Cours d'algorithmique
Recherche de chaînes de caractères - EISTI - ING 1

Ecole Internationale des Sciences du Traitement de l'Information

Recherche de chaînes de caractères

Généralités

- Problème fréquent dans les éditeurs de texte.
- Des algorithmes efficaces permettent d'améliorer la réactivité des programmes d'éditions.
- Egalement utilisé pour analyser les séquences ADN.

Formalisation

- Texte : $T(1..n)$
- Motif : $P(1..m)$

Recherche de chaînes de caractères

Définition

Le motif P apparaît avec un décalage s dans le texte T (ou apparaît à la position $s+1$ dans T) si :
$0 \leq s \leq n-m$ et $T(s+1..s+m) = P(1..m)$.

Problématique

Le problème de la recherche de chaînes de caractères revient donc à trouver tous les décalages valides pour lesquels un motif P donnée apparaît dans un texte T donné.

Algorithmes de recherche de chaînes

Comparaisons

Algorithmes	Temps de pré-traitement	Temps de recherche
naïf	0	$O((n-m+1)m)$
Rabin-Karp	$\Theta(m)$	$O((n-m+1)k), k \leq m$
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$

Définitions

- w est un préfixe d'une chaîne x si $x = wy$, noté $w \sqsubset x$
- w est un suffixe d'une chaîne x si $x = yw$, noté $w \sqsupset x$

Lemme de recouvrement des suffixes

Si x, y, z des chaînes telles que $x \sqsupset z$ et $y \sqsupset z$.

- Si $|x| \leq |y|$ alors $x \sqsupset y$
- Si $|x| \geq |y|$ alors $y \sqsupset x$
- Si $|x| = |y|$ alors $x = y$

Algorithme naïf

```
procedure rechercheNaive(E T:tableau(n):caractere,E P:tableau(m):
      caractere)
  variables
    s:entier
  pour s←0 a n-m pas 1
    si P(1..m)=T(s+1..s+m) alors
      ecrire("La chaîne apparaît avec le décalage"+s)
    fsi
  fpour
fprocedure
```

Complexité

- $P(1..m) = T(s+1..s+m) : O(m)$
- pour ... : $O(n-m+1)$
- Complexité totale : $O((n-m+1)m) \equiv O(n^2)$ si $m = n/2$
- Algorithme non optimal : par exemple pour $P = aaab$, si on trouve que $s = 0$ est valide, alors on peut prédire qu'aucun des décalages $1, 2, 3$ ne sera valide car $T(4) = b$.

Algorithme de Rabin-Karp

Idée

- Codage des mots en base d :
  $p = P(1)*d^{m-1}+P(2)*d^{m-2}+\cdots+P(m-1)*d+P(m)$
- Pour tout décalage s de $(0, n-m)$:
  $t_s = T(s+1)*d^{m-1}+T(s+2)*d^{m-2}+\cdots+T(s+m)$
- ts + 1 peut se calculer facilement à partir de t_s :
  $t_{s+1} = (t_s - T(s+1)*d^{m-1})*d + T(s+m+1)$

Algorithme de Rabin-Karp

Exemple

alphabet : $\{A, T, G, C\}$.
codage : $A = 0, T = 1, G = 2, C = 3$ donc $d = 4$
Pour $T = ACACGTT$:, si on cherche un motif de taille 3, les décalages sont :

$$t_0 = 0 * d^2 + 3 * d + 0 = 12$$
$$t_1 = 3 * d^2 + 0 * d + 3 = 51$$
$$t_2 = 0 * d^2 + 3 * d + 2 = 14$$
$$t_3 = 3 * d^2 + 2 * d + 1 = 57$$
$$t_4 = 2 * d^2 + 1 * d + 1 = 37$$

Algorithme de Rabin-Karp

Algorithme

```
procedure rabin-karp(tableau P(M):entier,tableau T(N):entier,
                     d:entier,m:entier,n:entier)
  variables
    h,m,t,i,s:entier
  h ← d^(m-1)
  p ← 0
  t ← 0
  pour i ← 1 a tm pas 1
    p ← p + d^(m-i) * P(i)
    t ← t + d^(m-i) * T(i)
  fpour
  pour s ← 0 a n-m-1 pas 1
    si p=t alors
      ecrire("decalage valide: "& s)
    fsi
    t ← (t - T(s+1) * h) * d + T(s+m+1)
  fpour
  si p=t alors
     ecrire("decalage valide: "& s)
  fsi
fprocedure
```

Algorithme de Rabin-Karp

Complexité

- calcul de h en $O(\log(m))$
- calculs de p et t initial en $O(m)$
- $(n-m)$ calculs de t : $(n-m) * O(1)$

Dans le pire des cas : $O(n-m)$, algorithme optimal !

Algorithme de Rabin-Karp

Problèmes

Si d et/ou m sont grand, les nombres manipulés sont gigantesques ! !
$26^{10} = 121167095653376$

Idée (cf TD)

Effectuer les calculs modulo un entier ni trop grand ni trop petit.
Attention, il risque d'y avoir des décalages non valides (même valeur du modulo pour deux chaînes différentes).

Fonction préfixe d'une chaîne de caractères

Fonction préfixe : $pi(q) = max\{k : k < q \text{ et } P_k \sqsupset P_q\}$

$pi(q)$ est la longueur du plus long préfixe de P qui est un suffixe propre de P_q.

```
fonction calculPrefixe(E tableau P(M):caractere,m:entier):tableau():
      entier
  variables
    q,k:entier
    tableau pi():entier
  creerTableau(pi,m)
  pi(1) ← 0
  k ← 0
  pour q ← 2 a m pas 1
    tantque k>0 et P(k+1) ≠ P(q) faire
      k ← pi(k)
    ftq
    si P(k+1)=P(q) alors
      k ← k+1
    fsi
    pi(q) ← k
  fpour
  retourner pi
ffonction
```

Exemple

i	1	2	3	4	5	6	7	8	9	10
P(i)	a	b	a	b	a	b	a	b	b	a
pi(i)	0	0	1	2	3	4	5	6	0	1

Algorithme KMP (Knuth-Morris-Pratt)

```
procedure KMP(E tableau T(N):caractere,E tableau P(M):caractere,
              n:entier,m:entier)
  variables
    tableau pi():entier
    i,q:entier
  pi ← calculPrefixe(P,m)
  q ← 0   //nb de caracteres concordant
  pour i ← 1 a n pas 1  //balaie le texte de gauche a droite
    tant que q>0 et P(q+1) ≠ T(i) faire
      q ← pi(q)  //prochain caractere ne concorde pas
    ftq
    si P(q+1)=T(i) alors
      q ← q+1    //prochain caractere concorde
    fsi
    si q=m alors  //est ce que tout P a concordé?
      ecrire("le motif apparaît en position"+(i-m))
      q ← pi(q)  //chercher prochaine correspondance
    fsi
  fpour
fprocedure
```

Exemple pour l'algorithme KMP

Motif = ABCDABD pi :

0	0	0	0	1	2	0

Texte = ABC ABCDAB ABCDABCDABDE

k=0, **ABC** ABCDAB ABCDABCDABDE

ABCDABD

On constate que la 4ème lettre diffère donc on passe directement au décalage k=4 (k=1 si naïf)

Exemple pour l'algorithme KMP

Motif = ABCDABD pi :

0	0	0	0	1	2	0

Texte = ABC ABCDAB ABCDABCDABDE

k=0, **ABC** ABCDAB ABCDABCDABDE

ABCDABD

On constate que la 4ème lettre diffère donc on passe directement au décalage k=4 (k=1 si naïf)

k=4, ABC **ABCDAB** ABCDABCDABDE

ABCDABD

On ne peut pas avoir de décalage k=6 ou 7 donc on passe directement a k=8 (seul le motif AB se répète)

Exemple pour l'algorithme KMP

Motif = ABCDABD pi :

0	0	0	0	1	2	0

Texte = ABC ABCDAB ABCDABCDABDE

k=0, **ABC** ABCDAB ABCDABCDABDE

ABCDABD

On constate que la 4ème lettre diffère donc on passe directement au décalage k=4 (k=1 si naïf)

k=4, ABC **ABCDAB** ABCDABCDABDE

ABCDABD

On ne peut pas avoir de décalage k=6 ou 7 donc on passe directement a k=8 (seul le motif AB se répète)

k=8, ABC ABCD**AB** ABCDABCDABDE

ABCDABD

Exemple pour l'algorithme KMP

Motif = ABCDABD pi :

0	0	0	0	1	2	0

Texte = ABC ABCDAB ABCDABCDABDE

k=0, **ABC** ABCDAB ABCDABCDABDE

ABCDABD

On constate que la 4ème lettre diffère donc on passe directement au décalage k=4 (k=1 si naïf)

k=4, ABC **ABCDAB** ABCDABCDABDE

ABCDABD

On ne peut pas avoir de décalage k=6 ou 7 donc on passe directement a k=8 (seul le motif AB se répète)

k=8, ABC ABCD**AB** ABCDABCDABDE

ABCDABD

k=11, ABC ABCDAB **ABCDAB**CDABDE

ABCDABD

Exemple pour l'algorithme KMP

Motif = ABCDABD pi :

0	0	0	0	1	2	0

Texte = ABC ABCDAB ABCDABCDABDE

k=0, **ABC** ABCDAB ABCDABCDABDE

ABCDABD

On constate que la 4ème lettre diffère donc on passe directement au décalage k=4 (k=1 si naïf)

k=4, ABC **ABCDAB** ABCDABCDABDE

ABCDABD

On ne peut pas avoir de décalage k=6 ou 7 donc on passe directement a k=8 (seul le motif AB se répète)

k=8, ABC ABCD**AB** ABCDABCDABDE

ABCDABD

k=11, ABC ABCDAB **ABCDAB**CDABDE

ABCDABD

k=15, ABC ABCDAB ABCD**AB**CDABDE

ABCDABD

Exemple pour l'algorithme KMP

Motif = ABCDABD pi :

0	0	0	0	1	2	0

Texte = ABC ABCDAB ABCDABCDABDE

k=0, **ABC** ABCDAB ABCDABCDABDE

ABCDABD

On constate que la 4ème lettre diffère donc on passe directement au décalage k=4 (k=1 si naïf)

k=4, ABC **ABCDAB** ABCDABCDABDE

ABCDABD

On ne peut pas avoir de décalage k=6 ou 7 donc on passe directement a k=8 (seul le motif AB se répète)

k=8, ABC ABCD**AB** ABCDABCDABDE

ABCDABD

k=11, ABC ABCDAB **ABCDAB**CDABDE

ABCDABD

k=15, ABC ABCDAB ABCD**AB**CDABDE

ABCDABD

On trouve une occurrence du motif à la position 15 en ayant effectué seulement 24 comparaisons