Structures de données avancées

Hachage digital

Pr ZEGOUR DJAMEL EDDINE
Ecole Supérieure d'Informatique (ESI)
www.zegour.uuuq.com
email: d_zegour@esi.dz

HD : Hachage digital

Environnement du hachage digital

- **Chaque article est identifié par une clé primaire.**
- **Le fichier est constitué de cases numérotées 0, 1, 2, . . N.**
- **Chaque case contient au maximum b articles (b : capacité)**
- **Case : unité de transfert entre la RAM et le disque.**

$C = d_0 d_1 \ldots . d_i \ldots . d_{k-1} d_k$

d_i : digits d'un alphabet donné.

$k+1$ étant la longueur de la clé.

i est appelé le numéro (niveau) du digit

':' désigne la valeur maximale du digit.

'_' représenté la valeur minimale du digit.

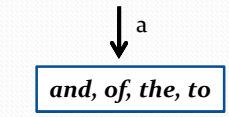
HD : Hachage digital

Illustration par un exemple (b = 4)

- **Séquence des clés à insérer (empruntée de KNUTH) : *the, of, and, to, a, in, that, is, i, it, for, as, with, has, his, he, be, not, by, but, have, you, which, are, on, or, her, had, at, from, this***
- **Le fichier est supposé créé avec seulement la case 0.**
- **Au départ, f(clé) = 0 quelque soit la clé.**
- **Donc, les 4 premières clés vont dans la case 0**
  Case 0 : (*and, of, the, to*)

HD : Hachage digital

Illustration

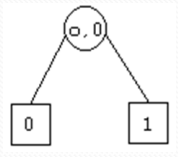


a, and, of, the, to

C' C'

Sélectionner la plus petite séquence de digits dans C' telle que C' < C'' : '0'

f(clé) = 0 si clé <= '0::... :'
= 1 si clé > '0::... :'



0 ***a and of***

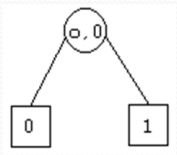
1 ***the, to***

HD : Hachage digital

Illustration

- **Par la suite, toutes les clés seront 'hachées' par la nouvelle fonction.**
- **la clé *in* est insérée dans la case 0,**
- **la clé *that* dans la case 1.**
- **L'insertion de la clé *is* provoque une collision sur la case 0.**

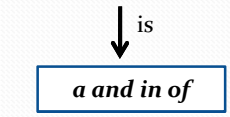
f(clé) = 0 si clé <= 'o::... :'
= 1 si clé > 'o::... :'



0	*a and in of*
1	*the, that, to*

HD : Hachage digital

Illustration

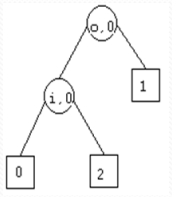


a, and, in, is, of

C' C'

Sélectionner la plus petite séquence de digits dans C' telle que C' < C'' : 'i'

f(clé) = 0 si clé <= 'i::... :'
= 2 si 'i::. :' < clé <= 'o::... :'
= 3 si clé > 'o::... :'



0	*a and in is*
1	*of*
2	*the, to*

HD : Hachage digital

Illustration

i

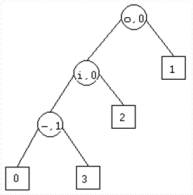
a and in is

a, and, i, in, is

C' C'

Sélectionner la plus petite séquence de digits dans C' telle que C' < C'' : 'i_'

f(clé)= 0 si clé <= 'i_::... :'
= 3 si 'i_::... :' < clé <= 'i::... :'
= 2 si 'i::... :' < clé <= 'o::... :'
= 1 si clé > 'o::... :'

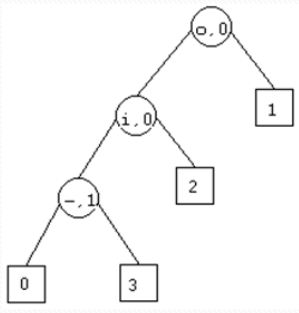


0	*a and i*
1	*of*
2	*the, to*
3	*in is*

HD : Hachage digital

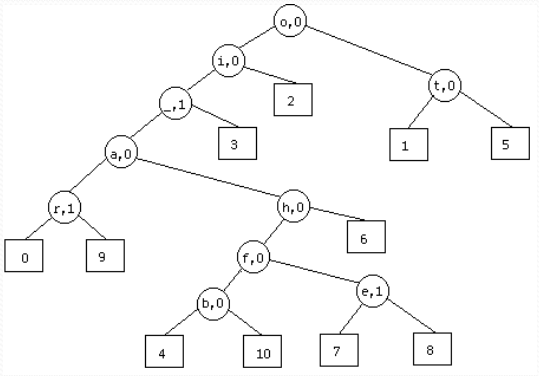
Transformation : clé ---> adresse

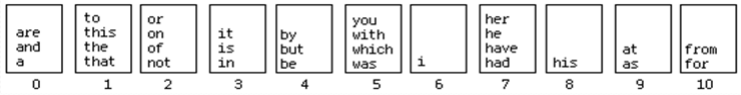
(Clé = $c_0c_1c_2$.....)
si c_0 > 'o' : m = 1
sinon
si c_0 > 'i' : m = 2
sinon
si c_0c_1 > 'i_' : m = 3
sinon
m = 0
finsi
finsi
finsi



HD : Hachage digital

Illustration





HD : Hachage digital

f(clé) = 0 si clé <= ‘mi::... :‘
= 1 si ‘mi::. :' < clé <= ‘m::... :'
= indéfini si clé > ‘m::... :'

Illustration

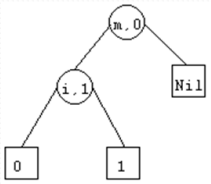
meet

mat mile monk mut

mat meet mile monk mut

C’ **C’**

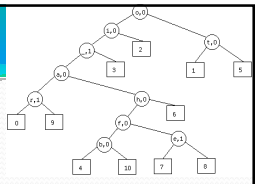
Sélectionner la plus petite séquence de digits dans C' telle que C' < C“ :
‘mi’



0 ***mat meet mile***

1 ***monk mut***

HD : Hachage digital



Propriétés

- **Le graphe généré par la méthode est appelé 'trie' ou arbre digital.**
- **l'arbre digital représente une fonction de hachage qui est modifiée dynamiquement. → l'appellation 'trie hashing' (hachage digital)**
- **Fichier trié en ordre ascendant des clés.**
- **Le fichier s'étend linéairement avec l'arbre à raison d'un nœud interne par case.**
- **Pas de débordements.**
- **Toute recherche de clé avec ou sans succès coûte au plus un accès disque.**

HD : Hachage digital

Opérations

- **<u>Recherche</u> :**
- **A chaque comparaison , choisir soit la branche gauche, soit la branche droite.**
- **Le parcours s'arrête quand une feuille est trouvée.**

<u>Considérations :</u>

Soit Cle la clé à rechercher.

Soit $(Cle)_i$ les i+1 premiers digits de Cle.

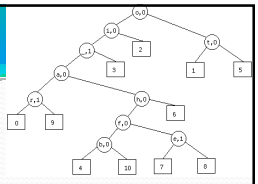
Soient C' et C'' des chaînes de caractères initialisées à ':'.

C'_i dénote le i+1-ème digit de C'.

Soit (d,i) le nœud visité si le nœud est interne, autrement soit A la feuille visitée.

$(C)_{-1}$ désigne la chaîne vide

HD : Hachage digital



Algorithme de transformation Clé → adresse
(Au départ n est la racine de l'arbre)

C', C " ← ""
TANTQUE interne (n) //n=(d, i) ;

C' ← $(C")_{i-1}$ d
SI $(Cle)_i \leq C'$:
C" ← C'
n ← fg(n)
SINON
n ← fd(n)
FINSI
FINTANTQUE
Retourner la feuille (adresse de case)

HD : Hachage digital

Opérations

- **Insertion** (En cas d'éclatement)

✓ Constituer la suite ordonnée des (b+1) clés composée des b clés de la case surchargée et de la nouvelle clé.
Soit C' la clé du milieu de la séquence des (b+1) clés, C" la clé maximale.

✓ Sélectionner la plus petite séquence de digits dans C' telle que C' < C". Soit $c'_0 c'_1 \ldots c'_k$ cette séquence.

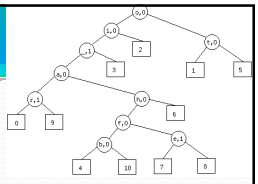
✓ Retrouver le nombre i de digits qui existent déjà dans L'arbre digital.

✓ Générer (k-i-1) nœuds Nil avec les digits $c'_{i+1}, c'_{i+2}, \ldots c'_{k-1}$

✓ Générer le nœud correspondant au digit c'_k.

✓ Diviser la suite des (b+1) clés entre la case qui a subit la collision et la prochaine case à allouer.

HD : Hachage digital



Opérations

- **Suppression**

✓**Peut entraîner une fusion de cases dans les cas suivants :**

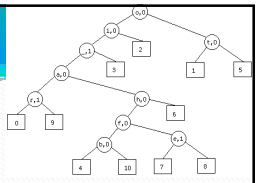
(i) La case sœur existe.

(ii) Le nombre des articles dans les deux cases sœurs est au plus égal à b.

✓**la fusion consiste à grouper tous les articles dans la case de gauche. La case droite est libérée et les articles de la dernière case N sont alors transférés vers cette case. (Éliminer les trous)**

✓**Une façon de retrouver le parent de la case N est de rechercher un élément quelconque se trouvant dans la case N.**

HD : Hachage digital



Opérations

- **Suppression (suite)**

✓ **Le test des conditions (i) et (ii) se fait uniquement quand la case subissant la division aura un nombre d'articles proche de b/2.**

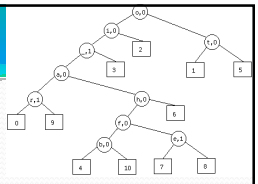
✓ **L'opération de suppression est une opération inverse à l'opération d'insertion.**

✓ **Elle permet de contracter l'arbre digital.**

✓ **Si une case devient vide avant qu'une case sœur apparaisse, alors cette case est remplacée par Nil.**

✓ **possibilité de fusions en cascade**

HD : Hachage digital



Opérations

- **Requête par intervalle**

✓ **un fichier construit par la technique du hachage digital est ordonné.**

✓ **Dans l'arbre digital, les cases sont ordonnées de la gauche vers la droite.**

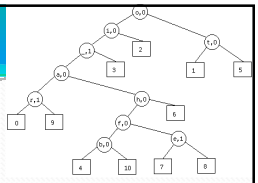
✓ **Une requête par intervalle consiste à trouver toutes les clés comprises entre 2 clés a et b données (a < b).**

✓ **Une façon simple consiste à parcourir l'arbre digital en inordre, préordre ou postordre.**

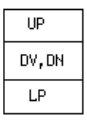
✓ **Une autre façon de réaliser cette opération est de chaîner les cases.**

✓ **On peut aussi examiner les clés en ordre décroissant. dans ce cas, on utilisera l'inordre inverse(preordre inverse ou postordre inverse) ou chaîner les cases en sens inverse.**

HD : Hachage digital



Représentation mémoire



Structure d'un noeud

- 4	2	3	- 5	5	6	- 7	8	9	10
o, 0	i, 0	_, 1	a, 0	t, 0	h, 0	f, 0	e, 1	r, 1	b, 0
- 1	- 2	- 3	- 8	1	- 6	- 9	7	0	4
0	1	2	3	4	5	6	7	9	10

Table représentant l'arbre digital

HD : Hachage digital

Encombrement

- **6 octets par nœud. (si fichier de l'ordre de 2^{15} cases)**

- **Quelques raffinements :**

✓**Optimisation sur DV et DN**

- **Si le digit est un bit, le champ DV n'est pas nécessaire. Comme une clé ne dépasse pas 16 digits quatre bits suffisent pour DN. Donc, au total 4,5 ou 5 octets**
- **Si digit numérique : 4 bits suffisent pour DV et DN chacun. Donc, chaque élément occupe 5 octets.**

✓**Optimisation sur les pointeurs**

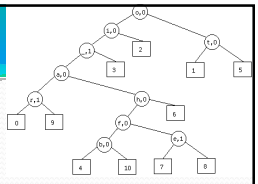
- **Si fichier ne dépasse pas 2^l cases alors l+1 bits suffisent pour chaque pointeur.**
- **Si fichier n'excède pas 2K cases (K = 1024), trois octets suffisent pour représenter à la fois LP et UP.**

HD : Hachage digital

Autres représentations

- **Utiliser des représentations mémoire beaucoup plus compactes appelées représentations séquentielles.**
- **Les nœuds internes suivent un ordre prédéfini.**
- **Éviter la représentations des pointeurs internes, donc on peut gagner beaucoup d'espace mémoire.**
- **En contrepartie, l'algorithmique est plus complexe.**

HD : Hachage digital



Exemple de représentation

- **a r <0> <9> b <4> f <10> h e <7> < 8> i _ <6> <3> o <2> t <1> < 5>**

- **<= 'ar:...' c'est o**
- **> 'ar:' et <= 'a:' c'est 9**
- **> 'a:' et < 'b:' c'est 4**
- **Etc...**

HD : Hachage digital

Analyse des performances

- **<u>Facteur de chargement</u>**
  Similaire à celui des arbres-B .
  70% : insertions aléatoires
  50% : insertions en ordre ascendant.
- **<u>Taille de la fonction d'accès.</u>**

✓ **Le nombre de cases est pratiquement égal au nombre de nœuds internes**

✓ **Si M désigne le nombre d'octets nécessaires pour l'arbre digital on aura M=6n dans le cas des digits alphabétiques**

✓ **Donc pour M=3K, on peut avoir un fichier de 35 000 articles si on prend b=100.**

✓ **Pour des mémoires de 12 et 64K, on peut construire des fichiers de 140000 et de 760000 articles respectivement pour la même valeur de b.**

HD : Hachage digital

Analyse des performances

- **Avec l'hypothèse que l'arbre digital peut résider en mémoire, toute recherche de clé est exécuté en un accès disque au plus.**
- **Une insertion coûte 2 accès s'il elle n'entraîne pas une collision et 3 accès si elle l'entraîne.**
- **Pour b=100, une insertion coûte en moyenne 2,03 accès.**

HD : Hachage digital

Conclusion

- **Le hachage digital est une technique d'accès aux fichiers conçue en 1981**
- **La technique est aussi bien consacrée aux fichiers statiques qu'aux fichiers dynamiques**
- **Le fichier est ordonné.**
- **Son facteur de chargement est de 70% pour des insertions aléatoires**
- **On peut construire des fichiers de quelques millions d'articles et dans lesquels toute recherche de clé est exécutée en un accès disque au plus.**