Algoritmos e Estrut. de Dados
Lab 11 - Tabelas Hash

Prof. Dr. Paulo César Rodacki Gomes
IFC - Instituto Federal Catarinense - Campus Blumenau

21 de junho de 2022

1 Objetivo

O objetivo desta atividade prática em laboratório é implementar uma estrutura de tabela de disperção (*hash table*) para armazenar cadastros de alunos utilizando **estratégia de listas encadeadas** para tratamento de colisão na tabela, de acordo com a representação esquemática da figura 1:

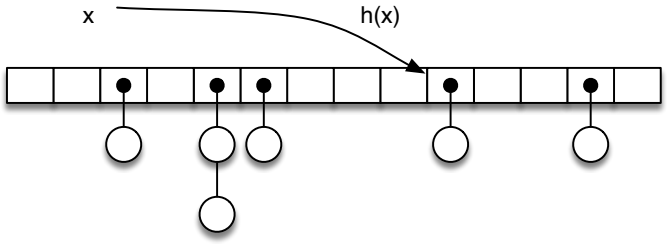

Figura 1: Representação da estratégia de tratamento de colisão

O cadastro de um aluno é armazenado em um elemento de uma lista encadeada, e a chave de busca é seu numero de matrícula. A tabela é um *array* de ponteiros para o primeiro elemento de cada lista. **OBS.:** os slides com a matéria de *hash table* estão no Google Classroom.

Abaixo, temos a estrutura de dados q ue deve ser implementada para o cadastro de alunos:

```
struct aluno {
   int matricula;
   char nome[81];
   char turma;
   char email[41];
   struct aluno* prox;
};

typedef struct aluno Aluno;
```

Podemos definir a tabela hash como um **tipo de dados** composto por um array de ponteiros para alunos, com uma quantidade de posições que seja um número primo, por exemplo:

```
#define N 127
typedef Aluno* Hash[N];
```

Desta forma, quando declaramos uma variável do tipo `Hash`, já estamos criando o array com quantidade prima de posições.

2 Descrição das funções a serem implementadas

Todas as definições e funções (incluindo a definição do tipo de dados Aluno) podem ser declaradas e implementadas nos arquivos `hash.h` e `hash.c`, respectivamente. A função main, testando e demonstrando o funcionamento de sua tabela hash deve ser implementada em um arquivo `principal.c`.

1. `static int hash(int k)`: função de dispersão. Calcula um índice na tabela, a partir da chave k. No caso deste exercício, k é o número de matrícula de um aluno. A função deve ser declarada estática para que seu esc opo seja local (restrito ao módulo `hash.c`);

2. `Aluno* hsh_get(Hash tab, int mat)`: operação para buscar e retornar a referência para um Aluno com número de matrícula `mat`. Se o aluno não for encontrado na tabela `tab`, retorna `null`.

3. `Aluno* hsh_set(Hash tab, int mat, char* n, char *e, char t)`: insere os dados de um novo aluno na tabela `tab`. Se já existir o cadastro do aluno na tabela, altere os dados inserindo os valores passados para a função. A função deve retornar o ponteiro para o novo aluno inserido, ou para o aluno com dados modificados;

4. `void hsh_remove(Hash tab, int mat)`: remove da tabela `tab` o registro do aluno com matrícula `mat`. Caso este aluno não seja encontrado, a função não faz qualquer alteração na tabela;

5. `void hsh_imprime(Hash tab)`:imprime o conteúdo de toda a tabela hash.