TD d'algorithmique numéro 12 - Recherche de motifs
Daniel Hirschkoff et Natacha Portier
semaine du 3 au 8 janvier 2000

Question 1 ⋆ Variantes de 3-SAT ⋆

Montrer la NP complétude des deux variantes de 3-SAT suivantes :

1. 3-SAT NAE (*not all equal*), où l'on impose que les trois variables de chaque clause ne soient pas toutes à la même valeur ;
2. 3-SAT OIT (*one in three*), où l'on impose qu'exactement une variable soit à VRAI dans chaque clause.

Question 2 ⋆ Rabin-Karp ⋆

On cherche toutes les occurrences d'une chaîne de caractères, appelée aussi motif, dans un texte. C'est une fonction qui existe par exemple dans un éditeur de texte.

L'idée de base de l'algorithme de Rabin-Karp peut se formuler en travaillant avec l'alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, et en pensant aux mots de l'alphabet comme à des nombres écrits en base dix.

On cherche un motif $P[1 \ldots m]$ (tableau de m chiffres), auquel correspond une valeur décimale p dans un texte $T[1 \ldots n]$ (tableau de n chiffres) ; on note t_s la valeur décimale correspondant à la sous-chaîne $T[s + 1 \ldots s + m]$ de longueur m . Lorsque $t_s = p$, on dira que s *est un décalage valide*.

Question 2.1

Comment calculer p et t_0 en $O(m)$?

Comment calculer les valeurs restantes $t_1, t_2, \ldots, t_{n-m}$ en $O(n - m)$?

À supposer que l'on fait abstraction de la possibilité pour p et les t_i d'être des nombres très grands, indiquer une manière de résoudre le problème de la recherche de motif, et en évaluer la complexité.

Question 2.2

Seulement il n'est pas raisonnable de faire abstraction de la taille des nombres manipulés. On se restreint à raisonner sur un espace de mots réduit en travaillant modulo un certain entier q. On ne compare les mots que s'ils sont égaux modulo q. En pratique, lorsque l'on veut travailler avec un alphabet d−aire $\Sigma = \{0, 1, \ldots, d - 1\}$, on choisit q de telle manière que dq tienne dans un mot d'ordinateur (ici d est égal à dix).

Écrire l'algorithme.

Quel est son temps d'exécution dans le pire des cas?

Donner une configuration pour laquelle les vérifications prennent effectivement un temps correspondant à l'estimation ci-dessus.

Dans la pratique on s'atends à ce que ce genre de cas soit peu fréquent.

Question 3.0 ⋆ Recherche de motif avec des automates finis ⋆

Un automate fini M est défini par la donnée d'un quintuplet $(Q, q_0, A, \Sigma, \delta)$:

- Q est un ensemble fini d'*états*, parmi lesquels on distingue $q_0 \in Q$, l'*état initial*, et $A \subseteq Q$, l'ensemble des états terminaux ;
- Σ est un *alphabet* fini ;
- $\delta : Q \times \Sigma \to Q$ est la *fonction de transition* de M.

On le représente graphiquement par son graphe de Moore : les sommets sont les états, les arêtes sont étiquetées par des lettres et représentent la fonction de transition.

Donner par exemple un automate sur l'alphabet $\{0; 1\}$ qui reconnaisse les mots se terminant par un nombre pair de 1.

On veut «programmer» un automate fini pour résoudre le problème de la recherche d'un motif $P[1 \dots m]$ dans un texte. Pour ce faire, on aura besoin de la *fonction suffixe* σ associée à P. Pour tout entier k compris entre 1 et m, le mot P_k est constitué des k premières lettres de P c'est-à-dire $P_k = P[1]...P[k]$. Par convention P_0 est le mot vide. On définit alors $\sigma(x)$ comme le plus grand k tel que P_k soit un suffixe de x :

$$\sigma(x) \; = \; \max\{k : P_k \sqsupset x\}$$

(la notation $P_k \sqsupset x$ signifie que P_k est un suffixe de x).

Question 3.1 ⋆ Fonction suffixe ⋆

1. Pourquoi la fonction σ est-elle bien définie?

   Quelle relation peut-on établir entre $\sigma(x)$ et $\sigma(y)$, pour $x \sqsupset y$?

2. Montrer que pour toute chaîne x et pour tout caractère a, on a $\sigma(xa) \leq \sigma(x) + 1$.

3. Montrer que si $q = \sigma(x)$, alors $\sigma(xa) = \sigma(P_q a)$.

On définit alors un automate de la manière suivante : l'ensemble des états Q est pris égal à $\{0, 1, \dots, m\}$, on part de l'état $q_0 = 0$ et il y a un seul état terminal m (correspond à la reconnaissance des m caractères du motif). La fonction de transition δ est donnée, pour tous état q et caractère a, par :

$$\delta(q, a) \; = \; \sigma(P_q a)$$

Question 3.2 ⋆ Exemple ⋆

Construire l'automate pour $m = 3$, $\Sigma = \{a, b\}$ et $P = aba$. Le faire tourner sur le mot $M = abbababaaaba$ et retenir les indices i pour lesquels la lecture de M_i fait arriver dans l'état 3.

Question 3.3 ⋆ Fonction d'état final ⋆

Si Σ est un alphabet, c'est-à-dire un ensemble fini de lettres, on note Σ^* l'ensemble des mots sur Σ, i.e. l'ensemble des suites finies de Σ. La suite de longueur nulle est le mot vide.

À tout automate fini est associée une fonction $\phi : \Sigma^* \to Q$, appelée *fonction d'état final* qui associe à une chaîne w l'état dans lequel M se trouve après lu avoir celle-ci.

Montrer que si ϕ est la fonction d'état final associée à l'automate défini ci-dessus, automate que l'on fait tourner sur un texte $T[1 \ldots n]$, alors, pour $i = 0, 1, \ldots, n$:

$$\phi(T_i) = \sigma(T_i)$$

Question 3.4 ⋆ Complexité ⋆

On peut simuler l'exécution de l'automate à l'aide de la procédure suivante :

Recherche_Automate_Fini$(T, \delta, m) :=$
  $n :=$ *longueur*$[T]$
  $q := 0$
  `pour` i `de` 1 `à` n
    `faire` $q := \delta(q, T[i])$
    `si` $q = m$
      `alors` imprimer *le motif apparaît avec le décalage* $(i - m)$

Pourquoi la procédure ci-dessus est correcte ?

Le calcul de la fonction de transition se fait à l'aide de la procédure suivante :

Calcul_Fonction_Transition$(P, \Sigma) :=$
  $m :=$ *longueur*$[P]$
  `pour` q `de` 0 `à` m
    `faire pour` chaque caractère $a \in \Sigma$
      `faire` $k := \min(m + 1,\ q + 2)$
        `répéter` $k := k - 1$ `jusqu'à` $P_k \sqsupset P_q a$
        $\delta(q, a) := k$
  Retourner δ

Quelle est la complexité de la recherche de motifs à l'aide d'automates finis si l'on calcule la fonction de transition comme ci-dessus ?