Algorithmique avancée
TD Programmation Dynamique

J.M. Dischler

Chaînes de caractères

On considère le problème suivant : soit deux chaînes de caractères s et t. On se propose de comparer ces deux chaînes sachant que la première chaîne peut contenir des caractères spéciaux : $\sharp$, $\star$ et ?. Le $\sharp$ remplace un ou plusieurs caractères, le $\star$ zéro ou plusieurs caractères et le ? un caractère. La fonction de comparaison renvoie un booléen, indiquant si la première chaîne peut correspondre à la seconde (on parle de *pattern matching*).

1. Proposer un algorithme récursif. Montrer que la complexité au pire est exponentielle.
2. Proposer un algorithme basé sur le principe de la programmation dynamique. Quelle est sa complexité ?

Programmation dynamique : répartition des espaces en typographie

On considère le problème de la composition équilibrée d'un paragraphe dans un traitement de texte. Le texte d'entrée est une séquence de n mots de longueurs l_1, l_2, ..., l_n mesurées en nombre de caractères. On souhaite composer ce paragraphe de manière équilibrée sur un certain nombre de lignes contenant chacune exactement M caractères. Chaque ligne comportera un certain nombre de mots, les espaces nécessaires à séparer les mots les uns des autres (une espace [1] entre deux mots consécutifs) et des *caractères d'espacement supplémentaires* complétant la ligne pour qu'elle contienne exactement M caractères. La figure **??** présente un exemple de ligne contenant trois mots, deux espaces nécessaires à séparer ces trois mots, et six caractères d'espacement supplémentaires. Si une ligne donnée contient les

U	n		e	x	e	m	p	l	e		t	r	i	v	i	a	l							

FIGURE 1 – Une ligne de 24 caractères contenant 3 mots et 6 caractères d'espacement supplémentaires.

mots de i à j, où $i \leq j$, et étant donné que nous avons besoin d'une unique espace pour séparer deux mots consécutifs, le nombre de caractères d'espacement supplémentaires c nécessaires pour compléter la ligne est égal au nombre de caractères de la ligne (M), moins le nombre de caractères nécessaires pour écrire les mots ($\sum_{k=i}^{j} l_k$), moins le nombre de caractères nécessaires pour séparer les mots ($j - i$), autrement dit : $c = M - \sum_{k=i}^{j} l_k - (j - i)$. Le nombre c de caractères d'espacement supplémentaires doit bien évidemment être positif ou nul.

Notre critère « d'équilibre » est le suivant : on souhaite minimiser la somme, *sur toutes les lignes hormis la dernière*, des cubes des nombres de caractères d'espacement supplémentaires.

Travail minimum requis

Donnez un algorithme de programmation dynamique permettant de composer de manière équilibrée un paragraphe de n mots de longueurs l_1, ..., l_n données, et analysez la complexité de votre algorithme.

1. En typographie, « espace » est un mot féminin, comme vous le confirmera le premier dictionnaire venu.

Travail idéal

Vous pourrez, *mais ce n'est pas obligatoire*, établir une relation de récurrence définissant la composition optimale, proposer un algorithme récursif implémentant cette récurrence et montrer que sa complexité est médiocre, proposer un algorithme de programmation dynamique et analyser sa complexité, proposer un algorithme par recensement, et finalement proposer un contre-exemple à l'algorithme glouton naïf.

Indication pour l'établissement d'une récurrence

Comme l'on cherche ici un algorithme suivant le paradigme de la programmation dynamique, il est raisonnable de définir la composition optimale par une formule de récurrence. Pour ce faire, vous pourrez remarquer que dans un paragraphe de m lignes composé optimalement, les $(m-1)$ *dernières* lignes sont composées optimalement.