# Algorithmique avancée
# devoir en temps libre

Frédéric Vivien

## Travail demandé

Ce travail est à effectuer seul ou en binôme, à l'exclusion de tout regroupement de taille supérieure. Vous devez me rendre votre travail pour le lundi 14 janvier, soit directement, soit dans mon casier à la scolarité du département. La date du lundi 14 janvier ne pourra en aucun cas être dépassée : à partir du lendemain, vous pourrez retirer au secrétariat les annales du cours contenant les corrigés de tous les TDs et TPs et de ce devoir.

Vous tâcherez de proposer des solutions les plus simples et les plus esthétiques possibles. Tout vos algorithmes devront être expliqués et justifiés : un algorithme, même parfait, dont la justesse ne serait pas prouvée ne se verra attribuer que la moitié des points prévus par le barème.

À toutes fins utiles, je vous rappelle que les corrigés des TDs et TPs, ainsi que les notes de cours, sont disponibles à l'url : `http://icps.u-strasbg.fr/~vivien/Enseignement/Algo-2001-2002/`.

## Algorithme glouton : l'art de rendre la monnaie

On considère le problème consistant à rendre $n$ centimes (de franc ou d'euro) en monnaie, en utilisant le moins de pièces possible.

1. Décrivez un algorithme glouton permettant de rendre la monnaie en utilisant des pièces de cinquante, vingt, dix, cinq et un centime.
2. Démontrez que votre algorithme aboutit à une solution optimale.
3. On suppose que les pièces disponibles ont pour valeur $p^0$, $p^1$, ..., $p^k$ pour deux entiers $p > 1$ et $k \geq 1$ donnés. Montrez que l'algorithme glouton aboutit toujours à une solution optimale sur un tel jeu de pièces.
4. Donnez un ensemble de valeurs de pièces pour lesquelles l'algorithme glouton ne donne pas une solution optimale.

## Diviser pour régner : intervalle de plus grande somme

Nous avons un tableau $A$ de $n$ entiers relatifs. Nous recherchons un sous-tableau de $A$ dont la somme des éléments soit maximale. Autrement dit, nous recherchons un couple d'entiers $i$ et $j$, $1 \leq i \leq j \leq n$ tel que $\sum_{k=i}^{j} A[k]$ soit maximale. La figure 1 montre un exemple de tableau pour lequel les valeurs cherchées sont $i = 4$ et $j = 5$.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 5 | -8 | 6 | 5 | -9 | 3 | 4 |

FIG. 1 – Tableau dont l'intervalle de plus grande somme est défini par $i = 4$ et $j = 5$.

1. Proposez un algorithme naïf.
2. Quelle est sa complexité ?
3. Proposez un algorithme « diviser pour régner » découpant le tableau en deux moitiés.
4. 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 2 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 mots de $i$ à $j$, où $i \leq j$, et étant donné

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

FIG. 2 – Une ligne de 24 caractères contenant 3 mots et 6 caractères d'espacement supplémentaires.

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.

### 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.

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