Informatique

TP X1

Algorithme de Rabin-Karp

1 Introduction

Dans ce TP, on s'intéresse à la recherche de chaînes de caractères dans un texte. Nous allons revoir l'algorithme naïf de recherche de motifs vu en cours, puis l'algorithme de Rabin-Karp, qui repose sur un codage des chaînes de caractères par des entiers.

1.1 Chaînes de caractères

Les **chaînes de caractères** permettent de représenter du texte. Pour créer une chaîne de caractères, on place simplement le texte entre **doubles guillemets droits** (« " »).

Exemple 1.1. *Dans la console :*

```
>>> "Bonjour"
'Bonjour'
>>> "123 Bonjour"
'123 Bonjour'
>>> "123"
'123'
```

On peut voir les chaînes de caractères Python comme des listes non modifiables.

Exemple 1.2.

```
>>> s = "bonjour"
>>> s[0]
'b'
>>> s[1]
'o'
>>> s[7]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> s[0] = 'a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

1.2 Lecture de fichiers en Python

Afin de tester les algorithmes de recherche de chaînes de caractères, on pourra utiliser des fichiers texte (crées par exemple avec l'utilitaire **NotePad++**). Pour tester les algorithmes sur des textes contenus dans des fichiers, il faut pouvoir ouvrir les fichiers et convertir leur contenu en chaînes de caractères. Comme la plupart des langages de programmation, Python propose des primitives pour effectuer ces opérations.

Pour obtenir comme chaîne de caractères le contenu d'un fichier appelé « `text.txt` », on peut utiliser la commande suivante :

```
open("text.txt").read()
```

Par exemple :

```
>>> s = open("text.txt").read()
>>> s[0]
```

Remarque 1.3 (Ouverture de fichiers sous Windows)**.** *Pour ouvrir un fichier dans Python sous Windows, il faut remplacer les « \ » par des « / » dans le chemin d'accès.*

Par exemple, pour ouvrir le fichier

```
\\servdfs\users\home\criba\Desktop\essai.txt
```

il faut écrire

```
>>> open("//servdfs/users/home/criba/Desktop/essai.txt").read()
```

1.3 Notations

Nous utiliserons les notations suivantes dans tout le TP. On considère deux chaînes de caractères : un motif M de longueur m et un texte T de longueur t.

On cherche, s'il existe, l'indice de la première occurrence de M dans T, c'est-à-dire, en utilisant les notations Python, le plus petit $i \leq t-1$ tel que

$$\forall j = 0, \ldots, m-1, \quad M[j] = T[i+j] \tag{$*$}$$

Question 1.1. *Donner, en fonction de t et m, la valeur maximale possible de i dans ($*$).*

2 Algorithme naïf

Question 2.1. *Rappeler l'algorithme naïf de recherche de motifs. Le tester sur un fichier texte de votre choix.*

3 Algorithme de Rabin-Karp simplifié

Un défaut de l'algorithme naïf de recherche de motifs est le coût de la boucle utilisée pour tester ($*$). En effet, cette boucle doit parcourir (presque) toute la chaîne M et ce pour (presque) tous les indices de la chaîne T. L'algorithme de Rabin-Karp utilise une représentation des chaînes des caractères par des entiers qui permet de tester ($*$) beaucoup plus efficacement. Nous allons tout d'abord voir une version simple de cet algorithme.

3.1 Idée

Rappelons que toutes les données manipulées par un ordinateur sont au bout du compte représentées par des suites de bits. Par exemple, la chaîne de caractères

`"Bonjour"`

est représentée à l'aide d'une suite de mots mémoire, chacun de ces mots représentant un caractère de la chaîne.

Il existe de nombreux formats (ou conventions) pour représenter des caractères par des mots de bits. Parmi les plus courants, citons par exemple les formats `UTF-8` (`Unicode`), la famille de formats `Western` (avec notamment les formats `ISO 8859-1` et `ISO 8859-15`), les formats spécifiquement définis pour les systèmes Windows (par exemple `Windows-1252`).

En général (et en particulier pour les formats cités ci-dessus), les caractères sont représentés sur 8 bits, ce qui fait qu'un seul mot mémoire (typiquement de 32 ou 64 bits) peut concrètement représenter plusieurs caractères.

Quoi qu'il en soit, les 8 bits servant à représenter un caractère peuvent aussi être vus comme représentant un entier. On dispose ainsi d'une fonction Python `ord` associant à chaque caractère un entier. Par exemple :

```
>>> ord('a')
97
```

L'idée de l'algorithme de Rabin-Karp simplifié est la suivante :

— On choisi un entier $k > 0$ tel tous les caractères peuvent être codés par des entiers parmi

$$\{0, \ldots, k-1\}$$

(on prendra typiquement $k = 2^8 = 256$).

— Le motif M (de longueur m) que l'on cherche est codé par l'entier représentant le nombre base k correspondant :

$$\text{reprM} \quad = \quad M[0] \cdot k^{m-1} + M[1] \cdot k^{m-2} + \cdots + M[m-2] \cdot k + M[m-1]$$

— Au fur et à mesure que l'on parourt les indices de T (avec la variable i), la fenêtre courante $T[i] \ldots T[i+m-1]$ est codée par l'entier représentant le nombre base k correspondant :

$$\text{reprT} \quad = \quad T[i] \cdot k^{m-1} + \cdots + T[i+m-1]$$

— Ensuite, l'algorithme procède de manière très similaire à l'algorithme naïf :

1. On parcourt les indices de T, et pour chaque indice i de T (avec $i \leq t-m$), on teste si reprM = reprT.
2. Si l'égalité est vraie, alors on renvoie i,
3. Sinon, on passe à l'indice suivant dans T, et on met à jour la représentation reprT de la fenêtre $T[i] \ldots T[i+m-1]$.

3.2 Représentation des chaînes de caractères par des entiers

Question 3.1 (Calcul de la représentation de la fenêtre courante)**.** *Pour $i = 0, \ldots, t-m-1$, on note*

$$\mathit{repr}_i \quad = \quad T[i] \cdot k^{m-1} + \cdots + T[i+m-1]$$

Comment calculer repr_{i+1} en fonction repr_i ?

Question 3.2 (Valeurs initiales de reprM et reprT)**.** *Écrire une fonction Python qui prend en arguments une chaîne de caractères* `s` *et un entier* `k`*, et qui utilise la méthode de Horner pour calculer l'entier*

$$\mathtt{s}[0] \cdot \mathtt{k}^{m-1} + \mathtt{s}[1] \cdot \mathtt{k}^{m-2} + \cdots + \mathtt{s}[m-2] \cdot \mathtt{k} + \mathtt{s}[m-1]$$

où m est la longueur de `s`*.*

3.3 Algorithme

Question 3.3. *Écrire une fonction Python qui implémente un algorithme de recherche de chaînes de caractères utilisant la représentation des chaînes de caractères par des entiers (et donc prennant comme arguments une chaîne de caractères et l'entier* `k`*). La tester sur un fichier texte de votre choix.*

4 Algorithme de Rabin-Karp

4.1 Idée

L'algorithme simplifié vu au §3.3 souffre d'une limitation dûe à la taille des entiers. En effet, si on se base sur les codages standards de chaînes de caractères sur 8 bits alors les codages reprM et reprT doivent se faire base 256 (car $2^8 = 256$). Pour un motif de longueur m, on peut donc avoir à utiliser des nombres de l'ordre de 256^m. On arrive vite à des valeurs très grandes pour reprM et reprT :

- pour un motif de 5 caractères on peut avoir reprM et reprT de l'ordre de $256^5 = 1\,099\,511\,627\,776 \simeq 10^{12}$
- pour un motif de 10 caractères on peut avoir reprM et reprT de l'ordre de $256^{10} = 1\,208\,925\,819\,614\,629\,174\,706\,176 \simeq 10^{24}$.

D'autre part, avec la représentation Python des entiers relaltifs, sur k bits on peut représenter des entiers positifs jusqu'à $2^{k-1} - 1$, soit

- sur 32 bits $2^{31} - 1 = 2\,147\,483\,647 \simeq 10^9$,
- sur 64 bits $2^{63} - 1 = 9\,223\,372\,036\,854\,775\,807 \simeq 10^{18}$.

Ainsi, pour des motifs de 5 caractères, sur une machine 32 bits les entiers rerprM et reprT peuvent nécessiter plusieurs mots mémoire, de même que pour des motifs de longueur 10 sur une machine 64 bits.

L'idée de l'algorithme de Rabin-Karp est de représenter les entiers rerprM et reprT **modulo** p, pour un p bien choisi (typiquement inférieur à 2^{31} pour des machines 32 bits et inférieur à 2^{63} sur 64 bits), de manière à pouvoir tester avec un coût unitaire si reprM = reprT.

4.2 Algorithme de Rabin-Karp

Dans l'algorithme de Rabin-Karp, lorsque reprT = reprM, on doit effectivement vérifier si on a bien trouvé une occurrence. En effet, comme reprM et reprT sont des entiers modulo p, on peut avoir reprM = reprT sans pour autant avoir trouvé une occurrence de M dans T.

Question 4.1. *Écrire une fonction Python qui implémente l'algorithme de Rabin-Karp. Cette fonction doit prendre comme arguments les chaînes de caractères M et T, ainsi que les entiers k et p.*

La tester sur un fichier texte de votre choix.