Aix*Marseille université
Initiative d'excellence
Année 2019-20

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Bloc 5 : TP 6

Programmation dynamique 2

1 Différence entre deux fichiers

L'utilitaire `diff` sous Unix permet de comparer deux fichiers. Prenons un exemple. Dans `fichier1.txt`, à gauche, se trouve le poème *Le message* de Prévert. Dans `fichier2.txt`, à droite, une version du poème mal récitée par un élève qui n'a pas pris le temps de bien l'apprendre.

La porte que quelqu'un a ouverte	Le bois que quelqu'un traverse
La porte que quelqu'un a refermée	La rivière où quelqu'un se jette
La chaise où quelqu'un s'est assis	L'hôpital où quelqu'un est mort
Le chat que quelqu'un a caressé	L'hôpital où quelqu'un est mort
Le fruit que quelqu'un a mordu	La porte que quelqu'un a refermée
La lettre que quelqu'un a lue	La chaise où quelqu'un s'est assis
La chaise que quelqu'un a renversée	La lettre que quelqu'un a lue
La porte que quelqu'un a ouverte	La route où quelqu'un court encore
La route où quelqu'un court encore	
Le bois que quelqu'un traverse	Toto, qui a mal appris sa leçon
La rivière où quelqu'un se jette	
L'hôpital où quelqu'un est mort	
Jacques Prévert, Paroles	

La commande `diff fichier1.txt fichier2.txt` fournit alors le résultat suivant :

```
0a1,3
> Le bois que quelqu'un traverse
> La rivière où quelqu'un se jette
> L'hôpital où quelqu'un est mort
4,5d6
< Le chat que quelqu'un a caressé
< Le fruit que quelqu'un a mordu
7,8d7
< La chaise que quelqu'un a renversée
< La porte que quelqu'un a ouverte
10,12d8
< Le bois que quelqu'un traverse
< La rivière où quelqu'un se jette
< L'hôpital où quelqu'un est mort
14c10
< Jacques Prévert, Paroles
---
> Toto, qui a mal appris sa leçon
```

Les lignes ajoutées sont notées à l'aide de la lettre `a`, les lignes supprimées avec un `d` (pour *delete*), et les lignes modifiées par un `c` (pour *change*). Pour calculer les lignes à supprimer et les lignes à ajouter, il s'agit de se baser sur les parties *communes* dans les deux fichiers. Il faut donc trouver la plus longue partie commune des deux fichiers.

Pour simplifier, considérons plutôt le cas de la comparaison de deux chaînes de caractères (chaque caractère représentant une ligne du fichier précédent). On appelle sous-séquence d'une chaîne $s = s_0 s_1 \cdots s_{n-1}$ toute séquence $s_{i_0} s_{i_1} \cdots s_{i_{k-1}}$ telle que $0 \leq i_0 < i_1 < \cdots < i_{k-1} \leq n-1$: on note $r \triangleleft s$ si r est une sous-séquence de s. Les sous-séquences communes à deux séquences s et t sont $ssc(s,t) = \{r \mid r \triangleleft s \wedge r \triangleleft t\}$. Les plus longues sous-séquences communes sont regroupées dans l'ensemble $plssc(s,t)$. Par exemple, $plssc(\text{MARSEILLE}, \text{MASSILIA}) = \{\text{MASIL}\}$ et $plssc(\text{EMPIRE}, \text{MERE}) = \{\text{ERE}, \text{MRE}\}$.

Le problème est alors de trouver une des plus longues sous-séquences communes à deux séquences s et t, dont on notera m et n les longueurs respectives.

Question 1 Quelle est la complexité dans le meilleur des cas d'un algorithme qui essaierait naïvement toutes les possibilités ?

Question 2 Pour mieux faire, on se dirige vers une définition récursive de la plus longue sous-séquence commune : en effet, la plus longue sous-séquence commune de s et t est *presque* une plus longue sous-séquence commune aux séquences s' et t' obtenues en enlevant respectivement la première lettre de s et de t. S'inspirer de cela pour trouver une définition récursive de $plssc(s,t)$ dépendant de la première lettre de s et t.

Question 3 En déduire un algorithme de programmation dynamique qui stockera dans une matrice les résultats des calculs de $plssc(s_i s_{i+1} \cdots s_{m-1}, t_j t_{j+1} \cdots t_{n-1})$ nécessaires au calcul d'une plus longue sous-séquence commune à $s_0 s_1 \cdots s_{m-1}$ et $t_0 t_1 \cdots t_{n-1}$. Quelle est sa complexité ?

Question 4 Que changer si l'on cherche à calculer l'ensemble des plus longues sous-séquences communes, plutôt que l'une d'entre elles ?

Pour poursuivre... À partir de cet algorithme de plus longue sous-séquence commune, comment calculer le `diff` entre deux chaînes de caractères, voire de deux fichiers ?

2 Recherche d'une sous-liste maximum

Le problème de la sous-liste maximum consiste à trouver au sein d'une liste L, une suite d'éléments contigus dont la somme est maximale, c'est-à-dire deux indices $0 \leq i \leq j \leq n$ (avec n la longueur de la liste) qui rendent maximale la somme $L[i] + L[i+1] + \cdots + L[j-1]$. La somme est nulle lorsque $i = j$.

Question 5 Expliquer ce que calcule le programme suivant, utilisant des définitions par compréhensions et la fonction `sum` calculant la somme des éléments d'une liste :

```
{(i,j): sum(L[i:j]) for j in range(len(L) + 1) for i in range(j + 1)}
```

Question 6 En vous inspirant du programme précédent, proposer un programme naïf calculant la somme de la sous-liste maximum (on ne demande pas de renvoyer les indices i et j, mais juste la somme maximale). Quelle est sa complexité ?

Question 7 Afin d'obtenir un algorithme plus rapide, on souhaite utiliser un algorithme de programmation dynamique. Il faut donc une équation caractérisant la somme de la sous-liste maximum. On note $maxsegsum(L,k)$ la somme maximale des sous-listes de $L[0:k]$ (avec k inférieur ou égal à la longueur de L). On définit donc $maxsegsum(L,0) = 0$, puisque $L[0:0]$ est la liste vide. Expliquer alors pourquoi **on n'a pas**

$$maxsegsum(L,k) = \max(maxsegsum(L,k-1), maxsegsum(L,k-1) + L[k-1])$$

Question 8 On introduit donc une nouvelle notation : $maxsufsum(L,k)$ calcule la somme maximale d'une sous-liste suffixe de $L[0:k]$, c'est-à-dire la somme $L[i] + L[i+1] + \cdots + L[k-1]$ maximale pour $0 \leq i \leq k$. Compléter les équations suivantes pour qu'elles caractérisent $maxsufsum$:

$$maxsufsum(L,0) = ? \qquad \text{et} \qquad \forall k > 0 \quad maxsufsum(L,k) = \max(?, maxsufsum(L,k-1) + ?)$$

Question 9 En déduire une équation de récurrence caractérisant $maxsegsum$.

Question 10 Écrire un nouveau programme Python calculant la somme de la sous-liste maximum, en vous aidant des équations de récurrence précédentes. Quelle est sa complexité ?

Question 11 Comment modifier le programme pour qu'il renvoie une sous-liste de somme maximale plutôt que sa somme ?