Algorithmique Avancée

TD 8-10 : Techniques de hachage

Exemplaire enseignant

1 Hachage dynamique

Exercice* 1.1 *: Quelques exemples
Dans cet exercice, les éléments sont les lettres de l'alphabet. Leurs valeurs de hachage, ou clés sont indiquées dans ce tableau.

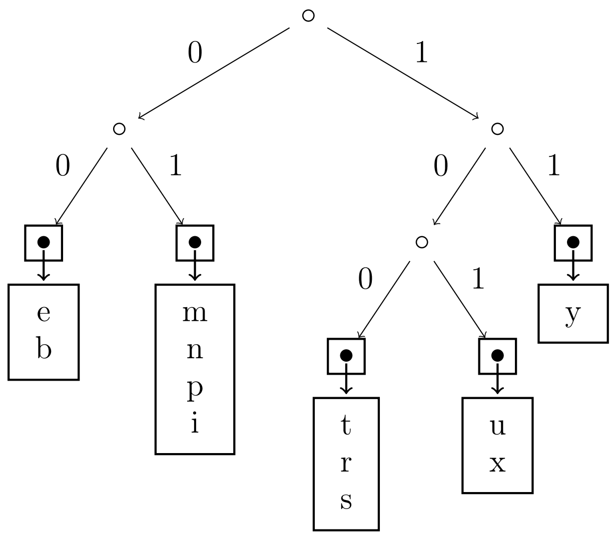
Lettre	Clé	Lettre	Clé	Lettre	Clé	Lettre	Clé
a	00000	i	01000	q	10000	y	11000
b	00001	j	01001	r	10001	z	11001
c	00010	k	01010	s	10010		
d	00011	l	01011	t	10011		
e	00100	m	01100	u	10100		
f	00101	n	01101	v	10101		
g	00110	o	01110	w	10110		
h	00111	p	01111	x	10111		

Question 1.1.1 Réaliser le hachage dynamique des clés $t, m, y, u, n, r, p, x, e, s, i, b$, dans cet ordre, avec pages de taille 4.
Que se passe-t-il si on modifie l'ordre d'insertion des clés ?

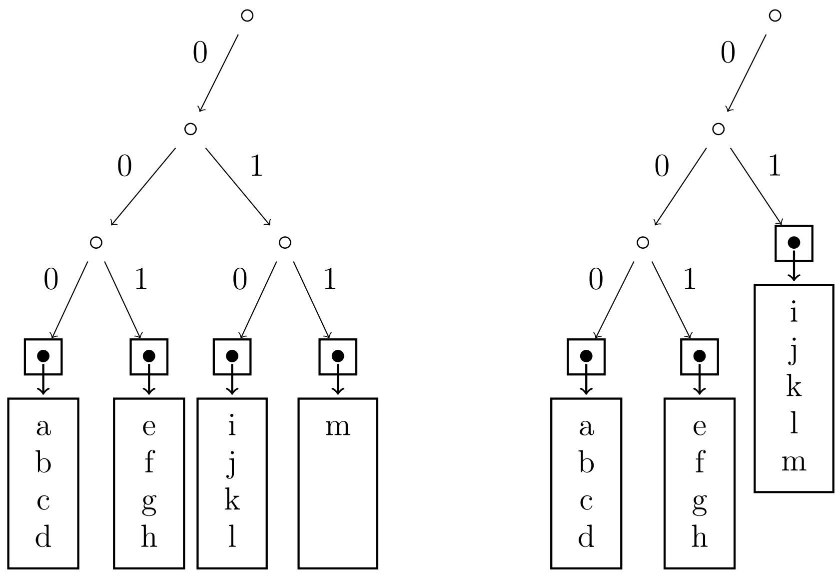
Question 1.1.2 Réaliser le hachage dynamique des clés $a, b, c, d, e, f, g, h, i, j, k, l, m$, dans cet ordre, avec pages de taille 4, puis avec pages de taille 7.

Solution

1. Rappel du principe : au début il n'y a qu'une seule page. Lorsqu'une page est remplie d'éléments, et qu'on souhaite insérer un nouvel élément, on dédouble la page.



2. Les résultats ne diffèrent que par l'introduction de m (sans cette clé ils sont identiques). J'ai l'impression qu'entre 2^k et $2^{k+1} - 1$ la différence est marginale.



Exercice 1.2 ***: Recherche et insertion***

Pour travailler sur le hachage dynamique, on utilisera les primitives suivantes (vous en compléterez la spécification) :

`IndexArbre`	*index* $\times$ *index* $\rightarrow$ *index*
`IndexFeuille`	*page* $\rightarrow$ *index*
`PageVide`	$\rightarrow$ *page*
`IndexGauche`	*index* $\rightarrow$ *index*
`IndexDroit`	*index* $\rightarrow$ *index*
`PageDeFeuille`	*index* $\rightarrow$ *page*
`Élément`	*page* $\times$ *entier* $\rightarrow$ *élément*
`EstFeuille`	*index* $\rightarrow$ *booléen*
`EstPagePleine`	*page* $\rightarrow$ *booléen*
`EstDansPage`	*élément* $\times$ *page* $\rightarrow$ *booléen*
`InsertionDansPage`	*page* $\times$ *élément* $\rightarrow$ *page*

On suppose que l'on dispose aussi d'une fonction :

`BitHachage`	*élément* $\times$ *entier* $\rightarrow$ *bit*
`BitHachage(x,k)`	*renvoie le k-ième bit de la valeur de hachage de x*

Question 1.2.1 (Recherche) Écrire un algorithme de recherche dans un index.

Question 1.2.2 (Insertion) Écrire un algorithme d'insertion dans un index.

Solution

1.
```
RechercheIndexN(table, elt, nb)
  si EstFeuille(table)
  alors retourne (EstDansPage(PageDeFeuille(table)))
  sinon si BitHachage(elt,nb)=0
        alors retourne RechercheIndexN(IndexGauche(table),elt,nb+1)
        sinon retourne RechercheIndexN(IndexDroit(table),elt,nb+1)
        fin si
  fin si
fin RechercheIndexN

RechercheIndex(table, elt)
  RechercheIndexN (table, elt, 1)
fin RechercheIndex
```

2. Version fonctionnelle.

```
InsertionIndexN (table ,elt ,nb)
  si EstFeuille(table)
     alors p <- PageDeFeuille (table)
     si non EstPagePleine (p)
            alors
                retourne IndexFeuille(InsertionDansPage (p, elt))
            sinon
                /* on separe la page en deux, on les remplit avec les elements de p */
                pg = PageVide()
                pd = PageVide()
                pour tout e dans p faire
                    si BitHachage (e,nb)=0
                         alors  pg = InsertionDansPage(pg,e)
                         sinon pd = InsertionDansPage(pd,e)
                     fin si
                fin pour
                retourne InsertionIndexN(IndexArbre(IndexFeuille(pg),IndexFeuille(pd)),
                                                                 elt,nb)
     fin si
  sinon si BitHachage (elt ,nb )=0
     alors retourne IndexArbre(InsertionIndexN(IndexGauche(table),elt ,nb +1),
                                        IndexDroit(table))
     sinon retourne IndexArbre(IndexGauche(table),
                               InsertionIndexN(IndexDroit(table),elt ,nb +1))
     fin si
  fin si
fin InsertionIndexN

InsertionIndex (table , elt)
      retourner InsertionIndexN (table , elt , 1)
fin InsertionIndex

La boucle 'pour tout e dans p' peut \^etre r\'ealis\'ee par la fonction Element
\`a condition de disposer aussi d'une fonction qui donne la taille des pages de la table
```

2 Familles de fonctions de hachage

Exercice** 2.1 : **Application des familles universelles

Cet exercice étudie une stratégie de hachage qui, étant donné un ensemble *statique* de n clés, effectue une recherche avec une *complexité au pire* en $O(1)$ comparaisons entre clés, en utilisant une mémoire totale en $O(n)$ (la mémoire est comptée en nombre de cases pouvant contenir une clé).

On suppose que l'on dispose d'un ensemble universel de fonctions de hachage $\mathcal{H}$.

Question 2.1.1 Montrer que si l'on hache un ensemble statique de n clés dans une table de hachage de taille $m = n^2$, à l'aide d'une fonction de hachage h choisie aléatoirement dans un ensemble universel de fonctions de hachage, alors la probabilité qu'il n'y ait aucune collision est supérieure à 1/2.
On pourra utiliser l'inégalité de Markov : $\Pr(X \geq a) \leq \mathbb{E}[X]/a$.

Question 2.1.2 On suppose que l'on peut réserver une table de hachage dont la taille est le carré du nombre d'éléments à stocker. Un algorithme permettant de hacher les éléments sans aucune collision (qui a donc une complexité en $O(1)$ dans le pire des cas) est obtenu en essayant plusieurs fonctions de hachage de $\mathcal{H}$. Montrer que le nombre moyen d'essais de fonctions est inférieur à 2.

Question 2.1.3 Lorsque l'on ne peut pas réserver une table de hachage dont la taille est le carré du nombre d'éléments à stocker, on procède en 2 niveaux.

— Au premier niveau, en utilisant une fonction aléatoire h de $\mathcal{H}$, on répartit les clés en m sous-ensembles S_j, formés de n_j clés ayant même valeur de hachage j, pour $j = 0..m-1$.
(Le nombre total de clés est $n = \sum_{j=0}^{m-1} n_j$).

— Au second niveau, on crée pour chaque S_j, une taille de hachage de taille n_j^2, via une fonction de hachage $h^{(j)}$, choisie aléatoirement dans $\mathcal{H}$.

1. Décrire l'algorithme précédent en pseudo-code.
2. Montrer que la taille de la mémoire requise par cet algorithme est en $O(n)$.
(On pourra montrer – ou admettre – que la valeur moyenne de $\sum_{j=0}^{m-1} n_j^2$ est inférieure à $2n$).

Solution

Note : l'inégalité de Markov donnée par $\Pr(X \geq a) \leq \mathbb{E}[X]/a$ est prouvée par les relations ci-dessous :

$$\begin{aligned}
E(X) &= \sum_{\omega} Pr(\omega)X(\omega) \\
&= \sum_{\omega|X(\omega)\geq a} Pr(\omega)X(\omega) + \sum_{\omega|X(\omega)<a} Pr(\omega)X(\omega) \\
&\geq \sum_{\omega|X(\omega)\geq a} Pr(\omega)X(\omega) \\
&\geq \sum_{\omega|X(\omega)\geq a} Pr(\omega)a \\
&\geq a \sum_{\omega|X(\omega)\geq a} Pr(\omega) \\
&\geq a\; Pr(X(\omega) \geq a)
\end{aligned}$$

1. Soit X la variable aléatoire égale au nombre de collisions de la fonction h tirée au hasard. On a $\forall(x,y), \Pr(h(x) = h(y)) \leq \frac{1}{m} \sim \frac{1}{n^2}$. Le nombre de couples (x,y) est $\binom{n}{2}$. Donc le nombre moyen de collisions est

$$E(X) = \binom{n}{2}\frac{1}{n^2} \leq \frac{1}{2}\,.$$

En appliquant l'inégalité de Markov avec $a = 1$, on obtient $\Pr(X \geq 1) \leq E(X) \leq 1/2$. En passant à l'événement contraire, la probabilité qu'il n'y ait aucune collision est $\Pr(X < 1) = 1 - \Pr(X \geq 1) \geq 1/2$.

2. On choisit une fonction dans l'ensemble universel de fonctions de hachage ; s'il y a des collisions on recommence. Le nombre moyen d'essais de fonction de hachage est inférieur à 2 : la probabilité qu'on reussise à la première fonction choisie est $c \leq \frac{1}{2}$; la probabilité qu'on échoue à la première fonction choisie et qu'on reussise à la deuxième est $(1-c)c$... la probabilité qu'on échoue aux $k-1$ premières fonctions choisies et qu'on reussise à la k-ième est $(1-c)^{k-1}c$. Donc le nombre moyen d'essai de fonctions est $c\sum_k k(1-c)^{k-1} = c/c^2 = 1/c \leq 2$.
3. Algorithme :
— choisir m premier, $m \sim n$ et $h_1 \in \mathcal{H}$ universel.
— pour $i = 1..m$, soit n_i le nombre de clés tq $h_1(x) = i$
— pour chaque i, choisir m_i premier, $m_i \sim n_i^2$ et $h_{2,i} \in \mathcal{H}$ universel.
Mémoire totale $O(n)$ en moyenne car $\sum \mathbb{E}(n_i^2) = O(n)$. En effet $\forall i, n_i^2 = n_i + 2\binom{n_i}{2}$, donc $\mathbb{E}[\sum n_i^2] = n + 2\mathbb{E}[\sum \binom{n_i}{2}]$. Or $\sum \binom{n_i}{2}$ = nb total collisions. Et comme $\mathcal{H}$ universel, $\mathbb{E}$[nb total collisions] $= \frac{1}{m}\binom{n}{2}$ = (n-1)/2.

Exercice* 2.2 : *Hachage k-universel

Soit $\mathcal{H}$ une famille de fonctions de hachage dans laquelle chaque fonction $h \in \mathcal{H}$ envoie l'univers de clés U dans l'intervalle d'entiers $[0, 1, \ldots, m-1]$. On dit que $\mathcal{H}$ est k-**universelle** ssi, pour toutes clés $x_1, \ldots, x_k$ deux à deux distinctes et pour toutes valeurs $v_1, \ldots, v_k$ dans $[0, 1, \ldots, m-1]$:

$$|\{h \in \mathcal{H}; h(x_1) = v_1, \ldots, h(x_k) = v_k\}| = \frac{|\mathcal{H}|}{m^k}.$$

Autrement dit, en munissant $\mathcal{H}$ de la probabilité uniforme :

$$\Pr(\{h \in \mathcal{H}; h(x_1) = v_1, \ldots, h(x_k) = v_k\}) = \frac{1}{m^k}$$

ou encore, pour h choisie au hasard dans $\mathcal{H}$:

$$\Pr(h(x_1) = v_1, \ldots, h(x_k) = v_k) = \frac{1}{m^k}.$$

Question 2.2.1 Soit U un univers ayant n clés et $\mathcal{H}$ la famille de toutes les fonctions de U dans $[0, 1, \ldots, m-1]$. Montrer que $\mathcal{H}$ est k-universelle (pour $k \leq n$).

Question 2.2.2 Écrire la définition de famille 2-universelle.

Question 2.2.3 Montrer que si une famille $\mathcal{H}$ de fonctions de hachage est 2-universelle alors elle vérifie, pour toute clé x et pour toute valeur v dans $[0, 1, \ldots, m-1]$: $\Pr(h(x) = v) = \frac{1}{m}$.

Question 2.2.4 Montrer que si une famille $\mathcal{H}$ de fonctions de hachage est 2-universelle alors elle est universelle.

Question 2.2.5 Soit $U = \{x_1, x_2, x_3, x_4\}$, on considère les familles $\mathcal{H}_0$ et $\mathcal{H}_1$ de fonctions de U dans $\{0, 1\}$ données par les tableaux suivants :

$\mathcal{H}_0$	h_1	h_2	h_3	h_4
x_1	0	0	0	0
x_2	0	1	0	1
x_3	0	0	1	1
x_4	0	1	1	0

$\mathcal{H}_1$	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
x_1	0	1	0	1	0	1	0	1
x_2	0	1	1	0	0	1	1	0
x_3	0	1	0	1	1	0	1	0
x_4	0	1	1	0	1	0	0	1

La famille $\mathcal{H}_0$ est-elle universelle ? 1-universelle ? 2-universelle ? et la famille $\mathcal{H}_1$?

Question 2.2.6 Une famille universelle est-elle nécessairement 2-universelle ?

Question 2.2.7 Montrer que si une famille $\mathcal{H}$ de fonctions de hachage est $(k+1)$-universelle alors elle est k-universelle.

Question 2.2.8 En déduire que si une famille $\mathcal{H}$ de fonctions de hachage est k-universelle, avec $k \geq 2$, alors elle est universelle et elle vérifie, pour toute clé x et pour toute valeur v dans $[0, 1, \ldots, m-1]$: $\Pr(h(x) = v) = \frac{1}{m}$.

Solution

1. Soit $x_1, \ldots, x_k$ des clés deux à deux distinctes et $v_1, \ldots, v_k$ des valeurs dans $[0, 1, \ldots, m-1]$. Soit $x_{k+1}, \ldots, x_n$ les autres clés de U. Le nombre de fonctions h de $\mathcal{H}$ telles que $h(x_1) = v_1, \ldots, h(x_k) = v_k$ est égal au nombre de choix possibles des images de $x_{k+1}, \ldots, x_n$ parmi les m éléments de $[0, 1, \ldots, m-1]$, c'est-à-dire à m^{n-k}. Comme $|\mathcal{H} |= m^n$, on a bien :

$$|\{h \in \mathcal{H}; h(x_1) = v_1, \ldots, h(x_k) = v_k\}|= \frac{|\mathcal{H}|}{m^k}.$$

2. Une famille $\mathcal{H}$ de fonctions de hachage est 2-universelle ssi, pour toutes clés distinctes x et y et pour toutes valeurs v et w dans $[0, 1, \ldots, m-1]$:

$$|\{h \in \mathcal{H}; h(x) = v, h(y) = w\}|= \frac{|\mathcal{H}|}{m^2}.$$

Autrement dit, pour h choisie au hasard dans $\mathcal{H}$:

$$\Pr(h(x) = v, h(y) = w) = \frac{1}{m^2}.$$

Remarque :
On pourrait aussi demander ce qu'est une famille 1-universelle. C'est une famille telle que pour toute clé x et pour toute valeur v : $|\{h \in \mathcal{H}; h(x) = v\}|= \frac{|\mathcal{H}|}{m}$. Autrement dit, pour h choisie au hasard dans $\mathcal{H}$:
$\Pr(h(x) = v) = \frac{1}{m}$.
Attention à la notation $\Pr(h(x) = v)$ qui, habituellement, est un raccourci de
$\Pr(\{x \in U; h(x) = v\})$.

3. Soit y une clé distincte de x alors
$\{h \in \mathcal{H}; h(x) = v\} = \bigcup_{w=0}^{m-1} \{h \in \mathcal{H}; h(x) = v, h(y) = w\}$.
Cette union est disjointe donc
$\Pr(h(x) = v) = \sum_{w=0}^{m-1} \Pr(h(x) = v, h(y) = w) = m * \frac{1}{m^2} = \frac{1}{m}$.

4. On rappelle qu'une famille $\mathcal{H}$ de fonctions de hachage est universelle ssi, pour toutes clés x et y distinctes :

$$|\{h \in \mathcal{H}; h(x) = h(y)\}|= \frac{|\mathcal{H}|}{m}$$

ou encore, pour h choisie au hasard dans $\mathcal{H}$:

$$\Pr(h(x) = h(y)) = \frac{1}{m}$$

Soit $\mathcal{H}$ une famille 2-universelle. Soit x et y deux clés distinctes alors
$\{h \in \mathcal{H}; h(x) = h(y)\} = \bigcup_{v=0}^{m-1} \{h \in \mathcal{H}; h(x) = v, h(y) = v\}$.

Cette union est disjointe donc
$\Pr(h(x) = h(y)) = \sum_{v=0}^{m-1} \Pr(h(x) = v, h(y) = v) = m * \dfrac{1}{m^2} = \dfrac{1}{m}$.
La famille $\mathcal{H}$ est donc universelle.

5. Ici $m = 2$ donc :
- une famille $\mathcal{H}$ est 2-universelle ssi $\Pr(\{h \in \mathcal{H}; h(x) = v, h(y) = w\}) = \frac{1}{4}$,
- une famille $\mathcal{H}$ est universelle ssi $\Pr(\{h \in \mathcal{H}; h(x) = h(y)\}) = \frac{1}{2}$.

La famille $\mathcal{H}_0$ n'est pas 1-universelle car $\Pr(\{h \in \mathcal{H}; h(x_1) = 0\}) = 1$.
La famille $\mathcal{H}_0$ n'est donc pas 2-universelle. On peut vérifier directement que $\Pr(\{h \in \mathcal{H}; h(x_1) = 0, h(x_2) = 0\}) = \Pr(\{h_1, h_3\}) = \frac{1}{2}$.
La famille $\mathcal{H}_0$ est universelle car :
$\Pr(\{h \in \mathcal{H}; h(x_1) = h(x_2)\}) = \Pr(\{h_1, h_3\}) = \frac{1}{2}$
$\Pr(\{h \in \mathcal{H}; h(x_1) = h(x_3)\}) = \Pr(\{h_1, h_2\}) = \frac{1}{2}$
$\Pr(\{h \in \mathcal{H}; h(x_1) = h(x_4)\}) = \Pr(\{h_1, h_4\}) = \frac{1}{2}$
$\Pr(\{h \in \mathcal{H}; h(x_2) = h(x_3)\}) = \Pr(\{h_1, h_4\}) = \frac{1}{2}$
$\Pr(\{h \in \mathcal{H}; h(x_2) = h(x_4)\}) = \Pr(\{h_1, h_2\}) = \frac{1}{2}$
$\Pr(\{h \in \mathcal{H}; h(x_3) = h(x_4)\}) = \Pr(\{h_1, h_3\}) = \frac{1}{2}$

La famille $\mathcal{H}_1$ est 2-universelle car :
$\Pr(\{h \in \mathcal{H}; h(x_1) = 0, h(x_2) = 0\}) = \Pr(\{h_1, h_5\}) = \frac{1}{4}$
$\Pr(\{h \in \mathcal{H}; h(x_1) = 0, h(x_2) = 1\}) = \Pr(\{h_3, h_7\}) = \frac{1}{4}$
$\Pr(\{h \in \mathcal{H}; h(x_1) = 1, h(x_2) = 0\}) = \Pr(\{h_4, h_8\}) = \frac{1}{4}$
$\Pr(\{h \in \mathcal{H}; h(x_1) = 1, h(x_2) = 1\}) = \Pr(\{h_3, h_7\}) = \frac{1}{4}$
$\Pr(\{h \in \mathcal{H}; h(x_1) = 0, h(x_3) = 0\}) = \Pr(\{h_1, h_3\}) = \frac{1}{4}$
$\Pr(\{h \in \mathcal{H}; h(x_1) = 0, h(x_3) = 1\}) = \Pr(\{h_2, h_6\}) = \frac{1}{4}$
etc... (il y a 24 cas en tout) Elle est donc universelle et 1-universelle.

6. Non, *cf.* question précédente.

7. Supposons que $\mathcal{H}$ soit $(k+1)$-universelle. Soit $x_1, \ldots, x_k$ k clés distinctes de U, soit $v_1, \ldots, v_k$ k valeurs de $[0, \ldots, m-1]$ et soit x_{k+1} une clé de U, distincte de $x_1, \ldots, x_k$.
$\Pr(h(x_1) = v_1, \ldots, h(x_k) = v_k) = \sum_{v=0}^{m-1} \Pr(h(x_1) = v_1, \ldots, h(x_k) = v_k, x_{k+1} = v)$
$\Pr(h(x_1) = m \ \dfrac{1}{m^{k+1}} = \dfrac{1}{m^k}$

8. Si une famille $\mathcal{H}$ est k-universelle ($k \geq 2$) alors elle est 2-universelle (puisque $(k+1)$-universelle $\Rightarrow$ k-universelle). Par conséquent elle est universelle. De plus, elle est 1-universelle, cad qu'elle vérifie $\Pr(h(x) = v) = \frac{1}{m}$.
Encore une fois, attention à la notation $\Pr(h(x) = v)$.

3 Filtres de Bloom

Exercice* 3.3 : *Appartenance à un ensemble

Étant donné un ensemble A de n éléments, l'objectif est de stocker les éléments de A avec peu de mémoire, pour ensuite tester l'appartenance à A de façon probabiliste.

La structure est un tableau T de m bits, et on utilise k fonctions de hachage uniformes et indépendantes $h_1, \ldots, h_k$ à images dans $\{0, \cdots, m-1\}$. Pour "placer" un élément x de A dans cette structure, on met la valeur 1 dans chaque $T[h_i(x)]$, pour $i = 1, \ldots, k$.

Fonction `Consruction` (ensemble A, entier m, fonctions de hachage $h_1, \ldots, h_k$)
T = table de m bits, initialisée à 0
PourChaque a_i dans A
  PourChaque fonction de hachage h_j
    $T[h_j(a_i)] = 1$
  FinPour
FinPour
Retourne T

Ensuite, pour rechercher si un élément y appartient à l'ensemble, on calcule tous les $h_i(y)$ et on vérifie que tous les bits correspondant dans T valent 1.

Fonction `Appartient?` (élément y, table T , fonctions de hachage $h_1, \ldots, h_k$)
PourChaque fonction de hachage h_j
  Si $T[h_j(y)] = 0$ **Alors Retourne** NON
FinPour
Retourne OUI

Lorsque la fonction `Appartient?` retourne NON, c'est que l'élément y n'est pas dans A mais il est possible qu'elle retourne OUI pour un élément qui n'appartient pas à l'ensemble (on parle alors de *faux positif*).

Question 3.3.1 Construire la table T pour $A = \{9, 11\}$, avec $m = 5$, $k = 2$, et les fonctions de hachage $h_1(x) = x \mod 5$ et $h_2(x) = 2 \times x + 3 \mod 5$.
Appliquer ensuite la fonction `Appartient?` pour $y = 15$ et $y = 16$. Expliquer le résultat.

Question 3.3.2 Étant données les tables T_A et T_B de taille m, associées à 2 ensembles A et B, en utilisant les mêmes fonctions de hachage $h_1, ..., h_k$. Expliquer comment construire la table associée à l'union $A \cup B$.

Question 3.3.3 Dans la suite on considère une table T_A associée à un ensemble A.
Montrer que la probabilité pour qu'un bit donné de la table soit égal à 0 est $p_0 = (1 - 1/m)^{kn}$.
En déduire la probabilité p_r que la fonction `Appartient?` donne un faux positif est majorée par $(1 - \exp(-kn/m))^k$, pour m grand (on utilisera l'approximation $(1 - 1/m)^{kn} \sim \exp(-kn/m)$).

Question 3.3.4 Étant donné un ensemble de n éléments, on peut jouer sur la taille de la table et le nombre de fonctions de hachage pour minimiser la probabilité de faux positifs.
Pour m et n fixés, montrer que la valeur de k qui minimise la probabilité p_r est $\frac{m}{n} \log 2$ (pour m grand).
Discuter du choix des valeurs de k et m.

Solution

1. Pour $A = \{9, 11\}$, $m = 5$, $k = 2$, $h_1(x) = x \mod 5$ et $h_2(x) = 2 \times x + 3 \mod 5$, on obtient $h_1(9) = 4$, $h_2(9) = 1$, $h_1(11) = 1$, $h_2(11) = 0$, d'où la table $T = [1, 1, 0, 0, 1]$.
`Appartient?`(15, T, h_1, h_2) : $h_1(15) = 0$, $h_2(15) = 3$ et $T[3] = 0$ donc NON.

`Appartient?`$(16,\ T,\ h_1, h_2)$: $h_1(16) = 1$, $h_2(16) = 0$ et $T[1] = 1$, $T[0] = 1$ donc OUI, c'est un faux positif.

2. **Fonction `Union`** (table T_A, table T_B)
T = table de m bits, initialisée à 0
Pour i variant de 0 à $m-1$:
  Si $T_A[i] = 1$ **ou** $T_B[i] = 1$ **Alors** :
    $T[i] = 1$
  FinSi
FinPour
Retourne T

On prouve que $x \in A \cup B \Rightarrow T[h_j(x)] = 1$ pour tout $j = 1, \ldots, k$ (c'est très simple).
Mais je pense que ça ne suffit pas à montrer que T est la table que l'on obtient avec l'algo `Consruction` appliqué à $A \cup B$.
On doit pouvoir énoncer la propriété suivante.
Propriété : T est la table obtenue avec l'algo `Construction` appliqué à E ssi :
– $x \in E \Rightarrow T[h_j(x)] = 1$ pour tout $j = 1, \ldots, k$
– $T[i] = 1 \Rightarrow \exists x \in E, \exists j \in \{1, \ldots, k\}$ tels que $h_j(x) = i$.
On montre facilement que la table T construite par l'algo `Union` vérifie cette propriété.
Question : que se passe-t-il pour l'intersection de A et B ?
Si T_A est la table de A, si T_B est la table de T_B et si T est la table définie par $T[j] = 1$ ssi $T_A[j] = 1$ et $T_B[j] = 1$ alors on montre facilement que $x \in A \cap B \Rightarrow T[h_j(x)] = 1$ pour tout $j = 1, \ldots, k$. Mais T n'est pas nécessairement la table que l'on obtient avec l'algo `Construction` appliqué à $A \cap B$. Par exemple, si on reprend les données de la question précédente et si on prend $B = \{16\}$ $T = [1, 1, 0, 0]$ qui n'est pas la table que l'on obtient avec l'algo `Construction` appliqué à $A \cap B = \emptyset$.

3. Lorsqu'on insère un élément dans le tableau :
– la proba qu'un certain bit du tableau ne soit pas mis à 1 par l'une des fonctions de hachage est $1 - \frac{1}{m}$
– les fonctions de hachage étant indépendantes, la proba qu'un certain bit du tableau ne soit mis à 1 par aucune des k fonctions de hachage est $(1 - \frac{1}{m})^k$.
Lorsqu'on insère un élément, la proba qu'un bit du tableau reste à 0 est donc $(1 - \frac{1}{m})^k$. Après insertion des n éléments de A, la proba qu'un bit du tableau reste à 0 est donc $(1 - \frac{1}{m})^{kn}$.
La proba p_r que la fonction `Appartient?` donne un faux positif est inférieure ou égale à la proba que la fonction `Appartient?` retourne OUI.
La proba qu'un bit soit égal à 0 est $(1 - \frac{1}{m})^{kn}$. La proba qu'un bit soit égal à 1 est donc $1 - (1 - \frac{1}{m})^{kn}$ et la proba que k bits soient égaux à 1 est $(1 - (1 - \frac{1}{m})^{kn})^k$.
En utilisant l'approximation $(1 - \frac{1}{m})^{kn} \sim e^{-\frac{kn}{m}}$, on a $(1 - (1 - \frac{1}{m})^{kn})^k \sim (1 - e^{-\frac{kn}{m}})^k$.
Donc p_r est majorée par (une approximation de ?) $(1 - e^{-\frac{kn}{m}})^k$.

4. On peut remarquer que p_r croît quand n croît et décroît quand m croît.
Étudions les variations de p_r en fonction de k pour m et n fixés. Posons $\lambda = \frac{n}{m}$ et étudions la fonction

$$f(x) = (1 - e^{-\lambda x})^x = e^{x \log(1 - e^{-\lambda x})} \qquad \text{pour } x > 0.$$

Il faut calculer un minimum pour $f(x)$, donc étudier le signe de

$$f'(x) = (\log(1 - e^{-\lambda x}) + \lambda x \frac{e^{-\lambda x}}{1 - e^{-\lambda x}}) e^{x \log(1 - e^{-\lambda x})} \qquad \text{pour } x > 0.$$

Comme $e^{x \log(1 - e^{-\lambda x})} > 0$, $f'(x)$ a même signe que :

$$\log(1 - e^{-\lambda x}) + \lambda x \frac{e^{-\lambda x}}{1 - e^{-\lambda x}} \qquad \text{pour } x > 0.$$

En posant $X = 1 - e^{-\lambda x}$, on est ramené à l'étude du signe de :

$$g(X) = \frac{X \log(X) - (1 - X) \log(1 - X)}{X} \qquad \text{pour } 0 < X < 1.$$

Comme $0 < X < 1$, $g(X)$ a même signe que :

$$h(X) = X \log(X) - (1 - X) \log(1 - X).$$

— $h(X)$ s'annule en $X = 1/2$ (racine évidente),
— pour $0 < X < 1/2$, on a $X < 1 - X$ et $\log(X) < \log(1 - X)$ donc $h(X) < 0$,
— comme $h(X) = -h(1 - X)$ on a $h(X) > 0$ pour $1/2 < X < 1$.

Par conséquent :
— $f'(x) = 0$ pour $1 - e^{-\lambda x_0} = 1/2$, c'est-à-dire pour $x_0 = \frac{1}{\lambda} \log(2) = \frac{m}{n} \log(2)$.
— $f'(x) < 0$ pour $0 < 1 - e^{-\lambda x} < 1/2$, c'est-à-dire pour $0 < x < x_0$,
— $f'(x) > 0$ pour $1/2 < 1 - e^{-\lambda x} < 1$, c'est-à-dire pour $x > x_0$.

Donc $f(x)$ admet un minimum en $x_0 = \frac{m}{n} \log(2)$.
La proba minimum de faux positifs est donc $f(x_0)$ qui vaut $((\frac{1}{2})^{\log(2)})^{\frac{m}{n}}$ (environ $0.6185^{\frac{m}{n}}$).
Si on veut avoir une proba de faux positifs majorée par p_0 fixé pour A de taille n, comment choisir m et k ?
On choisit le k qui minimise la proba de faux positifs : $k = \frac{m}{n} \log(2)$. La proba de faux positifs est alors majorée par $((\frac{1}{2})^{\log(2)})^{\frac{m}{n}}$. On veut que $((\frac{1}{2})^{\log(2)})^{\frac{m}{n}} = p_0$, d'où :
$m = -\frac{n \log(p_0)}{(\log(2))^2}$.
On prend ce m et $k = \frac{m}{n} \log(2) = -\frac{\log(p_0)}{\log(2)}$.