

Técnicas de programação para Games

Aula08

Hash

Professor Mestre: Adilson Lopes Khouri

4 de novembro de 2018

Sumário

Hash

Cronograma

Aula	Conteúdo
12/04/2018	XP e banco de dados
17/04/2018	Introdução de estruturas de dados
24/04/2018	Arrays / Matrizes e Ordenação
26/04/2018	Recursão
03/05/2018	Lista Ligada
08/05/2018	Pilha, Fila
10/05/2018	Hash
15/05/2018	Árvore Binária
17/05/2018	Heap
22/05/2018	Grafos
24/05/2018	Prova

Hash

- ▶ São estruturas de dados feitas para garantir operações básicas como inserir, remover buscar em tempo constante independente do tamanho da estrutura.
- ▶ Para isso são assumidos alguns requisitos, como saber o tamanho total de elementos do conjunto e localizar uma função hash com espalhamento uniforme, em outras palavras poucas colisões.
- ▶ A primeira técnica de hash que estudaremos é o endereçamento direto. Usado quando o universo total de chaves é pequeno, não existem colisões entre duas chaves distintas e o tamanho da tabela de mapeamento direto é do tamanho do universo de chaves.

Hash

- ▶ Desenhar na lousa.
- ▶ Citar o exemplo de números: $D =$
(1, "um"), (2, "dois"), (3, "três"), (4, "quatro"), (5, "cinco"), (6, "seis")
.
- ▶ Citar o exemplo de palavras: Queremos construir um dicionário dinâmico simplificado, com 4 palavras de no máximo 8 letras. Este dicionário dinâmico pode ter somente as seguintes palavras: "concha", "casa", "hospital" e "time".

Hash

- ▶ No exemplo das palavras qual seria o tamanho do vetor de endereçamento?
- ▶ Vamos pensar...
- ▶ Número de diferentes String's de no máximo 8 letras é:
- ▶ O que seria inviável em termos de memória! O número de palavras armazenados é bem menor do que esse conjunto total, ou seja, seria gasto muita memória alocando um vetor grande sem necessidade.

Hash

- ▶ Vamos supor que temos em média 500 palavras, se o número de palavras do universo ultrapassar 500 teremos colisões.
- ▶ Todas as operações sobre a tabela hash devem ter complexidade assintótica constante $O(k)$ onde k é constante.
- ▶ Colisões sempre vão existir, como podemos lidar com elas?
- ▶ Discutir com os alunos!

Hash encadeamento

- ▶ Supondo que cada célula do vetor seja um objeto em C# que contenha uma chave, usada para indexar o hash e uma estrutura de dados lista ligada/árvore.
- ▶ Nesse caso sempre que ocorrer uma colisão teremos uma estrutura contendo os valores dentro.

Hash encadeamento

- ▶ Vamos supor uma função hash simples: $\langle\langle \text{Key} \rangle\rangle \bmod 7$, por exemplo.
- ▶ Em seguida, adicionamos as chaves: 50, 700, 76, 85, 92, 73, 101

Pilhas



Figura: Endereçamento aberto

Pilhas



Figura: Endereçamento aberto

Pilhas

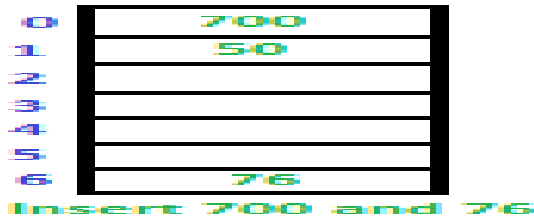


Figura: Endereçamento aberto

Pilhas

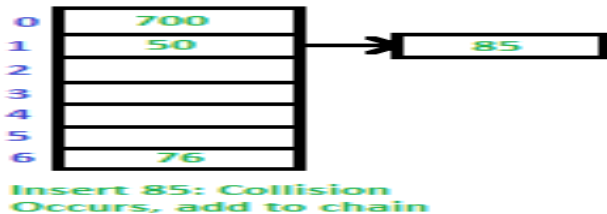


Figura: Endereçamento aberto

Pilhas

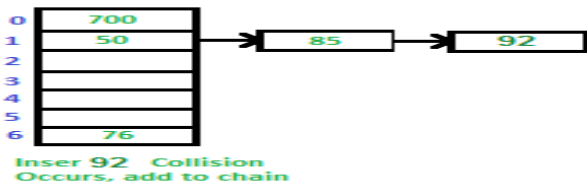


Figura: Endereçamento aberto

Pilhas

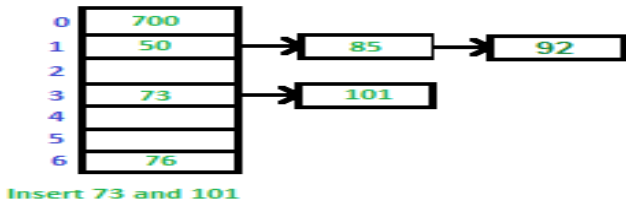


Figura: Endereçamento aberto

Hash encadeamento

- ▶ Quais seriam as vantagens e desvantagens dessa solução?
- ▶ Conversar com os alunos sobre isso.

Hash encadeamento

- ▶ Como vantagens temos:
- ▶ A simplicidade em implementar
- ▶ A tabela não fica vazia pois podemos encadear mais nós com colisão nas listas ligadas
- ▶ Usada frequentemente quando não temos certeza do número de elementos do conjunto

Hash encadeamento

- ▶ Como desvantagens temos:
- ▶ A performance do cache é inferior a abordagem de endereçamento aberto
- ▶ Desperdício de espaço, muitas células da tabela podem nunca ser usadas
- ▶ Se o encadeamento for longo, o tempo de pesquisa tenderá a $O(n)$ o que não é bom
- ▶ Usa espaço extra para armazenar os links

Hash endereçamento aberto

- ▶ Com essa abordagem usamos apenas o vetor para armazenar todas as chaves.
- ▶ Dessa forma, o tamanho da tabela tem que ser maior ou igual ao número de chaves totais do nosso universo.
- ▶ Evidente que podemos copiar os dados antigos para uma nova tabela, no caso de estourar o tamanho máximo. Mas dentro do possível queremos evitar essa situação.

- ▶ Uma primeira solução é a sondagem linear, ao ocorrer uma colisão procuramos no slot seguinte se há espaço vazio.
- ▶ No caso de ter espaço vazio, inserimos nele, caso contrário prosseguimos procurando mais slots vazios.

```
If slot  $\text{hash}(x) \% S$  is full, then we try  $(\text{hash}(x) + 1) \% S$   
If  $(\text{hash}(x) + 1) \% S$  is also full, then we try  $(\text{hash}(x) + 2) \% S$   
If  $(\text{hash}(x) + 2) \% S$  is also full, then we try  $(\text{hash}(x) + 3) \% S$   
.....
```

Figura: Sondagem Linear

- ▶ A desvantagem desse método é que após a inserção de diversas colisões o tempo de pesquisa passa a ser linear.

Hash: sondagem linear

- ▶ Vamos supor uma função hash simples: $\langle\langle \text{Key} \rangle\rangle \bmod 7$, por exemplo.
- ▶ Em seguida, adicionamos as chaves: 50, 700, 76, 85, 92, 73, 101

Pilhas



Figura: Endereçamento aberto

Pilhas



Figura: Endereçamento aberto

Pilhas



Figura: Endereçamento aberto

Pilhas

0	700
1	50
2	85
3	
4	
5	
6	76

Insert 85: Collision Occurs, insert 85 at next free slot.

Figura: Endereçamento aberto

Pilhas

0	700
1	50
2	85
3	92
4	
5	
6	76

Insert 92, collision occurs as 50 is there at index 1. Insert at next free slot

Figura: Endereçamento aberto

Pilhas

0	700
1	50
2	85
3	92
4	73
5	101
6	76

Insert 73 and 101

Figura: Endereçamento aberto

Hash: sondagem quadrática

- ▶ Uma outra técnica usada é a sondagem quadrática.
- ▶ Técnica que procura pela próxima posição vazia de célula elevada ao quadrado.

```
let hash(x) be the slot index computed using hash function.  
If slot  $\text{hash}(x) \% S$  is full, then we try  $(\text{hash}(x) + 1*1) \% S$   
If  $(\text{hash}(x) + 1*1) \% S$  is also full, then we try  $(\text{hash}(x) + 2*2) \% S$   
If  $(\text{hash}(x) + 2*2) \% S$  is also full, then we try  $(\text{hash}(x) + 3*3) \% S$ 
```

Figura: Endereçamento aberto quadrático

Hash: sondagem quadrática

- ▶ Exercício para os alunos aplicar o conceito de hash com sondagem quadrática para o exemplo anterior.

Hash: hash duplo

- ▶ Uma outra técnica usada é o hash duplo.
- ▶ Técnica que usa duas funções de hash para encontrar o próximo slot vazio.

let $\text{hash}(x)$ be the slot index computed using hash function.

If slot $\text{hash}(x) \% S$ is full, then we try $(\text{hash}(x) + 1 * \text{hash2}(x)) \% S$

If $(\text{hash}(x) + 1 * \text{hash2}(x)) \% S$ is also full, then we try $(\text{hash}(x) + 2 * \text{hash2}(x)) \% S$

If $(\text{hash}(x) + 2 * \text{hash2}(x)) \% S$ is also full, then we try $(\text{hash}(x) + 3 * \text{hash2}(x)) \% S$

Figura: Hash Duplo

Hash: sondagem quadrática

- ▶ Exercício para os alunos aplicarem o conceito de hash duplo para o exemplo anterior.

Hash: sondagem quadrática

- ▶ Usar a collection do C# de hashtable:
- ▶ [https://msdn.microsoft.com/pt-br/library/system.collections.hashtable\(v=vs.110\).aspx](https://msdn.microsoft.com/pt-br/library/system.collections.hashtable(v=vs.110).aspx).

Dúvidas...

Alguma dúvida?

Contato

- ▶ E-mail: *0800dirso@gmail.com* (alunos SENAC)
- ▶ E-mail: *adilson.khoury.usp@gmail.com*
- ▶ Phone: +55119444 – 26191
- ▶ [Linkedin](#)
- ▶ [Lattes](#)
- ▶ [GitHub](#)

- [1] A. V. Aho, J. E. Hopcroft, and J. Ullman, *Data Structures and Algorithms*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1983.
- [2] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change (2Nd Edition)*. Addison-Wesley Professional, 2004.
- [3] Beck, *Test Driven Development: By Example*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.
- [4] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley, 1999.
- [5] Geeks. (2018) A computer science portal for geeks. [Online]. Available: <https://www.geeksforgeeks.org>
- [6] Fernanda. (2014) Sql join: Entenda como funciona o retorno dos dados. [Online]. Available: <https://www.devmedia.com.br/sql-join-entenda-como-funciona-o-retorno-dos-dados/31006>
- [7] S. team. (2018) Sqlite sample database. [Online]. Available: <http://www.sqlitetutorial.net/sqlite-sample-database/>