_{i+1,j}$ par composition des solutions. Soit cette lettre est associée à une lettre w_ℓ avec $i + 1 \leq \ell \leq j$. Cette situation n'est possible seulement si $w_i = ($ et $w_\ell =)$. Les chaînes $w_{i+1,\ldots,\ell-1}$ et $w_{\ell+1,\ldots,j}$ forment des sous-problèmes indépendants car les associations de parenthèses ne peuvent pas se croiser. Dans ce cas nous avons alors la récursion $A_{i,j} = A_{i+1,\ell-1} + A_{\ell+1,j}$.

En résumé le programme dynamique consiste en $O(n^2)$ variables $A_{i,j}$, dont le cas de base est $A_{i,j} = 0$ si $i = j + 1$ et sinon

$$A_{ij} = \min \left\{ 1 + A_{i+1,j}, \min_{\ell} A_{i+1,\ell-1} + A_{\ell+1,j} \right\},$$

où le minimum intérieur est pris sur les valeurs $i + 1 \leq \ell \leq j$ avec $w_i = ($ et $w_\ell =)$. Par convention le minimum sur l'ensemble vide est défini comme valant $+\infty$. La réponse au problème est $A_{1,n}$. Ces variables doivent être calculées dans l'ordre suivant. D'abord pour tout $j = \{1, \ldots, n\}$ $A_{j+1,j} = 0$ est posé. Puis pour tout $k = 0, \ldots, n-1$ (et dans cet ordre) et pour tout $j \in \{1 + k, \ldots, n\}$, $A_{j-k,j}$ est calculée en utilisant la récursion ci-haute.

Comme il y a $O(n^2)$ variables et que chacune est calculée par minimisation sur $O(n)$ alternatives, le programme dynamique a une complexité en temps de $O(n^3)$. Il existe de meilleurs algorithmes pour ce problème, mais ce n'est pas le propos pour l'instant.

A.1 Coefficients binomiaux (1 points)

Le développement de l'expression $(a+b)^n$ est une somme d'expressions de la forme $\binom{n}{k} a^k b^{n-k}$ pour tout $k = 0, \ldots, n$. Par exemple $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$. Donnez un programme dynamique pour calculer $\binom{n}{k}$ en temps $O(n^2)$.

A.2 Subsetsum (2 points)

Dimitri fabrique un modèle d'avion. Il estime que pour que le centre de gravité soit à l'endroit nécessaire, il doit ajouter du leste dans le nez de l'avion. Plus précisément il doit ajouter exactement L grammes dans le nez. Pour cela il dispose de n billes de différents poids, la i-ème bille pesant exactement p_i grammes. Étant données L et $p_1, \ldots, p_n$, Dimitri doit décider s'il existe un ensemble $S \subseteq \{1, \ldots, n\}$ avec $\sum_{i \in S} p_i = L$. Aidez Dimitri avec un programme dynamique de complexité $O(Ln)$.

A.3 Emballage de jouets (4 points)

Odile doit s'occuper d'une machine étrange à fabriquer des jouets. Elle fabrique deux sortes de jouets, des voitures pesant 100 grammes et des camions pesant 150 grammes. Ce qui est

étrange est dans une journée la machine fabrique n jouets selon un programme prédéfini et décrit par un vecteur $x \in \{100, 150\}^n$, où $x_i = 150$ ssi le i-ème jouet produit dans la journée est un camion. Odile doit emballer les jouets dans des cartons au fur et à mesure qu'ils sont produits, en respectant l'ordre de production. Pour cela elle dispose d'un stock inépuisable de k différents cartons. Le j-ème type de carton lui coûte p_j Euros à l'unité et peut contenir plusieurs jouets d'un poids total ne dépassant pas c_j grammes.

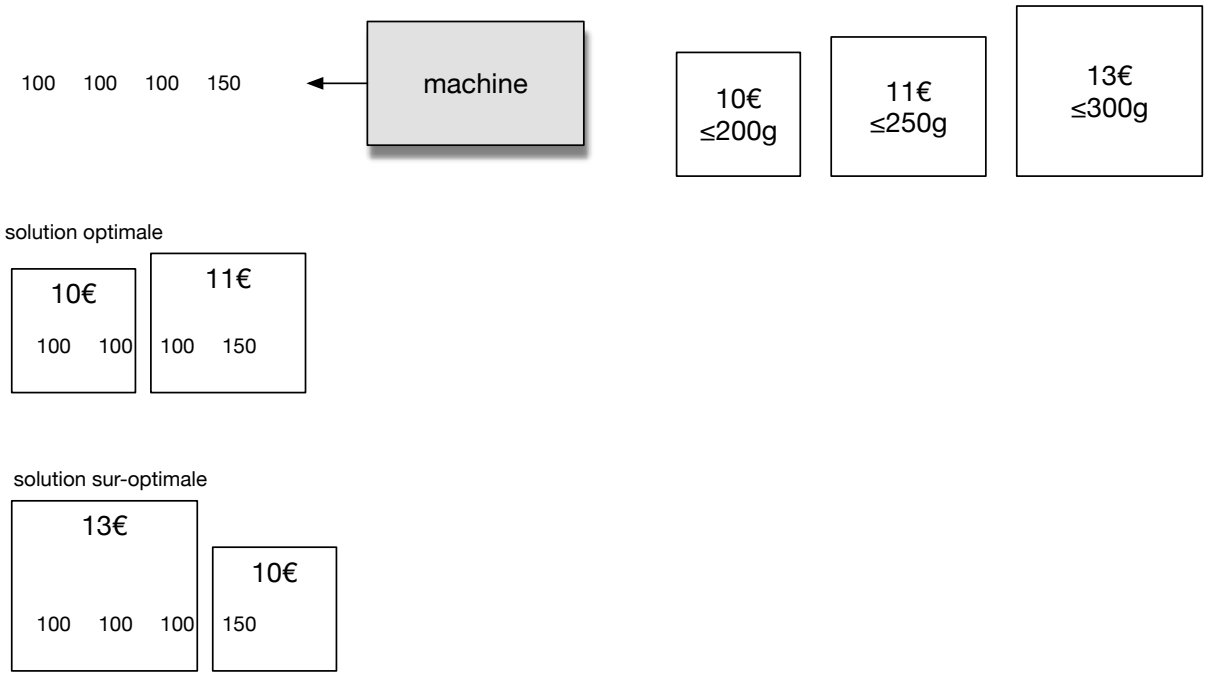

Figure 2: Un exemple au problème d'emballage de jouets. Odile pourrait avoir envie d'emballer les trois premières voitures dans un carton à 13 Euros, qui minimise le ratio p_j/c_j. Mais cette approche gloutonne finalement mène à une solution sur-optimale.

Étant données les entiers n, k avec $1 \leq k \leq n$ et les vecteurs $x \in \{100, 150\}^n$ et $p, c \in \mathbb{N}^k$ avec $c_1 \geq 150$, aidez Odile à déterminer le coût total minimal des cartons nécessaires pour emballer tous les jouets produits pendant la journée.

Il existe une solution de complexité en temps $O(n^2)$. Néanmoins des points partiels sont données pour des algorithmes de plus grande complexité.

complexité	points obtenus
$O(n^3)$	2 points
$O(n^2 k)$	3 points
$O(n^2 \log k)$	3 points
$O(n^2)$	4 points

2 Éléments de correction

2.1 Coefficients binomiaux

Notons $A_{n,k} = \binom{n}{k}$. Ceci compte le nombre de sélections possibles de k éléments parmi n, numérotés de 1 à n pour fixer les notations. Une telle sélection soit contient l'élément n, soit ne la contient pas. Le nombre de sélections ne contenant pas n est $A_{n-1,k}$, car l'absence de l'élément n est sans influence sur ces sélections. Le nombre de sélections contenant n est $A_{n-1,k-1}$, car il dépend de la sélection des $k-1$ éléments restants parmi les $n-1$ éléments restants. Nous avons alors la récursion suivante

$$A_{n,k} = A_{n-1,k} + A_{n-1,k-1}$$

avec comme cas de base $A_{n,0} = 1$ pour l'unique solution, l'ensemble vide, et $A_{n,n} = 1$ pour l'unique solution, l'ensemble $\{1, \ldots, n\}$. Il y a donc $O(n^2)$ variables à calculer, chacune s'évaluant en temps en $O(1)$ par cette récursion.

2.2 Subsetsum

Pour $0 \leq T \leq L$ et $0 \leq j \leq n$ soit $A_{T,j}$ un booléen, qui indique s'il existe un sous-ensemble de $S \subseteq \{1, \ldots, j\}$ avec $\sum_{i \in S} p_i = T$. Le cas de base est $A_{T,0}$ =Vrai ssi $T = 0$. Sinon pour $j \geq 1$, un tel ensemble soit contient j soit ne contient pas j. D'où la récursion $A_{T,j} = A_{T-p_j,j-1} \vee A_{T,j}$. En étend la notation à de valeurs négatives T en posant $A_{T,j}$ =Faux dans ce cas. La réponse au problème est $A_{L,n}$.

Il y a $O(Ln)$ variables, à évaluer dans l'ordre de j croissant, et pour j fixé dans l'ordre de T croissant. Chaque évaluation prend un temps constant, résultant dans la complexité attendue.

2.3 Emballage de jouets

L'hypothèse $c_1 \geq 150$ assure qu'il existe toujours une solution, consistant tout simplement en n cartons de type 1.

En général, une solution est décrite par une partition P des entiers $\{1, \ldots, n\}$ en intervalles. Durant l'intervalle $I \in P$ le poids total des jouets produits est $X_I := \sum_{i \in I} x_i$. Le coût pour emballer ces jouets est $P_I := \min\{p_\ell : c_\ell \geq X_I\}$, où le minimum de l'ensemble vide est défini comme étant $+\infty$.

Soit A_j le coût minimal pour emballer les j premiers jouets. Considérons une solution optimale de valeur A_j. Si $j = 0$ alors $A_0 = 0$, car il n'y a pas de coût généré par l'absence de jouets à emballer. Sinon le dernier carton contient les jouets produits durant l'intervalle $I := [i+1, \ldots, j]$ pour un certain $0 \leq i < j$. Le coût A_j se décompose alors en P_I et en le coût optimal pour emballer les jouets restants, à savoir A_i.

On a alors le programme dynamique suivant. Le cas de base est $A_0 = 0$ et pour tout $1 \leq j \leq n$ on a la récurrence $A_j = \min_{0 \leq i < j} A_i + W_{[i+1,j]}$. La réponse au problème est A_n.

Le programme dynamique comporte $O(n)$ variables et chacune est calculée en identifiant le minimum parmi $O(n)$ alternatives. Toute la difficulté consiste alors à calculer les coûts P_I de manière efficace. Une première approche naïve calcule P_I, pour un intervalle fixé I, en déterminant d'abord X_I par une somme sur $O(n)$ valeurs, et puis P_I par une évaluation de chacun des k cartons. Comme $k \leq n$, la première partie domine et nous avons un premier algorithme en temps $O(n^3)$.

Cette complexité peut être descendue à $O(n^2k)$ par l'observation suivante. Supposons que pour j fixé nous examinons les différentes alternatives pour A_j dans l'ordre $i = j-1, j-2, \ldots, 0$. Pour chaque i fixé, le poids $W_[i+1, j]$ peut-être calculé en temps $O(1)$, en prenant $100 + 50x_{i+1}$ auquel on ajoute $W_{[i+2,j]}$ calculé à l'itération précédente, si $i < j-1$. Ensuite P_I peut être calculé en examinant chacun des k cartons. Au total il y a $O(k)$ opérations à effectuer pour chacune des $O(n^2)$ paires (i, j), résultant en la complexité annoncée.

Imaginons maintenant que les cartons soient triés par capacité. Le temps $O(k \log k)$ nécessaire pour le tri est dominé par $O(n^2)$ et peut donc être ignoré. Imaginons qu'en plus de l'ordre $c_1 \ldots c_k$ on ait l'ordre $p_1 \leq p_k$. Cette propriété est facile à obtenir en supprimant de l'entrée les cartons ℓ avec $c_\ell \leq c_{\ell+1}$ mais $p_\ell > p_{\ell+1}$. En effet aucune solution optimale n'utiliserait un carton de type i, car son remplacement par un carton de type $i+1$ préserverait le respect des capacité et donnerait une solution strictement meilleure.

Désormais on pourrait déterminer P_I simplement par recherche dichotomique sur les k cartons, en cherchant le plus petit ℓ tel que $c_\ell \geq X_I$. Ceci donnerait une solution en temps $O(n^2 \log k)$.

Pour gagner un facteur $\log k$ dans la complexité, nous pouvons utiliser le fait que pour j fixé nous avons à faire cette recherche pour j différentes valeurs X_I de poids croissant. Il suffit alors de maintenir pendant cette recherche un indice ℓ sur le carton le moins cher ayant la capacité nécessaire. Quand la capacité est dépassée, ℓ est incrémenté successivement jusqu'à atteindre un carton de capacité suffisante. Au total il y a au plus k incrémentations, et le travail effectué par indice j fixé est en $O(n)$. La complexité finale est en $O(n^2)$.