Algoritmos e Estruturas de Dados 3
Trabalho 2 - Cuckoo Hash

Jorge Lucas Vicilli Jabczenski & Vinicius Tikara Venturi Date

23 de Fevereiro 2021

Estrutura Célula e Tabelas

Cada tabela é formada por estruturas do tipo célula, as quais têm: a chave, e duas flags - uma que indica se está vazia e outra se foi excluída. Desta forma uma Tabela é formada por um vetor de tamanho M do tipo célula.

```
typedef struct celula{
    int num;
    int excluido;
    int vazio;
} celula;

celula T1[M], T2[M];
```

Inclusão

Se a posição em T1 estiver marcada como vazia ou excluída (utilizando-se das flags apresentadas anteriormente), então a chave k é inserida em T1 com a função $h1(k, m)$ determinando sua posição. Caso a posição esteja ocupada, copiamos a chave de T1 para T2 utilizando a função $h2(k, m)$ para determinar a posição nova, e inserimos o novo valor em T1 igual mostrado anteriormente.

```
pos1 = h1(n,M);

/* Como nao pode haver chaves repetidas */
if (T1[pos1].num != n)
{
    if (T1[pos1].vazio == 1 || (T1[pos1].excluido == 1))
    {
        T1[pos1].vazio    = 0;
        T1[pos1].excluido = 0;
    } else {
        int novaPos = h2(T1[pos1].num, M);

        T2[novaPos].num   = T1[pos1].num;
        T2[novaPos].vazio = 0;
    }
    T1[pos1].num = n;
}
```

Exclusão

Primeiro, a chave é procurada em T2, e, caso exista, a célula é simplesmente marcada como vazia. Se a chave não existir em T2, procuramos ela em T1, e se for encontrada, a célula é marcada como excluída, ao invés de vazia, pois podem existir chaves em T2 que dependam desta chave em T1, como descrito no algoritmo de inclusão.

```
pos1 = h1(n,M);
pos2 = h2(n,M);

// Trata tabela 2 antes
if(T2[pos2].num == n && !T2[pos2].vazio){
    T2[pos2].vazio = 1;
// Entao trata a tabela 1
} else if(T1[pos1].num == n){
    T1[pos1].excluido = 1;
}
```

Impressão

Primeiro, as duas tabelas são percorridas e todos os valores válidos (posições não marcadas como vazias ou excluídas) são armazenadas em uma matriz auxiliar de tamanho $2M$ por 3 (pois as duas tabelas podem estar cheias). Nessa matriz são guardadas 3 informações: o valor da chave, a tabela onde ela se encontra e a posição nessa tabela.

Por fim, essa matriz é ordenada usando o *Insertion Sort* utilizando o valor da chave como parâmetro, deixando a saída como requisitado na especificação.

Utilizando a entrada de *teste 2* como exemplo:

Tabela 1 de tamanho M:1

		24						19		

Tabela 2 de tamanho M:1

							13			

Tabela de impressão de tamanho 2M:3

24	13	19																			
1	2	1																			
2	7	8																			

Tabela de impressão após Insertion Sort

13	19	24																			
2	1	1																			
7	8	2																			

Saída:

```
13,T2,7
19,T1,24
24,T1,2
```