Fiche TD10 : Tables et fonctions de hachage

Exercice 1 *Une table sans collision*

On considère les couples (clé, valeur) suivants : (7, A), (8, ?), (16, b), (24, B), (19, !). On souhaite les insérer dans une table de hachage à 7 alvéoles avec la fonction de hachage $h : k \mapsto k \bmod 7$. Décrire le contenu de la table de hachage résultant de l'insertion des valeurs ci-dessus.

Exercice 2 *Gestion de collisions*

On considère un ensemble de clés que l'on souhaite directement stocker dans une table de hachage de taille $m = 11$ (remarquons que dans ce cas, la valeur associée à une clé est simplement égale à la clé) grâce à la fonction de hachage $h : x \mapsto x \bmod m$.

1. Décrire et dessiner les structures de données utilisées lors de la séquence d'ajouts $10, 22, 31, 4, 15, 28, 83, 88, 59, 37$ dans chacun des cas suivants :
   a) Les collisions sont résolues par chaînage.
   b) Les collisions sont résolues par sondage linéaire, c'est-à-dire en plaçant chaque valeur dans la première case libre modulo m à partir de celle indicée par le haché de sa clé.
2. Proposer dans chacun des cas un algorithme permettant d'obtenir la valeur associée à une clé si elle est présente dans la table (on renverra une erreur si ce n'est pas le cas). Quelle est leur complexité pire cas ? S'attend-t-on à ce que la complexité en pratique soit proche de cette complexité pire cas ?
3. Décrire ce qu'il se passe si après les opérations de la question 1 on supprime la valeur associée à la clé 83 puis on recherche la valeur associée à la clé 59. Que se passe-t-il dans le second cas ?

Remarque : De manière générale, les méthodes de gestion de collisions par adressage ouvert, c'est-à-dire les méthodes qui consistent en cas de collision à stocker les paires (clé,valeur) dans une autre alvéole que celle initialement prévue par le haché de la clé, engendrent des algorithmes de suppression plus compliqués que les méthodes chaînées. Pour plus de détails, consulter `https://algo.infoprepa.epita.fr/index.php/Epita:Algo:Cours:Info-Spe:Méthodes_de_hachage`

Exercice 3 *Résolution de collisions par coalescence*

On souhaite insérer dans une table de hachage les valeurs suivantes selon leur clé (entière) dans l'ordre suivant : (16, Rachel), (18, Julie), (6, Tom), (29, Octave), (50, Paula), (13, Sophie), (27, Aurélie), (31, Georges), (28, Ophélie). On utilise la fonction de hachage $h : x \mapsto x \bmod 11$.

Pour gérer les collisions, on utilise la méthode suivante. Chaque case de la table de hachage contiendra deux champs : un champ *valeur* permettant de stocker un couple (clé, valeur) et un champ *lien* permettant de stocker une valeur entière initialement égale à -1. On ajoute à la fin de la table de hachage à 11 éléments 4 cases qui serviront de réserve. Lorsqu'un couple (clé, valeur) doit être placé dans une case déjà occupée, on se rend à la case indiquée par son champ lien jusqu'à ce qu'on trouve une case C dont le champ lien est encore égal à -1. On place alors le couple **dans la réserve à la case d'indice k le plus élevé** possible et on place dans le champ lien de la case C la valeur k.

1. Indiquer l'état de la table de hachage suite aux insertions proposées en début d'énoncé.
2. Expliquer comment rechercher la valeur associée à une clé avec une telle méthode de gestion des collisions.
3. Que se passe-t-il si on essaie d'insérer le couple (Pierre, 5) dans la table résultant de la première question ?

Le choix de la taille de l'espace de réserve est un problème épineux. On propose une nouvelle méthode de gestion des collisions : la résolution par coalescence. Cette dernière supprime l'espace de réserve et gère les collisions "comme si" cet espace existait. Plus précisément, elle remplace la proposition en gras dans la méthode précédente par "dans la case libre de la table ayant l'indice k le plus élevé possible".

4. Reprendre la question 1 avec cette nouvelle méthode.
5. Que se passe-t-il lorsqu'on insère le couple (31, Georges) ? Que valent $h(31)$ et $h(50)$? Le couple $(31, 50)$ est-il une collision ?

Remarque : Lorsque deux clés se retrouvent dans la situation de la question 5, on parle de collision secondaire. Comme dans l'exercice 2, la suppression dans une table gérée par coalescence est un exercice délicat.

Exercice 4 *Hachage par extraction*

Une méthode simple pour hacher une clé consiste à en extraire des morceaux. Dans cet exercice, les clés sont des chaînes de caractères minuscules d'au moins cinq caractères. Pour les hacher, on extrait le deuxième et le cinquième caractère de la chaîne. Afin que le haché d'une clé soit un entier, on code la chaîne de deux caractères obtenue ainsi : on écrit bout à bout le code de ces deux caractères sur 6 bits avec la convention que 'a' est codé par 0, 'b' est codé par 1 ... puis on traduit en base 10 la séquence binaire obtenue. Remarquez que dans cet exemple, contrairement au cours, on commence par hacher avant de coder. Ainsi, le haché de la clé "tigre" est 516.

1. Donner le haché des clés "hello", "bonjour" et "coucou".
2. Que peut on dire des valeurs des hachés de ces deux dernières clés ? Proposer d'autres clés qui ont le même haché que "coucou".
3. Discuter de la pertinence de hacher une clé par extraction.

La situation observée ci-dessus inspire la règle générale suivante : une bonne fonction de hachage (c'est-à-dire une fonction pour laquelle les collisions ne sont pas trop fréquentes) doit faire intervenir tous les bits de la clé.

4. Dans cette question, les clés sont des entiers. On propose de les hacher (par division) avec la fonction $h : k \mapsto k \bmod 2^m$ avec $m \in \mathbb{N}$. Qu'en penser ?

On parle de fonctions de hachage dans d'autres contextes que celui des tableaux associatifs et en particulier en cryptographie. Une fonction de hachage cryptographique est une fonction $h : \{0,1\}^\star \to \{0,1\}^k$ qui associe à toute suite de bits m un haché $h(m)$ de taille fixe égale à k bits et qui vérifie trois propriétés :

1. Résistance à la préimage : pour tout $y \in \{0,1\}^k$, il est difficile (il faut de l'ordre de 2^k essais ; bien évidemment, cela présuppose que k soit suffisamment grand pour que ce nombre de tests soit irréalisable avec la technologie actuelle) de trouver $x \in \{0,1\}^\star$ tel $h(x) = y$.
2. Résistance à la seconde préimage : pour tout $x \in \{0,1\}^\star$, il est difficile (il faut de l'ordre de 2^k essais) de trouver $x' \neq x$ tel que $H(x) = H(x')$.
3. Résistance aux collisions : il est difficile (il faut de l'ordre de $2^{k/2}$ essais) de trouver $x, x' \in \{0,1\}^\star$ tels que $x \neq x'$ et $h(x) = h(x')$.

Ces fonctions sont utilisées dans de très nombreux contextes dont les exercices suivants donnent un bref aperçu.

Exercice 5 *Empreintes*

Afin de ne pas stocker les mots de passe des utilisateurs d'un système informatique en clair (ce qui serait catastrophique en cas de fuite de ces données), il est recommandé de les hacher par une fonction de hachage cryptographique et de stocker les hachés plutôt que les mots de passe.

1. Expliquer comment est effectuée la vérification d'un mot de passe saisi par l'utilisateur dans un tel contexte et pourquoi stocker les hachés des mots de passe offre une certaine sécurité en cas de fuite de ces données.

Remarque : Pour éviter certaines attaques, on préfère stocker les hachés "salés" des mots de passe : si m est le mot de passe, on choisit une chaîne aléatoire de caractères l, le sel, on calcule $h(m|l)$ où $|$ dénote la concaténation et on stocke $(h(m|l), l)$ à la place de $h(m)$.

La société X propose un service de sauvegarde longue durée très onéreux pour de gros volumes de données. L'entreprise Y est cliente de la société X : elle lui envoie les copies de centaines de milliers de fichiers. L'entreprise Y souhaite s'assurer que la société X n'est pas une escroquerie et qu'elle sera effectivement capable de lui restituer ses documents en cas de panne dans ses locaux. L'entreprise X demande donc à la société Y d'effectuer des simulations de restauration de données, autrement dit de lui apporter la preuve que cette société possède effectivement les sauvegardes de ses fichiers et qu'elle ne les a pas tout simplement supprimés après réception. Comme ces fichiers sont très nombreux et très lourds, leur transmission est coûteuse ; la société X propose donc le protocole de vérification suivant : "Chaque jour, demandez nous le haché par une fonction de hachage cryptographique H d'un fichier de votre choix. Nous vous fournirons le haché demandé, prouvant ainsi que nous disposons bien des données".

2. Expliquer pourquoi l'entreprise Y peut confirmer ou infirmer le fait que la valeur envoyée est bien le haché par H du fichier qu'elle a choisi.
3. Le protocole proposé par la société X prouve-t-il bien qu'elle dispose du fichier en question ?

Exercice 6 *Comment jouer à pile ou face par téléphone*

Alice et Bob sont en plein divorce et veulent tirer à pile ou face qui gardera la télévision. Seulement voilà, ils refusent de se voir et n'acceptent de communiquer que par téléphone. On suppose qu'ils disposent d'une ligne authentifiée : ils sont sûrs de parler l'un à l'autre et que ce qu'ils disent n'est pas déformé.

1. Expliquer pourquoi Alice ne peut pas simplement tirer une pièce et communiquer le résultat à Bob.

Dans la suite, on suppose que Alice et Bob se sont mis d'accord sur une (bonne) fonction de hachage cryptographique H prenant en entrée des chaînes de n bits.

2. Alice tire au hasard une chaîne de n bits $x_{n-1}...x_0$ qui représente une entier en base 2. Expliquer comment construire les événements "pile" et "face" à partir de cet entier.

3. Montrer qui, si Alice effectue le tirage avant le pari, elle peut calculer une quantité h à communiquer à Bob telle que : h ne révèle pas le résultat du tirage et Alice ne peut pas changer le résultat du tirage sans changer h.

4. Montrer que Bob peut parier (c'est à dire faire une déclaration du type "sur un pile, c'est moi qui garde la télé") et qu'Alice peut lui révéler après coup si l'événement qu'elle a tiré était pile ou face de manière à ce que Bob puisse vérifier qu'elle ne ment pas.

5. Récapituler un protocole permettant de tirer à pile ou face au téléphone de manière à ce que les deux interlocuteurs soient convaincus que l'autre ne triche pas.

Remarque : Bien que l'exercice soit assez peu sérieux, les méthodes employées le sont et permettent de mettre plusieurs partis d'accord quand bien même certains seraient indignes de confiance (corrompus, défectueux...).