Programmation efficace – Licence 3 Informatique
Ecrasons les chaînes !

# Problème 1

Pour s'échauffer, on considère le problème suivant : nous disposons d'une chaîne de caractères $s$ sur un alphabet $\Sigma$, et de $n$ chaînes de caractères $s_1 \ldots s_n$ (toujours sur $\Sigma$).
L'objectif est de trouver une plus petite sous-chaîne de $s$ qui contient tous les $s_i$ ($s_1, \ldots s_n$).

On souhaite donc construire un algorithme qui renvoie la sous-chaine optimale, sa longueur ainsi que son indice de début (dans $s$), et $\perp$ si cette sous-chaîne n'existe pas.

Voici quelques exemples et leurs solutions...
Prenons $s =$ `aabcaaadabbbccabadaaabaaacd`

1. si $s_1 =$ `ab`, $s_2 =$ `c` et $s_3 =$ `d`, alors le résultat sera (`cabad`, 5, 13), c-à-d. la sous-chaîne minimale de $s$ contenant les 3 chaînes $s_i$ est de longueur 5 et commence à la position 13.
2. si $s_1 =$ `ad`, $s_2 =$ `da` et $s_3 =$ `ab`, alors le résultat sera (`adab`, 4, 6).
3. si $s_1 =$ `add`, $s_2 =$ `ab` et $s_3 =$ `a`, alors le résultat sera $\perp$.

L'alphabet $\Sigma$ est inclu dans $\{a, \ldots, z, A, \ldots, Z\}$.

## Formats

Une instance du problème est donnée dans un fichier. On n'utilisera que des lignes de 80 caractères : une chaîne pourra donc être stockée sur plusieurs lignes. Chaque chaîne se terminera par le caractère # (qui n'est pas dans l'alphabet $\Sigma$).

Un fichier d'entrée a la structure suivante : la première ligne contient l'entier $n$, puis les lignes suivantes contiennent la description des $n+1$ chaînes de caractères (en suivant les convention ci-dessus) dans l'ordre suivant : $s_1, \ldots, s_n, s$.

Par exemple, le premier exemple ci-dessus sera décrit avec :

```
3
ab#
c#
d#
aabcaaadabbbccabadaaabaaacd#
```

Pour le format de sortie, on souhaite récupérer un fichier contenant d'abord la longueur de la sous-chaîne trouvée, son indice dans $s$ puis la sous-chaîne elle-même affichée selon le format précédent (par ligne de 80 caractères). Et dans le cas où il n'y a pas de solution, un 0 sur la première ligne et un # sur la seconde.

# Problème 2

Pour ce second problème, nous disposons de $n$ chaînes de caractères $s_1 \ldots s_n$ sur un alphabet $\Sigma$ et l'objectif est de trouver une plus petite chaîne sur $\Sigma$ qui contient tous les $s_i$ ($s_1, \ldots s_n$).

On souhaite donc construire un algorithme qui renvoie une chaine optimale (avec sa longueur).

Voici quelques exemples et leurs solutions...

1. si $s_1 =$ `ab`, $s_2 =$ `c` et $s_3 =$ `d`, alors un résultat possible serait `abcd`.
2. si $s_1 =$ `ad`, $s_2 =$ `da` et $s_3 =$ `ab`, alors le résultat sera `adab`
3. si $s_1 =$ `add`, $s_2 =$ `dda` et $s_3 =$ `a`, alors le résultat sera `adda`.

## Formats

On utilise le même format que précédemment pour les fichiers, on omet juste la dernière chaîne $s$. On suppose que l'alphabet est défini implicitement par les lettres utilisées dans les chaînes. Pour le format de sortie, on souhaite récupérer un fichier contenant d'abord la longueur de la chaîne trouvée, puis la chaîne elle-même affichée selon le format précédent (par ligne de 80 caractères).

**Toujours plus fort.** Et on considère le problème suivant : étant données deux séquences de chaînes de caractères $s_1, \ldots, s_n$ et $t_1, \ldots, t_m$, on cherche une plus petite chaîne qui contient tous les $s_i$ mais aucun $t_i$. Une idée ?

Pour le format, on donnera les deux entiers $n$ et $m$ sur la première ligne du fichier, et les $n+m$ chaînes ensuite.