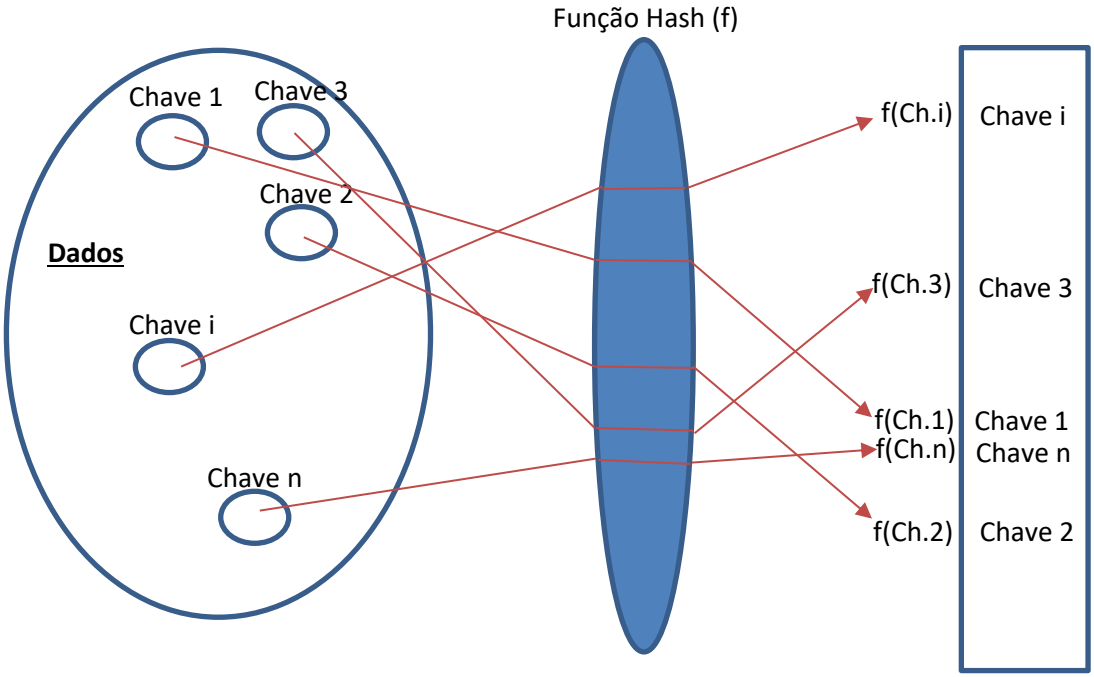
UNIVERSIDADE PRESBITERIANA MACKENZIE
- Faculdade de Computação e Informática –

Disciplina: Estrutura de dados II
Aula 10 – Profa Valéria Farinazzo e

Nome:	*TIA:*

Tabela Hash

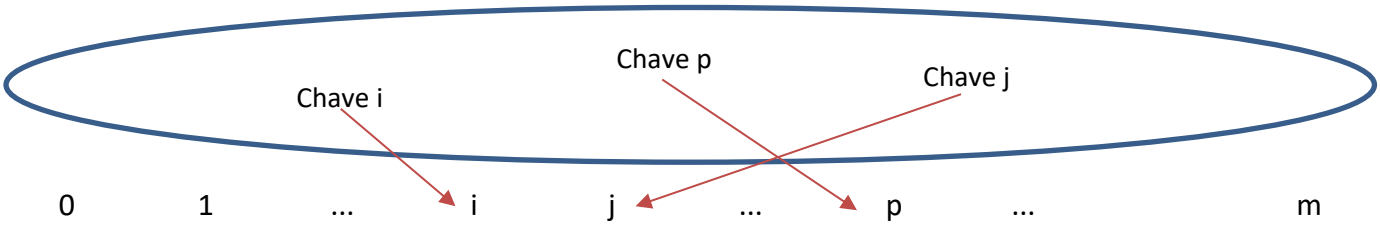
Hash é um método de armazenamento e busca que procura identificar imediatamente uma posição em um vetor para armazenar um elemento ou resgatar um elemento armazenado. O método funciona da seguinte forma: Dado um conjunto de dados formado por vários registros onde cada registro possui uma chave de armazenamento. A chave é então utilizada em uma função (função de Hash) para determinar a posição em um vetor onde o registro deve ser armazenado ou resgatado. De forma bem genérica a figura abaixo mostra o processo de Hash:



Existem várias formas de se implementar o método de Hash, estas variações estão associadas com o tipo de função Hash utilizada. A seguir vamos analisar algumas destas possibilidades:

Tabelas de Endereçamento Direto

Imagine que você possui um conjunto de dados com **n** chaves distintas. Agora assuma que os valores destas chaves estão em um intervalo entre [0, ..., m]. Se m > n, então é possível utilizar o método de endereçamento direto, onde o elemento que possui a chave j é armazenado na posição j do vetor.



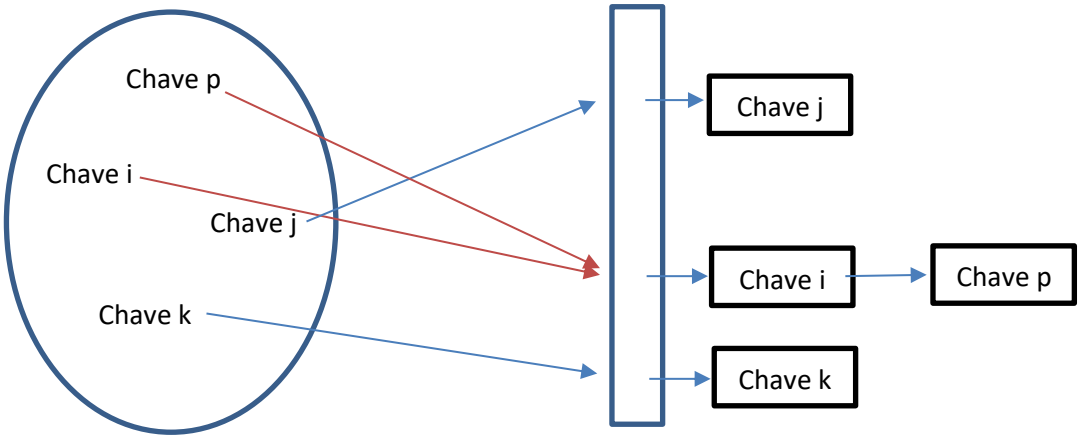
			Chave i	Chave j		Chave p			

Questão 1: O que acontece se o valor de m for muito maior que o valor de n? Imagine m = 10000 e n = 64.

Questão 2: Porque as chaves devem ser distintas? O que acontece se elas não forem distintas?

Os problemas levantados nas questões acima mostram que o método de endereçamento direto não é eficiente em vários casos práticos e precisamos então encontrar uma função hash para fazer o mapeamento entre as chaves existentes e as posições disponíveis no vetor. Note, porém, que encontrar uma função que mapeie cada chave para uma posição única em um vetor não é uma tarefa simples. Em muitos casos práticos pode-se ter mais de uma chave sendo mapeada para uma mesma posição do vetor, neste caso tem-se uma **colisão** de chaves. Existem algumas formas de lidar com colisões em tabelas Hash:

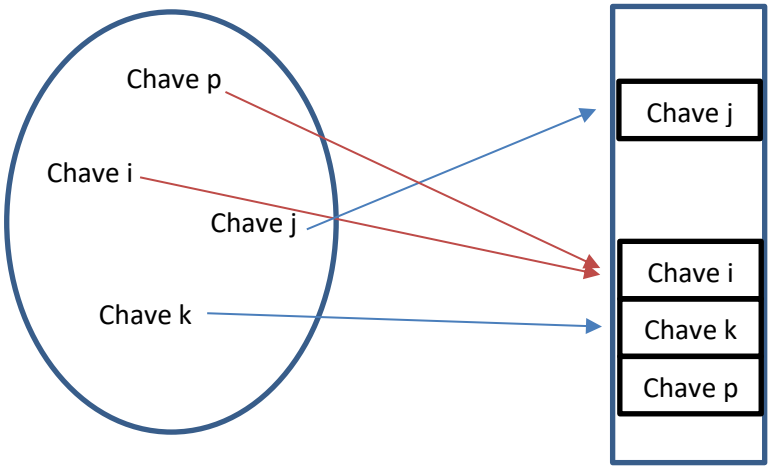
- Encadeamento: neste caso o vetor possui, para cada posição do vetor, uma lista encadeada para armazenar os elementos que colidirem naquela posição.



Questão 3: Sabendo que a colisão é praticamente inevitável no processo de Hashing, o que devemos tentar obter de uma função de Hash para utilizarmos o processo de encadeamento para resolver colisões?

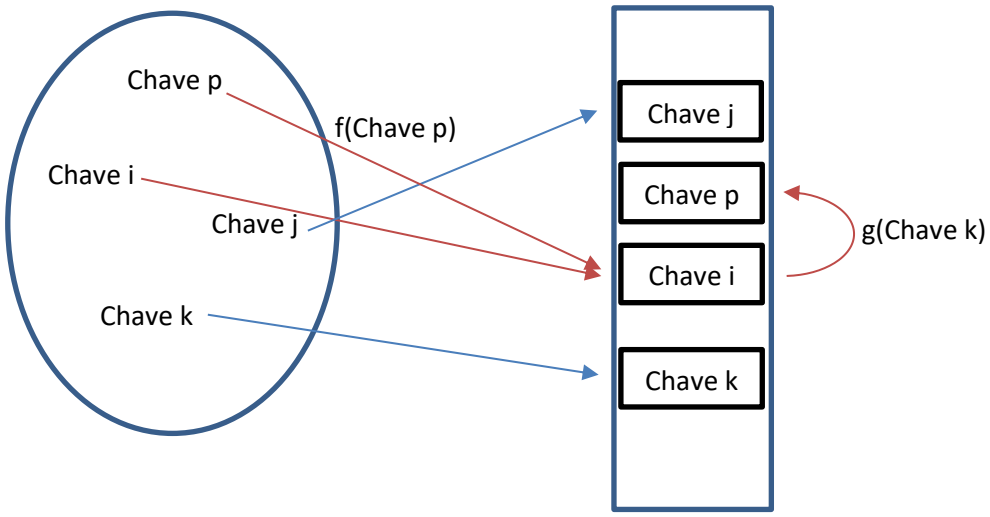
Questão 4: Qual seria o pior caso de uma operação de inserção ou de busca utilizando o Encadeamento num processo de Hash?

- Probing Linear: no processo de probing linear utiliza-se a própria tabela para se resolver os problemas de colisão. Ao se determinar a posição de armazenamento no vetor, caso exista colisão, verifica-se as próximas posições no vetor, linearmente, até que se encontre uma posição livre. O registro é então armazenado nesta posição.



Questão 5: Como é feito o processo de busca no método de Probing Linear?

- Hash duplo: o caso de hashing duplo é similar ao de probing linear, porém, antes de se procurar linearmente uma posição livre após a colisão, utiliza-se antes uma segunda função de hash para determinar uma outra posição para o elemento da colisão. Caso ocorra uma nova colisão utiliza-se então o probing linear.



É possível ainda ter uma segunda tabela onde a função g mapeia os elementos que colidem na primeira tabela para uma posição na segunda tabela. Neste caso, um terceiro elemento que colida com o primeiro na tabela 1 e com o segundo na tabela 2 é inserido utilizando Probing linear na tabela 2.

Questão 6: Como é feito o processo de busca no método de Hashing duplo?

Questão 7: Mostre um exemplo onde existam 3 colisões para chaves distintas e como seria o processo de inserção e busca neste caso.

Funcões Hash:

Uma função Hash deve ter uma propriedade simples: minimizar o número de colisões na Tabela Hash. Para que isto aconteça alguns tipos de funções e algumas propriedades da tabela devem ser seguidas.

Método da Divisão: a função hash neste caso é dada por: f(i) = i mod m, onde m é o tamanho da tabela, algumas características de m são:

- m > n (onde n é o número de elementos a ser inserido na tabela.
- m é um número primo.
- m pode ser próximo a uma potência de 2.

Método da multiplicação: neste caso a função f é dada por: f(i) = floor(m (i*c – floor(i*c))) onde c é um valor entre 0 e 1. Neste caso o valor de m não é crítico, porém valores de m como potências de 2 tornam a implementação mais eficiente.

Exercícios para o laboratório:

Questão 8: Utilizando o conceito de Probing Linear, implemente uma classe para tabelas hash. A classe deve criar uma tabela de tamanho m e utilizar uma função hash baseada em divisão. Defina uma quantidade de dados a ser inserida na tabela e crie métodos para inserir e buscar os elementos na tabela. A função de busca deve retornar VERDADE se o elemento estiver na tabela ou FALSO caso contrário. Crie um programa main que insere valores na tabela e mostre quantas colisões ocorrem no processo de inserção.

Questão 9: Utilizando o conceito de Encadeamento, implemente uma classe para tabelas hash. A classe deve criar uma tabela de tamanho m e utilizar uma função hash baseada em divisão. Defina uma quantidade de dados a ser inserida na tabela e crie métodos para inserir e buscar os elementos na tabela. A função de busca deve retornar VERDADE se o elemento estiver na tabela ou FALSO caso contrário. Crie um programa main que insere valores na tabela e mostre quantas colisões ocorrem no processo de inserção.