Resolução da prova 2 de AED-II

> Passada pelo professor Helton em 2012

Q1.

São três modalidades de função hash,

1. **Método da divisão:** O índice é o resto da divisão da chave pelo tamanho da tabela.

```
int hash(int key, int size) {
    return key % size;
}
```

2. **Método do quadrado médio:** Eleva-se a chave ao quadrado, seleciona-se os dígitos ao centro desta enquanto índice para colocação da chave nesta tabela. O número de dígitos selecionados é dado pelo tamanho da tabela.

```
int digits(int number) {
    return (number > 0) ? countDigits(number / 10) + 1 : 1;
}

int hash(int key, int size) {
    int index = key * key;
    if (index < size)
        return index;
    return index / (5 * (digits(key) - digits(size))) % size;
}
```

3. **Método da dobra:** Subdivide-se a chave em sua representação binária em subconjuntos de bits suficientes para descrever um índice da tabela e aplica-se a operação XOR entre estes subconjuntos. O resultado desta operação é o índice em que a chave há de ser colocada na tabela.

```c
int hash(int key, int size) {
    int i, digits = 0, bits = 8 * sizeof(unsigned int);
    unsigned int index, mask = 0;

    for (i = 1; i < size; i *= 2) {
        mask <<= 1;
        mask += 1;
        digits++;
    }

    for (i = index = 0; i < bits; i += digits) {
        index ^= (key & mask) >> i;
        mask <<= digits;
    }

    return (int) index;
}
```

Q2.

```c
#include <limits.h>
#define EMPTY INT_MIN
#define REMOVED INT_MAX

bool removeKey(int key, HashMap *h) {
    int i = 0, index = hash(key);

    while (h->keys[index] > key && i < h->entries)
        index = rehash(i++, key);
    if (h->keys[index] != key)
        return false;
    h->keys[index] = REMOVED;
    return true;
}
```

Q3.

Os algoritmos de hash extensível e linear tratam-se de métodos para gerenciar **dinamicamente** uma tabela de hash [1]. Ou seja, ajustando o tamanho desta de acordo com o número de registros que esta deverá comportar sem que, para isso, uma operação de *rehash* necessite ser aplicada sobre todas as chaves armazenadas na tabela (tal qual em tabelas estáticas). Ambos os algoritmos anteriormente mencionados

- utilizam uma estrutura de "baldes" para armazenar conjuntos de chaves os quais possuem uma mesma quantidade de dígitos menos significativos iguais.
- dividem o conteúdo destes baldes em dois assim que estes atingem uma dada capacidade máxima. O parâmetro para a divisão do conteúdo entre estes baldes passa a incorporar um bit menos significativo adicional.

Estes algoritmos diferem entre si principalmente pela maneira como decidem realizar a divisão dos baldes:

- O hash extensível divide os baldes assim que adicionar uma chave a estes provocaria um *overflow*. Assim, o intervalo em que a divisão dos baldes ocorre, quais baldes são divididos, e quando se dá o crescimento da tabela é *imprevisível*, pois é dependente do conjunto de chaves de entrada.
- O hash linear, por outro lado, divide os baldes sempre que uma dada quantidade de chaves é acrescentada à tabela e tem seus baldes divididos de forma pré-estabelecida. Para isso, entretanto, este faz uso de baldes adicionais para comportar chaves excedentes até que as condições para divisão dos baldes seja satisfeita.

Q4.

Primeiramente, é necessário explicitar que o limite $t-1$ (onde t é o grau da árvore) para o número mínimo de elementos por nó se aplica a todos os nós **senão** o nó na raiz da árvore, que tem no mínimo um elemento. Assim sendo, uma árvore de de altura h tem 1 nó na profundidade 0 (a raiz), no mínimo 2 nós na profundidade 1 (os dois filhos da raiz de um único elemento), no mínimo 2t nós na profundidade 2, no mínimo $2t^2$ nós na profundidade 3, e assim por diante, até que na profundidade h há no mínimo $2t^{h-1}$ nós. Portanto, o número c de chaves contidas desta árvore é

$$c \geq 1 + (t-1)\sum_{i=1}^{h} 2t^{i-1} \implies c \geq 1 + 2(t-1)\sum_{i=1}^{h} t^{i-1}$$

seja $S_n = \sum_{i=1}^{h} t^{i-1} = t^0 + t^1 + t^2 + \cdots + t^{h-1}$, então

$$t \cdot S_c = t + t^2 + \cdots + t^{h-1} = S_c + (t^h - 1) \implies t \cdot S_c = S_c + (t^h - 1)$$
$$\implies S_c(t-1) = t^h - 1 \implies S_c = \frac{t^h - 1}{t-1}$$

Logo,

$$c \geq 1 + 2(t-1)\left(\frac{t^h - 1}{t-1}\right) \implies c \geq 2t^h - 1 \implies t^h \leq \frac{c+1}{2}$$

$$\implies \log_t t^h = \log_t \left(\frac{c+1}{2}\right) \implies h \leq \log_t \left(\frac{c+1}{2}\right) \blacksquare$$

Q5.

O que é uma lista generalizada?

---

1. Fonte: Linear Hashing - Data Structures - YouTube ↩