Ch8-Hashing

Hashing

定義:

是一種資料儲存與擷取之技術，即欲存取Data X之前，必須先經過Hashing Function計算求出Hashing Address(or Home Address)，再到Hash Table中對應的Bucket中存取Data X，而Hash table結構是由B個buckets組成，每個bucket有S個Slots，每個Slot可存一筆Data

* Hash Table大小 = B\*S筆

優點:

1. 搜尋Data之前，無須事先排序過

2. 再沒有Collision情況下，Search X之time為: O(1)，與資料量n無關

3. 保密/安全性高，再不知Hashing Function下，很難取得Data

4. 可做為Data壓縮之用途

Collision(碰撞)

不同的Data(X,Y)經過Hashing function計算後，得出相同的Hashing Address稱之

即H(X)=H(Y)

Overflow(溢位)

當Collision發生後，且對應的Bucket已滿，則稱為Overflow

Note:

1. 有Collision，不一定會Overflow

2. 若每個bucket只有一個Slot(or 只能放一個Data)，則collision = overflow

Identifier Density and Loading Density

令T為identifier總數，n為使用identifier個數，B\*S為Hash table size

Identifier Density: n/T

Loading Density(負載密度): n/(B\*S)

若Loading Density越大，代表Hash Table利用度高，但相對Collision機會也大

Hashing Function Design

一個良好的Hashing function設計，應滿足下列3個Criteria

1. 計算宜簡單，不要過度複雜

2. Collision宜少

3. 不要造成Hash Table局部儲存之情況，要夠均勻才好

Perfect Hashing Function

此函數保證不會產生Collision

Uniform Hashing Function

N個Data存入Hash Table之B個Bucket時，每個Bucket內之Data個數大約相等，即n/B，則此函數為Uniform

常見的4種設計

1. Middle Square

將鍵值平方後，取中間適當位數，作為Hashing Address

1. Mod(or Divide)取餘數

如: H(X) = X % M

M的選擇最好滿足:

1. 質數

2. M不宜整除)，r為基底，k與a為很小的正整數

1. Folding Addition

將鍵值切成幾個相同長度之片段(P1~Pm)，再將這些片段加總，即得Hashing Address，相加方法有兩種: Shift(位移)、Bounding(邊界)

Shift:每個片段相加

Bounding:每個片段相加，偶數片段反向

1. Digits Analysis(位數值分析)

適用於資料事先知道之情況，分析每個資料在不同位數之值域分布情況，若分布很集中，則捨棄該位數，若散亂，則挑選，挑出之位數組成Hashing Address

例: 02-7415089 => 158

02-7434079 => 347

02-7424139 => 243

02-7458069 => 586

02-7473029 => 732

Overflow處理方法:

Linear Probing(線性探測)

Quadratic Probing(二次方探測)

Double Hashing (Algo版)

Chain (Chaining)

Rehashing (再散置)

Linear Probing (or Linear Open Addressing Mode)

當H(x)發生overflow時，則探測(H(X)+i)%B，B為Bucket數，i=1,2,3,…,B-1，直到有Bucket可存or Table全滿為止

分析:

優點:

1. 簡單、易於實施

2. 保證Table空間可以充分利用

缺點:

容易發生Primary clustering現象，造成Search/Insert/Delete X等時間大幅增加之問題

Primary clustering: 具有相同Hashing Address之Data極易占用相鄰的Buckets存放，形成群聚

Quadratic 探測(二次方)

當H(X)發生Overflow時，則探測(H(X))%B，或(H(X))%B，B為Bucket數目，I=1,2,3…,[B/2]，直到有Bucket可存，或探測位置接滿，無法存入為止

分析:

優點: 解決Primary Clustering Problem

缺點:

1. 有Secondary Clustering Problem

=> 具有相同的Hashing Address之Data，他們的探測軌跡均相同，後進之Data仍需花大量時間去探測先前Data已探測之位置，才能找到可用Bucket

所以 Search/insert/Delete 時間會大幅增加

2. Table Space不保證可充分利用

Double Hashing

令H1為Hashing Function，當H1(X)發生Overflow，則探測(H1(X)+i\*H2(X))%B

其中，B為Bucket數目，i從1,2,3…,B-1，H2(X)為探測間隔(Span)

，通常型式為: R-(X%R)，R為質數

直到找到空的Bucket or 探測位置皆滿為止

分析:

優點: 解決Primary Clustering and Second Clustering problem

缺點: Table Space不保證充分利用

Chain(Chaining)

具有相同的Hashing Address的Data均置入同一個Bucket去，而Bucket內之Data彼此已Linked List串連一起，Chain是屬於Closed address mode

而其他方法都是Open address mode

最差之Search Time in chain 是O(n)  
如何改善，如果串列裡面用AVL Tree、R-B Tree之結構，則改善時間為O(logn)

Rehashing:

提供一系列的Hashing Function H1~Hm，當H1發生overflow，則改用H2，以此類推，直到找到Bucket可存，或Hashing Function用完為止