Implementação de hash com sondagem linear

Para este trabalho, considere uma tabela *hash* implementada com o mecanismo de sondagem linear para o gerenciamento das colisões. A tabela armazena cadeias de caracteres (*strings*). A função *hash* utilizada é baseada no número de caracteres da *string*.

1 Sondagem linear

A sondagem linear é uma estratégia para lidar com as colisões em um tabela hash. O algoritmo que deve ser utilizado no trabalho pode ser resumido da seguinte forma. Considera uma tabela T armazenada em um vetor v.

- 1. Seja s a string a ser adicionada na tabela.
- 2. Encontre a posição, pos, na qual s deve ser inserida em T.
- 3. Adicione s na posição pos. Isto é, v[pos] = s.

A Subseção 1.1 detalha a operação 2.

1.1 Encontrando a posição

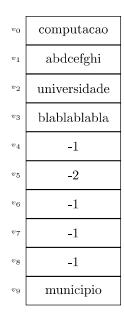
A posição de inserção é baseada no número de caracteres da *string* a ser inserida. Seja m o número de posições existentes no vetor $v = [v_0, v_1, \dots, v_{m-1}]$ e seja len(s) uma função que retorna o número de caracteres da *string* s. Então, a primeira tentativa de inserção é feita na posição hash(s), dada pela equação (1).

$$hash(s) = len(s) \% m. \tag{1}$$

Caso a posição hash(s) já esteja ocupada (houve uma colisão), então busca-se uma posição vaga a partir de hash(s), de forma circular, no vetor. A inserção ocorre na primeira posição vaga.

1.2 Exemplo de operações

No exemplo da Figura 1, é utilizado um vetor com 10 posições $[v_0, v_1, \ldots, v_9]$ para armazenar a tabela. O vetor está inicialmente vazio (indicado pelo valor -1) e a figura mostra o estado final do vetor após algumas operações de inserção e remoção. O valor -2 indica que posição que foi ocupada em algum momento, mas está atualmente vazia.



Operações realizadas:

- 1. Inserção de Teste. hash(Teste) = 5.
- 2. Inserção de municipio. hash(municipio) = 9.
- 3. Inserção de universidade. hash(universidade) = 2.
- 4. Inserção de computação. hash(computação) = 0.
- 5. Inserção de abdcefghi. hash(abdcefghi) = 9.
- 6. Inserção de abc. hash(abc) = 3.
- 7. Remoção de abc. hash(abc) = 3.
- 8. Inserção de blablablabla. hash(blablablabla) = 2.
- 9. Remoção de Teste. hash(Teste) = 5.

Figura 1: Exemplo de estado de vetor após uma sequência de operações.

2 Tarefas

Para esse trabalho você deverá desenvolver um programa que:

- 1. Lê o tamanho de um vetor que armazenará uma tabela hash.
- 2. Lê uma sequência de operações de inserção e remoção a serem feitas na tabela hash.
- 3. Implementa as operações de inserção e remoção em uma tabela *hash* utilizando a estratégia de sondagem linear para a solução de conflitos, de acordo com a Seção 1.1.
- 4. Imprime o estado final da tabela para o usuário. A impressão deve indicar os elementos presentes, as entradas da tabela que não foram utilizados (-1) e os espaços que foram utilizadas mas estão atualmente vazios (-2).
- 5. A leitura de dados e a impressão devem seguir o formato especificado na Seção 3.
- Submeter o código desenvolvido no Moodle-VPL na data de entrega definida (vide Moodle).

Além disso, a operação de remoção deve ignorar remoções de elementos que não estão na tabela e a inserção não deve ser realizada caso a tabela esteja cheia.

3 Formato de entrada e saída de dados

A primeira linha contém um inteiro com o tamanho da tabela hash, isto é, o número máximo de elementos que a tabela comporta. Seguem-se, então, 2n linhas, n > 1, correspondentes a n comandos. Cada comando é composto por duas linhas: a primeira com 0 ou 1 indicando respectivamente a adição ou remoção de uma string na tabela; e a segunda com a string a ser inserida (ou removida) de acordo com o comando (0 ou 1). Uma linha contendo -1 indica o final da entrada.

Considere o exemplo de entrada a seguir. A tabela tem tamanho 10 e são adicionadas respectivamente as strings: Teste, Ideia, universidade, computação, abdc, abc. Então, é removida a string abc e adicionada blablablabla.

Entrada:

```
1 10
2 0
3 Teste
4 0
5 Ideia
6 0
7 universidade
9 computação
10 0
11 abdc
12 0
13 abc
14 1
15 abc
16 0
17 blablablabla
```

A saída esperada é o estado do vetor $[v_0, v_1, \ldots, v_{m-1}]$ que representa a tabela hash, um elemento por linha, respectivamente. Posições que nunca foram ocupadas devem ser marcadas com -1 e posição que foram ocupadas em alguma momento, mas estão vazias, devem ser marcadas com -2. Portanto, para o exemplo fornecido, tem-se a saída a seguir.

Saída esperada:

```
computacao
comput
```