

Analyse d'Algorithmes (INF566)

Algorithmes Probabilistes : Examen

Responsable : Alexandre Zvonkine

12 janvier 2009, 9h00 – 10h30

Tous documents autorisés ; rédiger sur une copie séparée

Approximate set membership

Soit U un grand “univers” d’objets qui peuvent potentiellement être traités, et soit $X = \{x_1, x_2, \dots, x_m\} \subset U$ un ensemble de taille “raisonnable” que l’on envisage représenter. On va implémenter deux opérations :

- *insertion* : mettre un $x \in U$ donné dans l’ensemble X ;
- *vérification d’appartenance* : pour un $x \in U$ donné répondre si oui ou non x appartient à X .

Pour la question d’appartenance, la réponse NON doit être toujours correcte tandis que *pour la réponse OUI on admet une certaine probabilité d’erreur*, quand l’algorithme réponds OUI tandis qu’en réalité x n’appartient pas à X . (Exemple d’application : une liste des bar-codes utilisés pour marquer des produits. On veut créer un bar-code pour un nouveau produit, et on vérifie si ce code n’était pas déjà utilisé avant. Une interdiction d’utiliser un code qui en réalité est libre n’est pas très grave si elle n’est pas trop systématique.) Le fait d’avoir accepter une réponse erronée à la question d’appartenance permet parfois réaliser une grande économie de l’espace-mémoire.

Soit E un ensemble fini. Une fonction $h : U \rightarrow E$ est une *fonction de hachage* si elle satisfait les conditions (contradictaires) suivantes :

- elle est déterministe, en ce sens que si on rencontre le même élément $x \in U$ une deuxième fois, la valeur $h(x)$ sera la même ;
- elle se calcule facilement ; dans la suite, on supposera que la complexité de calcul de $h(x)$ est $O(1)$;
- le comportement de la fonction est “stochastique” et “irrégulier”, de telle sorte que l’on peut utiliser, pour analyser des algorithmes, la théorie des probabilités.

Une telle “analyse probabiliste” ne donne pas de théorèmes rigoureux mais plutôt des estimations euristiques. Quant aux méthodes de construction des fonctions de hachage, on ne les discute pas ici mais il est clair que leur nature est similaire à celle de construction de générateurs des nombres pseudo-aléatoires.

Exercice 1 Soit E l'ensemble des mots binaires de longueur k ; on appelle parfois le mot $y = h(x) \in E$ une *empreinte digitale* de x (tandis que x lui-même peut nécessiter beaucoup plus de k bits pour sa description complète). Au lieu de garder l'ensemble $X = \{x_1, x_2, \dots, x_m\}$ on garde $Y = \{y_1, y_2, \dots, y_m\}$ où $y_i = h(x_i)$. Donner quelques brèves indications sur une (ou des) méthode(s) de représentation de l'ensemble Y et sur une implémentation des opérations d'insertion et de vérification d'appartenance et leurs complexités.

Pour une vérification d'appartenance $x \in X$, on distingue deux cas de figure :

1. En vérité, $x \in X$; cette situation est *déterministe* car $y = h(x) \in Y$ et l'algorithme, après avoir trouvé $y \in Y$, va répondre OUI.
2. En vérité, $x \notin X$; cette situation est *probabiliste* : l'algorithme peut aussi répondre OUI *par hasard* (si le même mot $y = h(x)$ avait déjà été produit auparavant par un autre $x' \in X$).

Exercice 2 On suppose, dans cet exercice, que pour tout mot $u \in E$ la probabilité d'obtenir $h(x) = u$ soit $1/2^k$, et que divers événements que vous allez considérer soient indépendants.

1. Calculer la probabilité d'erreur dans la réponse OUI pour la vérification d'appartenance. (Cette probabilité dépend de k et de m .)
2. Montrer que si $k = 2 \log_2 m$ alors cette probabilité est bornée par $1/m$.
3. Montrer que si on veut que la probabilité d'erreur soit bornée par une constante p_0 , on doit prendre $k = \Omega(\log m)$.

Indication. Vous pouvez utiliser les estimations

$$\left(1 - \frac{1}{t}\right)^a \leq 1 - \frac{a}{t} \quad \text{et} \quad \left(1 - \frac{1}{t}\right)^a \approx e^{-a/t}.$$

Conclusion. Il faut prendre $k = \Theta(\log m)$. Notons que k dépend du nombre m d'éléments dans l'ensemble X mais ne dépend pas de la taille d'un élément $x \in U$ lui-même qui peut *a priori* être quelconque.

Exercice 3 Soit maintenant $E = \{0, 1, \dots, n-1\}$, et on utilise r fonctions de hachage h_1, \dots, h_r *indépendantes* (mais fixées une fois choisies). Soit T un tableau de bits de longueur n initialisés à 0. On représente un élément $x \in U$ en mettant à 1 les bits numéro $h_1(x), \dots, h_r(x)$ dans le tableau T .

Remarques. (1) Deux positions $h_i(x)$ et $h_j(x)$ peuvent “par hasard” coïncider ; dans ce cas l’algorithme met à 1 moins de r bits. (2) Si un bit était déjà égal à 1 à cause des opérations précédentes alors sa valeur ne change pas.

Quelle est la complexité des opérations d’insertion et de vérification d’appartenance ?

Exercice 4 On suppose que, pour un $i \in \{1, \dots, r\}$ fixé, et pour une position $j \in \{0, 1, \dots, n-1\}$ fixée dans le tableau T , la probabilité que $h_i(x) = j$ (et que l’on doit donc, en traitant x , mettre à 1 le bit $T[j]$) est égale à $1/n$.

1. Calculer la probabilité q qu’un bit donné du tableau T reste égal à 0 après l’insertion de tous les éléments de $X = \{x_1, \dots, x_m\}$.

Cette probabilité dépend des paramètres r , n et m ; exprimer la en fonction des paramètres r et $k = n/m$. (Le paramètre k n’est rien d’autre que le nombre moyen de bits utilisés pour stocker un élément de X .)

2. Calculer (ou plutôt donner une bonne approximation de) la probabilité p d’erreur de l’opération de vérification d’appartenance quand cette opération donne la réponse OUI.
3. Montrer que, pour le paramètre k fixé, la probabilité d’erreur p atteint le minimum p_{\min} quand $q \approx 1/2$ et $r \approx k \ln 2$. Calculer p_{\min} .

Indications. (1) Vous pouvez considérer tous les paramètres comme étant continus et utiliser la dérivation. (2) Une égalité approchée de l’indication à l’exercice 2 peut aussi être utile. (3) Le paramètre $t = r/k$ peut être utile pour simplifier le calcul. (4) Pour dériver une fonction $F(t) = f(t)^{g(t)}$ on utilise l’astuce suivante :

$$\ln F(t) = g(t) \cdot \ln f(t) \quad \Rightarrow \quad F(t) = \exp(g(t) \cdot \ln f(t)) .$$

Remarque. Quand $q = 1/2$ le tableau T ressemble à une suite des bits aléatoires.

Conclusion. Par rapport à la méthode des exercices 1 et 2 nous avons amélioré en même temps le nombre moyen de bits par un élément stocké et la complexité des opérations d’insertion et de vérification d’appartenance.

Exercice 5 Préciser la conclusion ci-dessus dans la situation quand l’objectif est de borner la probabilité d’erreur p par une constante p_0 .

FIN DE L’ÉNONCÉ