TP-Cours d'Informatique 03

Boucles imbriquées [Semaine 07]

ITC – PCSI^2 2024-2025 – Lycée Fabert (Metz) – L. Parise – D.Mengel – S.Limal

Après avoir découvert les listes, les dictionnaires, nous allons nous intéresser à deux algorithmes classiques : la **recherche d'un motif dans un texte** et la **recherche des valeurs les plus proches dans une liste**.

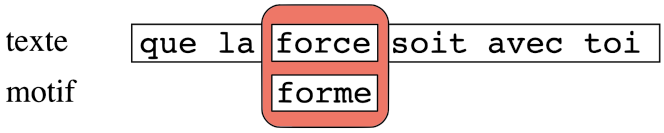
I. Recherche d'un motif dans un texte

La recherche de motifs dans un texte est un problème important qui apparaît dans de nombreux domaines scientifiques. En informatique, on le rencontre naturellement dans l'édition de textes, en analyse syntaxique ou en recherche d'information (comme sur votre moteur de recherche internet préféré).

Prenons l'exemple du texte `rechercher`. Celui-ci contient deux occurences du motif `cher`. Même si ce problème est simple, il existe de nombreux algorithmes permettant d'effectuer une telle recherche. Nous nous intéressons ici à l'algorithme « naïf ».

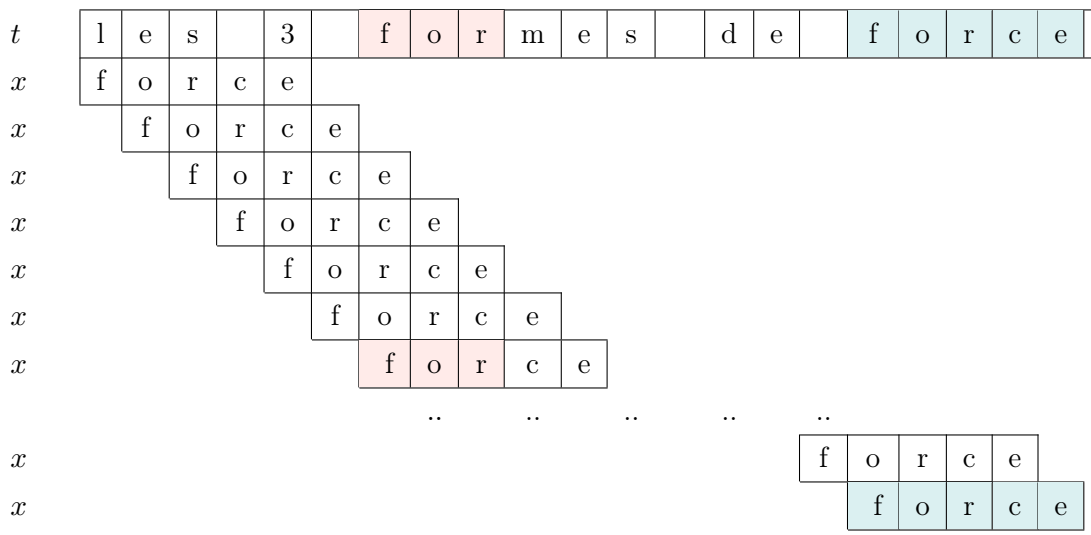
Posons $t = t_0 \cdots t_{n-1}$ et $x = x_0 \cdots x_{m-1}$ où les t_j et les x_i sont des lettres d'un alphabet. Nous nous proposons de déterminer si x est «facteur» de t et le cas échéant, de trouver une occurence de x dans t. L'algorithme que nous présentons ici est bâti sur le schéma suivant : le motif x est comparé aux facteurs $t_i \cdots t_{i+m-1}$ de longueur m de t jusqu'à trouver une coïncidence, si elle existe. L'efficacité d'un tel algorithme de recherche se mesurera en nombre de comparaisons de caractères.

L'algorithme naïf fait cette comparaison **pour toutes les positions i possibles**. De façon imagée, la méthode consiste à faire glisser le motif x le long de la chaîne t et à comparer x au bloc de t couvert.



Pour ce faire, on utilise un compteur j qui va décrire tous les indices des facteurs du motif x soit $[\![0; m-1]\!]$ et pour chaque valeur de j, on compare $x[j]$ à $t[i+j]$. En cas d'égalité, le compteur j est incrémenté de 1 pour continuer la comparaison de x avec les facteurs de t qui suivent le dernier facteur testé. En cas d'échec, on reprend la comparaison de x à partir de son premier facteur $x[0]$ que l'on comparera avec $t[i+1]$ en posant $j = 0$, un compteur de valeur i étant incrémenté de 1.

Le schéma suivant montre comment le motif $x =$`"force"` « glisse » sous le texte $t =$`"les 3 formes de forces"` et les cases en couleur montrent les caractères qui se correspondent lors des comparaisons.



On donne à présent l'algorithme de la fonction rechercheMotif dont nous venons de décrire le fonctionnement :

Algorithme 1 Recherche de la 1re occurrence d'un motif dans un texte

Fonction rechercheMotif(x, t : **Chaînes de caractères**) : Entier

Rôle : Déterminer l'indice de la première occurrence du motif x dans le texte t le cas échéant et indiquer que x n'est pas facteur de t sinon en retournant la valeur -1.

Déclaration : m, n, i, j : **Entiers**

Début

```
m ← longueur du motif x
n ← longueur du texte t

pour i de 0 à indiceMax faire
    j ← 0
    tant que condition1 and condition2 faire
        j ← j + 1
    fin tant que

    si condition3 alors
        retourner i
    fin si
fin pour
```

Fin Fonction

1. Le motif x est de longueur m. Étant donné que x « glisse » sous le texte t, déterminer la valeur de indiceMax qui correspond à la dernière position de x sous le texte t dont le dernier caractère est $t[n-1]$. Expliquer.

2. Déterminer les conditions condition1 et condition2 permettent :
   - de s'assurer que l'indice j ne dépassera pas l'indice du dernier caractère $x[m-1]$ de x
   - de vérifier que $x[j]$ coïncide avec le caractère de t sous lequel il est placé quand $x[0]$ est sous $t[i]$, avant d'incrémenter j de 1 pour passer à la comparaison suivante.

3. Justifier de l'intérêt particulier à écrire condition1 et condition2 dans un certain ordre et non dans l'ordre inverse. Dans quelle situation est-ce important ?

4. Que vaut la variable j à la sortie de la boucle tant que si le motif x est effectivement présent dans le texte t en position i (c'est-à-dire $x[0]$ correspondant à $t[i]$) ? En déduire l'expression de condition3.

5. Écrire la fonction rechercheMotif en langage Python en s'appuyant sur l'algorithme précédent.
6. Modifier la fonction rechercheMotif pour qu'elle retourne la liste des indices des occurrences d'un motif x dans un texte t. On fera des essais sur le fichier `spleen.txt` contenant un célèbre poème de Baudelaire.

   À cet effet, on rappelle comment récupérer le contenu du fichier texte `spleen.txt` en le stockant dans la chaîne de caractère t :

```
file = open("spleen.txt",'r',encoding="utf-8")
t = file.read()
file.close()
```

La syntaxe du poème de Baudelaire est marquée par l'anaphore de « Quand... Et que », qui renvoie les propositions principales aux deux dernières strophes :

Strophe 1	**Strophe 2**	**Strophe 3**	**Strophe 4**	**Strophe 5**
Quand	Quand	Quand	Des cloches ... sautent	Et défilent
Et que	Où	Et qu'un	Et lancent	l'Espoir pleure
		Ainsi que		Et l'Angoisse

L'attente ainsi provoquée est soulignée par l'effet des images, de la versification ... **Bref**, recherchez la liste des occurrences de `"Quand"` et `"Et"` et `"Et qu"`. On effectuera une vérification rapide des résultats obtenus en utilisant le slicing/découpage en tranche.

II. Recherche des deux valeurs les plus proches dans une liste d'entiers

Soit L une liste d'entiers de longueur $n \in [\![2; +\infty[\![$. On souhaite identifier le couple d'indices (imin,jmin) tel que les valeurs L[imin] et L[jmin] soient les plus proches parmi celles de la liste, soit :

$$|\text{L[imin]} - \text{L[jmin]}| = \min\{|\text{L[i]} - \text{L[j]}|,\ 0 \leqslant i < j \leqslant n-1\} \quad (*)$$

Par exemple, pour la liste ci-dessous, les valeurs les plus proches sont celles d'indices respectifs imin=1 et jmin=3.

L =	1	12	34	8	43	56

1. Écrire la fonction plusProches(L:**list**)→(**int**,**int**)
   - qui prend comme argument une liste d'entiers L avec **len**(L)⩾ 2
   - qui retourne le tuple d'entiers (imin,jmin) correspondant à un couple d'indices des deux valeurs les plus proches dans la liste L.

   Votre fonction sera constituée de deux boucles **for** imbriquées portant sur les indices i et j. On utilisera la définition $(*)$ de imin et jmin pour déterminer les intervalles dans lesquels devront évoluer i et j.
2. Vérifier votre algorithme en appliquant votre fonction plusProches à la liste L=[1,12,34,8,43,56].
3. Exprimer en fonction de n=**len**(L) le nombre de tests (**if**) effectués par l'appel plusProches(L).