Endereçamento Aberto

ACH2002 - Introdução à Ciência da Computação II

Delano M. Beder

Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo
dbeder@usp.br

11/2008

Revisado pelo professor Luciano Antonio Digiampietri em 11/2012.

Material baseado em slides do professor Marcos L. Chaim

Endereçamento Aberto

- No endereçamento aberto, todos os elementos estão armazenados na *própria* tabela hash.
- Ou seja, cada entrada da tabela contém um elemento do conjunto dinâmico ou null.
- Ao procurar por um elemento, examinamos sistematicamente as posições na tabela até encontrarmos o elemento desejado... ou não encontrá-lo.
- Diferença do encadeamento:
  - *não* há nenhuma lista e nenhum elemento armazenado fora da tabela;
  - a tabela hash pode ficar *cheia* de forma que não são permitidas novas inserções.

Endereçamento Aberto

Vantagem do endereçamento aberto:

- evita por completo o uso de listas encadeadas;
- ao invés de seguir os ponteiros nas listas, *calculamos* a seqüência de posições a serem examinadas;
- uso mais eficiente do espaço alocado para a tabela hash;
- o espaço não alocado para as listas pode ser usado para aumentar o tamanho da tabela hash, o que implica menor número de colisões.

Endereçamento Aberto – Inserção

- É feita uma *sondagem*, isto é, um exame sucessivo, da tabela hash até encontrarmos uma posição vazia na qual seja possível inserir a chave.
- Ao invés de fazer a sondagem na ordem 0, 1, .., m - 1 (o que exige tempo $\Theta(n)$), a seqüência de posições examinadas depende da *chave que está sendo inserida*.
- O que vem a ser isto?

Endereçamento Aberto – Inserção

- Estendemos a função hash com o objetivo de incluir o número de sondagens (a partir de 0) como uma segunda entrada. Desse modo, a função hash se torna:

$$h : U \times \{0, 1, ..., m-1\} \rightarrow \{0, 1, ..., m-1\}$$

onde U é o universo de chaves.

- Com o endereçamento aberto, exige-se que, para toda chave k, a *seqüência de sondagem* seja uma permutação de $< 0, 1, ..., m-1 >$. Por quê?

Endereçamento Aberto – Inserção

- Considere as seguintes simplificações:
  - os elementos da tabela hash são compostos apenas das chaves sem informações satélite (e.g., o significado da palavra é uma informação satélite);
  - além disso, vamos supor que as chaves são número naturais (mapear seqüências de caracteres para números naturais vocês já sabem!).
- A partir dessas suposições, são apresentados algoritmos para inserção, busca e eliminação na tabela hash escritos em pseudo-código.

Endereçamento Aberto – Inserção

insereHash(T,k)

```
begin
 i = 0
 j = h(k,i)
 while (i != m) do
     if (T[j] == NIL) then
          T[j] = k
           return j
     else
          i = i + 1
     fi
     j = h(k,i)
 od
 return -1 // Estouro da tabela hash
end
```

Endereçamento Aberto – Busca

buscaHash(T,k)

```
begin
 i = 0
 j = h(k,i)
 while (i != m) and (T[j] != NIL) do
     if (T[j] == k) then
          return j
     fi
     i = i + 1
     j = h(k,i)
 od
 return -1  // Não foi encontrada a chave k
end
```

Endereçamento Aberto – Eliminação

eliminaHash(T,k)

```
begin
 i = 0
 j = h(k,i)
 while (i != m) and (T[j] != NIL) do
     if (T[j] == k) then
          // Marca a posição j como eliminada
          T[j] = |eliminado|
          return j
     fi
     i = i + 1
     j = h(k,i)
 od
 // A chave k não está presente na tabela hash
 return -1
end
```

Endereçamento Aberto – Observações

A eliminação é feita colocando uma *marca* "eliminado"e não o valor NIL (igual a null no Java). Por quê?

- O problema é que se colocarmos o valor NIL na posição eliminada o algoritmo de busca não irá encontrar as demais chaves incluídas *depois* da chave eliminada.
- Colocando a *marca* sabemos que a posição está livre, mas há outras chaves que vêm depois dela e devem ser verificadas durante uma busca.
- o algoritmo de inserção pode ser modificado para incluir quando encontrar uma posição marcada como "eliminado".
- Problema: tempo de pesquisa não depende mais somente do número elementos presentes na tabela, mas também do número de elementos eliminados.

Endereçamento Aberto

Uma questão ainda fica em aberto. Como devem ser criadas as funções hash que recebem dois parâmetros:

- $h(k, i)$ onde $k \in U$ e $i \in \{0,1,...,m-1\}$.

Três técnicas são comumente usadas:

1. Sondagem linear;
2. Sondagem quadrática;
3. Hash duplo.

Sondagem Linear

Dada uma função hash comum $h' : U \rightarrow \{0,1,...,m\text{-}1\}$, chamada de *função hash auxiliar*, o método de *sondagem linear* usa a função hash:

- $h(k,i) = (h'(k) + i) \bmod m$

onde $i = 0,1,...,m\text{-}1$ e *mod* é a operação que retorna o resto de uma divisão (e.g., equivalente ao operador % do Java).

Valor de i	Posição sondada
0	T[$h'(k)$]
1	T[$h'(k)$+1]
...	...
...	T[m-1]
...	T[0]
...	T[1]
...	...
m-1	T[$h'(k)$-1]

Sondagem Linear

Observações:

- A posição inicial $h'(k)$ de sondagem determina toda a seqüência posterior.
- Como conseqüência, só existem m seqüências de sondagem distintas.
- Fácil de implementar.
- Sofre de um problema conhecido como *agrupamento primário*.

Sondagem Linear

Agrupamento primário:

- Longas seqüências de posições ocupadas são construídas, aumentando o tempo médio de pesquisa.
- Surgem agrupamentos, pois uma posição vazia precedida por i posições completas é preenchida em seguida com probabilidade (i+1)/m.
- Seqüências de posições ocupadas tendem a ficar mais longas e o tempo médio de pesquisa aumenta.
- Gera no máximo m seqüências distintas, ou seja, número possível de seqüências é $\Theta(m)$.

Sondagem Quadrática

A *sondagem quadrática* utiliza uma função hash da froma:

- $h(k,i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$

onde h' é uma função hash auxiliar, c_1 e $c_2 \neq 0$ são constantes auxiliares e $i = 0,1,...,m-1$.

Exemplo: h(k,i) = (h'(k) + 0.5*i = 0.5*i*i) mod 16, onde

m = 16, h'(k) = k mod 16, c1 = 0.5 e c2 = 0.5

Sondagem Quadrática

- A posição inicial sondada é T[$h'(k)$]; posições posteriores são deslocadas por quantidades que dependem de forma quadrática do número da sondagem i.

- Funciona melhor que a sondagem linear, mas para usar complementamente a tabela hash, os valores de c_1, c_2 e m são limitados.

- Se duas chaves têm a mesma posição de sondagem inicial, então suas seqüências de sondagem são iguais. Exemplo:
  $h(k_1, 0) = h(k_2, 0) \Rightarrow h(k_1, i) = h(k_2, i)$.
  - Esta situação é caracterizada como *agrupamento quadrático*.

- Analogamente à sondagem linear, a primeira sondagem determina a seqüência inteira, ou seja, o número de seqüências possíveis é $\Theta(m)$.

Hash Duplo

O hash duplo é um dos melhores métodos disponíveis para endereçamento aberto, porque as permutações produzidas têm muitas características de permutações escolhidas aleatoriamente.

O *hash duplo* usa uma função hash da forma:

- $h(k,i) = (h_1(k) + ih_2(k)) \bmod m$

onde h_1 e h_2 são funções hash auxiliares.

Valor de i	Posição sondada
0	T[$h_1(k)$]
1	T[$(h_1(k) + h_2(k)) \bmod m$]
2	T[$(h_1(k) + 2h_2(k)) \bmod m$]
...	...
m-1	T[$(h_1(k) + (m-1)h_2(k)) \bmod m$]

Hash Duplo

Observações:

- Diferentemente das sondagens quadrática e linear, a seqüência de sondagem depende da chave k de duas maneiras.
- A posição de sondagem inicial e o deslocamento, ambos, podem variar.

Questão importante: como escolher h_1 e h_2 ?

Hash Duplo

Para que a tabela hash inteira seja pesquisada, o valor de $h_2(k)$ e o tamanho m da tabela hash devem ser primos entre si (a e b são primos entre si se o máximo divisor comum for 1).

Formas de conseguir isto:

1. Fazer m uma potência de 2 e h_2 gerar sempre um número ímpar.
2. Fazer m igual a um primo e projetar h_2 para retornar um inteiro positivo sempre menor que m.

Hash Duplo

Para o caso 2, supondo m um número primo, podemos ter h_1 e h_2:

1. $h_1(k) = k \bmod m$,
2. $h_2(k) = 1 + (k \bmod m')$,

onde m' é escolhido com um valor ligeiramente menor que m (digamos, $m - 1$).

Exemplo:

- Para $k = 123456$, $m = 701$ e $m' = 700$, tem-se $h_1(123456) = 80$ e $h_2(123456) = 257$.
- Portanto, a primeira posição sondada é de número 80; as demais estão separadas por 257 posições.
- Ou seja: 80, 337, 594, 150, ...

Hash Duplo

O hash duplo é um aperfeiçoamento em relação à sondagem linear e quadrática:

- o número possível de seqüências geradas é proporcional a m^2, pois cada par $< h_1(k), h_2(k) >$ gera uma seqüência distinta.

Neste sentido, o hash duplo é mais próximo do desempenho ideal do *hash uniforme*.

- No hash uniforme, a função $h(k, i)$ pode gerar qualquer permutação das m posições, isto é, o número possível de seqüências seria $m!$, ou seja, $\Theta(m!)$.
- O hash uniforme é difícil de implementar; na prática, utiliza-se aproximações como o hash duplo.

Resumo

Endereçamento aberto é uma alternativa ao tratamento de colisões utilizando encadeamento.

Vantagens:

- Não utiliza apontadores/listas;
- Uso mais eficiente do espaço alocado para a tabela hash.

Desvantagens:

- Eliminação mais complicada; influencia o desempenho da busca.
- A tabela hash pode ficar *cheia*, com todos os seus elementos ocupados.

Resumo

Três técnicas para implementar o endereçamento aberto:

1. Sondagem linear $\Rightarrow$ número de seqüências possíveis é $\Theta(m)$. Problema: agrupamento primário.
2. Sondagem quadrática $\Rightarrow$ número de seqüências possíveis é $\Theta(m)$. Problema: agrupamento quadrático.
3. Hash duplo $\Rightarrow$ número de seqüências possíveis é $\Theta(m^2)$. Mais próximo do hash uniforme.

Exercícios

1. Considere a inserção das chaves 10, 22, 31, 4, 15, 28, 17, 88, 59 em uma tabela hash de comprimento m = 11 usando o endereçamento aberto com a função hash primário $h'(k) = k \bmod m$. Ilustre o resultado da inserção dessas chaves com
   - o uso da sondagem linear
   - o uso da sondagem quadrática com $c_1 = 1$ e $c_2 = 3$ e
   - o uso do hash duplo com $h_2(k) = 1 + (k \bmod (m-1))$
2. Escreva o código Java para os algoritmos insereHash(), buscaHash() e eliminaHash() apresentados. Implemente as funções de hash descritas no Exercício 1 e execute o exemplo contido nele.
3. Modifique o algoritmo insereHash() para tratar a *marca* "eliminado".