MATA54 - Estruturas de Dados e Algoritmos II

Hashing Universal e Hashing Perfeito

Flávio Assis
Versão gerada a partir de slides do Prof. George Lima

IC - Instituto de Computação

Salvador, setembro de 2021

Motivações

Hashing universal

O uso de uma única função de hashing pode resultar em mau desempenho de um sistema para conjuntos específicos de chaves, se a função resultar em muitas colisões (potencialmente todas as chaves podem colidir).

- Idéia: alterar a função de hashing e escolhê-la aleatoriamente

Motivações

Hashing universal

O uso de uma única função de hashing pode resultar em mau desempenho de um sistema para conjuntos específicos de chaves, se a função resultar em muitas colisões (potencialmente todas as chaves podem colidir).

- Idéia: alterar a função de hashing e escolhê-la aleatoriamente

Hashing perfeito

Se o conjunto de chaves é conhecido, uma função hashing pode ser projetada para garantir $O(1)$ no pior caso com pouco uso de espaço

- Há métodos para se encontrar uma tal função?

Motivações

Hashing universal

O uso de uma única função de hashing pode resultar em mau desempenho de um sistema para conjuntos específicos de chaves, se a função resultar em muitas colisões (potencialmente todas as chaves podem colidir).

- Idéia: alterar a função de hashing e escolhê-la aleatoriamente

Hashing perfeito

Se o conjunto de chaves é conhecido, uma função hashing pode ser projetada para garantir $O(1)$ no pior caso com pouco uso de espaço

- Há métodos para se encontrar uma tal função?

Solução a ser estudada:

- Hashing perfeito a partir de funções hashing universais [Cormen *et al.* 2009]

Hashing Universal

Definição

Seja $\mathcal{H}$ um conjunto finito de funções hashing que mapeiam um conjunto U de chaves no conjunto $\{0, 1, \ldots, m-1\}$ para um dado m. $\mathcal{H}$ é universal se, para quaisquer chaves distintas k e k^* em U, o número de funções hashing $h \in \mathcal{H}$ tais que $h(k) = h(k^*)$ é no máximo $\frac{|\mathcal{H}|}{m}$. Analogamente,

$$\forall k, k^* \in U, k \neq k^*, \Pr_{h \in \mathcal{H}}\{h(k) = h(k^*)\} \leq \frac{1}{m}$$

Exemplo de Conjunto Universal de Funções Hashing

Sejam:

- p um primo tal que toda chave possível k esteja no intervalo de 0 a $p-1$ (inclusive) e $p > m$
- $Z_p = \{0, 1., ..., p-1\}$
- $Z_p^* = \{1, 2, ..., p-1\}$
- as funções $h_{a,b}(k) = ((ak+b) \bmod p) \bmod m$, para $a \in Z_p^*$ e $b \in Z_p$

A família de funções de hashing

$$\mathcal{H}_{p,m} = \{h_{a,b} : a \in Z_p^* \text{ e } b \in Z_p\}$$

é universal (**Teorema 11.5 [Cormen et al. 2009]**).

Exemplo de Conjunto Universal de Funções Hashing

Seja o conjunto de chaves $U = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$ e $m = 9$.

Para $p = 101$, temos:

- $Z_{101} = \{0, 1., ..., 100\}$
- $Z^*_{101} = \{1, 2, ..., 100\}$

O seguinte conjunto de funções é universal:

$h_{1,0}(k) = ((k) \bmod 101) \bmod 9$
$h_{1,1}(k) = ((k+1) \bmod 101) \bmod 9$
$h_{1,2}(k) = ((k+2) \bmod 101) \bmod 9$
...
$h_{1,100}(k) = ((k+100) \bmod 101) \bmod 9$

$h_{2,0}(k) = ((2k) \bmod 101) \bmod 9$
$h_{2,1}(k) = ((2k+1) \bmod 101) \bmod 9$
$h_{2,2}(k) = ((2k+2) \bmod 101) \bmod 9$
...
$h_{2,100}(k) = ((2k+100) \bmod 101) \bmod 9$

$h_{3,0}(k) = ((3k) \bmod 101) \bmod 9$
$h_{3,1}(k) = ((3k+1) \bmod 101) \bmod 9$
...
$h_{100,0}(k) = ((100k) \bmod 101) \bmod 9$
$h_{100,1}(k) = ((100k+1) \bmod 101) \bmod 9$
...

$h_{100,100}(k) = ((100k+100) \bmod 101) \bmod 9$

Hashing Perfeito

Definição

Um método de hashing é **perfeito** se o número de acessos é garantidamente $O(1)$ no pior caso para uma operação de busca, remoção ou inclusão.

Observações

- O método a ser apresentado só se aplica a um conjunto fixo de chaves
- Aplicações: palavras reservadas em um compilador, diretórios em um sistemas de arquivos fixo (ex.: CD-ROM), palavras de um dicionário, etc..

Método: Hashing Perfeito em Dois Níveis

Idéia básica: hashing em dois níveis

- Uma função hash h de um conjunto de funções de hashing universal é usada para mapear n chaves em n endereços numa tabela T
- Usando a função h, cada entrada em T terá n_i elementos
- Funções hash h_i de um conjunto de funções hashing universal são usadas para mapear as n_i chaves mapeadas no endereço i em uma outra tabela T_i com n_i^2 endereços

$$
\begin{array}{r|c|cl}
 & T & & T_i \\
\hline
0 & n_0 & \rightarrow & \begin{array}{|c|c|c|c|}\hline 0 & 1 & \cdots & n_0^2 \\ \hline\end{array} \\
\hline
1 & n_1 & \rightarrow & \lambda \\
\hline
2 & n_2 & \rightarrow & \begin{array}{|c|c|c|c|}\hline 0 & 1 & \qquad\cdots\qquad & n_2^2 \\ \hline\end{array} \\
\hline
 & \vdots & & \\
\hline
n-1 & n_{n-1} & \rightarrow & \begin{array}{|c|c|c|c|}\hline 0 & 1 & \cdots & n_{n-1}^2 \\ \hline\end{array} \\
\hline
\end{array}
$$

Exemplo: Primeiro Nível

Usando hashing perfeito para armazenar as chaves:

$$K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$$

para $m = 9$.

Usando $p = 101$:

$Z_{101} = \{0, 1, 2, ..., 100\}$

$Z_{101}^{*} = \{1, 2, ..., 100\}$

Classe universal de funções de hashing:

$$\mathcal{H}_{101,9} = \{h_{a,b} : a \in Z_{101}^{*} \text{ e } b \in Z_{101}\}$$

onde:

$$h_{a,b}(k) = ((ak + b) \bmod 101) \bmod 9$$

Exemplo: Primeiro Nível

Usando hashing perfeito para armazenar as chaves:

$$K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$$

Usando, por exemplo, $a = 3, b = 42, p = 101$ e com $m = 9$:

$$h_{3,42}(k) = ((3k + 42) \bmod 101) \bmod 9$$

h(10)=0
h(22)=7
h(37)=7
h(40)=7
h(52)=7
h(60)=2
h(70)=5
h(72)=2
h(75)=2

0	$\longrightarrow$ 10
1	$\longrightarrow \lambda$
2	$\longrightarrow$ 60, 72, 75
3	$\longrightarrow \lambda$
4	$\longrightarrow \lambda$
5	$\longrightarrow$ 70
6	$\longrightarrow \lambda$
7	$\longrightarrow$ 22, 37, 40, 52
8	$\longrightarrow \lambda$

Hashing Perfeito em Dois Níveis

Foi visto anteriormente que a probabilidade de colisão de duas chaves usando uma função de um conjunto de funções de hashing universal é menor ou igual a $1/m$.

Teorema 11.9 [Cormen et al. 2009]

Se armazenarmos n chaves em uma tabela hash de tamanho $m = n^2$ usando uma função hash h escolhida aleatoriamente de uma classe universal de funções hashing, então a probabilidade de haver colisão é menor que 1/2.

Exemplo: Segundo Nível

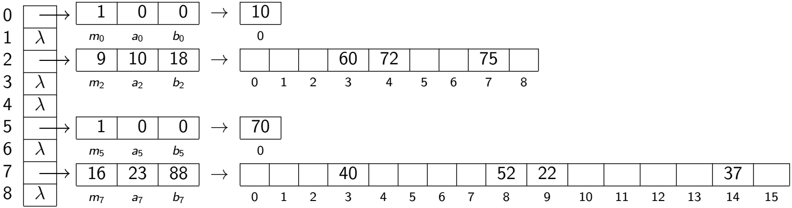
Usando hashing perfeito para armazenar as chaves:

$$K = \{10, 22, 37, 40, 52, 60, 70, 72, 75\}$$

Usando:

$h_2(k) = ((10k + 18) \text{ mod } 101) \text{ mod } 9$

$h_7(k) = ((23k + 88) \text{ mod } 101) \text{ mod } 16$



Hashing Perfeito em Dois Níveis

1. Escolha uma função hash da classe $\mathcal{H}_{p,n}$ tal que haja poucas colisões. Como a probabilidade de haver colisões é baixa, espera-se que poucos testes sejam necessários.

2. Para n_i chaves que são mapeadas no endereço i

   2.1 Escolha uma função h_i da classe universal de funções hashing $\mathcal{H}_{p,n_i^2}$

   2.2 Se houver colisões entre as n_i chaves usando-se a h_i escolhida, descarte h_i e repita o passo 2.1. Como a probabilidade de haver colisão é menor que $1/2$, espera-se que poucos testes sejam necessários.

Espaço para Hashing Perfeito em Dois Níveis

Quanto espaço seria necessário para usar o método descrito?

O espaço esperado de armazenamento é $O(n)$.

Teorema 11.10 [Cormen et al. 2009]

Se armazenarmos n chaves em uma tabela hash de tamanho $m = n$ usando uma função de hash h escolhida aleatoriamente de uma classe universal de funções de hashing então:

$$E\left[\sum_{j=0}^{m-1} n_j^2\right] < 2n$$

Espaço para Hashing Perfeito em Dois Níveis

Adicionalmente, a probabilidade de se precisar de mais do que $4n$ é menor que $1/2$.

Corolário 11.2 [Cormen et al.]

Se armazenarmos n chaves em uma tabela hash de tamanho $m = n$ usando uma função de hash h escolhida aleatoriamente de uma classe universal de funções de hashing e estabelecermos o tamanho de cada tabela secundária como sendo $m_j = n_j^2$ para $j = 0, 1, 2, ..., m - 1$ então a probabilidade de que o total de armazenamento necessário para as tabelas hash secundárias ultrapasse $4n$ é menor do que 1/2.