NSI Terminale - Algorithmie

Résumé : recherche textuelle

qkzk

2020/06/29

Recherche textuelle

Pourquoi parler de recherche textuelle ?

Qu'est-ce qu'un texte ?

Quelques exemples :

- 101010101010001
- ATCGTTTATGCGAA
- un texte
- la concaténation de toutes les pages web

Définition

Un texte est une suite finie de symboles.

Recherche dans un texte connu à l'avance (livres, sites...)

On dispose alors généralement d'un **index**

.index

L'index peut-être vu comme un *dictionnaire* : on repère la clé qui nous intéresse et sa valeur nous indique la position du motif.

L'usage est alors simpliste et peu coûteux, tout le travail a été réalisé en amont.

Recherche des occurences d'un motif dans un texte.

L'objectif est de retourner les positions du *motif* (exemple 'Robert') dans le texte : 'Bonjour Robert, ça va Robert ? Robert Robert !'

L'algorithme doit retourner : `[8, 22, 30, 37]`

Recherche naïve d'un motif dans un texte

Puis-je trouver le mot `P = atatac` ? dans `T = a t a g a c a c a a t a t a c t g a c a c g a t`

Tester la présence de `P` à chaque position de `T`

Au pire : $|T| \times |P|$ comparaisons.

Algorithme de Boyer-Moore-Horspool

Dernière occurrence

On commence par créer un tableau associant chaque caractère possible à la longueur du motif.

Ensuite, pour chaque caractère d'indice `i` du motif, la distance est donnée par `taille - 1 - i`

ce qui donne :

pseudo code : dernière occurrence

```
dernière occurrence (motif)
  m = longueur du motif
  créer un dictionnaire associant chaque lettre à m
  pour i allant de 0 à m-2,
      dictionnaire [ motif[i] ] = m - 1 - i
  fin du pour
  retourner le dictionnaire
```

Boyer-Moore-Horspool

- on commence avec `j` = 0
- on itère jusqu'à ce que `j = taille du texte - taille du motif`

  on parcourt le motif à partir de la fin, donc `i = taille du motif`.

  on recule sur `i` jusqu'à arriver à 0 ou jusqu'à ce que les caractères ne se correspondent plus.

  - si `i = -1` alors

    le motif commence en j et on augmente `j` de 1

  - sinon

    on augmente `j` de la distance correspondant à cette position différente dans le texte.

Pseudo-code Boyer-Moore-Horspool

```
Algorithme Boyer-Moore-Horspool(x, t):
  '''
  x : motif, t : texte, m : longueur motif, n : celle du texte
  d : tableau des dernières occurrences du motif
  '''
  tant que j <= n - m,
    i = m - 1
    tant que i >= 0 et t[j + i] = x[i]:
      i = i-1
    fin tant que
    si i = -1 alors
      j est une occurrence de x
      j = j + 1
    sinon
      j = j + d[ t[j + i] ]
    fin du si
  fin du tant que
```

Compléments

Cet algorithme comporte deux des trois idées principales de la version complète, dîtes de *Boyer-Moore* :

1. comparer en parcourant le motif par la droite,
2. utiliser un tableau de distances pré-calculé sur les motifs,
3. utiliser un autre tableau, dît du bon préfixe.