

Hashing

- Hashing já sabe em que posição está
- Complexidade = $O(1)$ (Se tiver muitas colisões a complexidade aumenta)
- É uma função que mapeia a posição do elemento
- Fazer busca + rápido, a complexidade nem sempre é $O(1)$ → temos a $O(n)$
- Mapeia uma chave

Método da divisão

- Chave K
- Tem um objeto qualquer e um algoritmo pra gerar uma chave (K)
- = Objeto → Chave → hash → posição
 - a partir da chave gera a posição
- m = Tamanho do vetor
- $h(K) = K \bmod m$ ($\bmod = \%$)
- Se $K = 100$ e $m = 12$ $h(K) | 100 \bmod 12 = 4$ (Resto da divisão)
- Se $h(h)$ retornar um número negativo, somar m na h (usa um if pra isso)

Problema

- Quando duas chaves querem armazenar na mesma posição temos um problema chamado **colisão**

Como resolver?

- Usando endereçamento aberto
- Quando há colisão, o elemento deve ser armazenado em uma outra posição da tabela
- Se um lugar estiver ocupado, então deve-se procurar por uma segunda alternativa. Esse processo se chama **SONDAGEM**, existe sondagem linear e quadrática e double hash

Endereçamento linear

Introdução à hash

$$h'(k) = k \bmod m \quad k = \text{chave}$$

$$h(k, i, m) = h'(k) + i \bmod m$$

↳ Quando tiver posições iguais

i = número de tentativas, sempre começa em 0

Exemplo: Inserir as chaves 4, 15 e 28 (8 posições)

$$h(4) = 4 \bmod 8 = 4$$

$$h(15) = 15 \bmod 8 = 7$$

$$h(28) = 28 \bmod 8 = 4 \quad (\text{Colisão com a chave } 4)$$

Então tentar de outro jeito:

$$h(28, 1, 8) = (h'(28) + 1) \bmod 8$$

$$h(28, 1, 8) = (4 + 1) \bmod 8 = 5$$

• Esse método não é usado na prática

Exemplo 2) Determine as 4 primeiras posições em que o elemento 37 poderia ser armazenado em uma tabela de $m = 11$ posições, ao usar a sondagem linear, com $h(k) = k \bmod m$

Formulas:

$$\rightarrow h'(k) = k \% m$$

$$\rightarrow h(k, i, m) = (h'(k) + i) \% m$$

$$1^{\circ} \text{ posição}) \quad h'(37) = 37 \bmod 11 = 4 \quad \checkmark$$

$$2^{\circ} \text{ posição}) \quad h(37, 1, 11) = (h'(37) + 1) \bmod 11$$

$$h(37, 1, 11) = (4 + 1) \bmod 11 = 5 \quad \checkmark$$

$$3^{\circ} \text{ posição}) \quad h(37, 2, 11) = (4 + 2) \bmod 11 = 6 \quad \checkmark$$

$$4^{\circ} \text{ posição}) \quad h(37, 3, 11) = (4 + 3) \bmod 11 = 7 \quad \checkmark$$

i começa no 0

→ Cabeçalho da função hash:

```
int hash1 (int k, int i, int m)
int hash_aux (int k, int m)
```

- O cabeçalho é padrão

Algoritmo de inserção

↳ Para fazer a sondagem linear no código deve ter a função hash variável chamar ela ($h'(k)$)

- Tem struct pro meio
- Struct dado {
 - int k;
 - int status;
}
- struct que armazena a chave e o status (ocupada, vazia ou removida)
- i = 0
- j = h(k, i, m)
- j vai ser a posição no vetor (retorna a posição)

Pesquisa hashing

- Hash - search (t, k, m)
- Retorna a posição da chave buscada na tabela (j)
- Se não tiver a chave retorna -1
- Têm que fazer tudo (os hashs e inserção na tabela)

Remoção em tabelas hash

- Usa os hash e insert
- A busca tem que continuar após a remoção por isso muda o status
- Vai mexer no status (0: Vazio, 1: Ocupado, 2: Removido)
- Vai mudar o status por causa do while da busca
- Status 0 ou 2: Pode inserir uma chave nessas posições

Como remover?

- 1º) Encontrar (busca a chave que deseja remover)
 - 2º) Mudar o valor de k para -1 e de status para 2
Se não encontrar k retorna -1
- Algoritmo int hash-remove (t, m, k)

Busca quadrática (Função quadrática) Estudar o código ▲▲

$$\cdot h(k, i, m) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

Mas em casos de colisão também

- C_1 e C_2 tem valores definidos no enunciado ou é digitado (bacharelado)
- cabeçalhos: int hash2(int k, int i, int m, int c1, int c2)
$$h(k, i) = (h'(k) + c1 \cdot i + c2 \cdot i^2) \bmod m$$

Double hashing

- promove um espalhamento mais uniforme
- chama double hashing porque usa 2 funções auxiliares
- $h_1(k, m) = k \bmod m$
- $h_2(k, m) = 1 + (k \bmod (m-1))$
- $h(k, i, m) = (h_1(k) + i h_2(k)) \bmod m$

Prova Ynoguti Pandemio

Questão de hash: Foi uma questão de sondagem quadrática com inserção na tabela.

Enunciado

Faça um programa que leia várias chaves e as insira em uma tabela hash de tamanho m , usando a técnica de sondagem quadrática, dada por:

$$h'(k, m) = k \bmod m$$

$$h(k, i, m) = (h'(k, m) + c_1 \cdot i + c_2 \cdot i \cdot i) \bmod m$$

Lembre-se que quando $h'(k, m)$ resultar negativo, você deve somar m .

Faça uma função para cada uma das expressões acima, com os seguintes cabeçalhos (isto é obrigatório):

```
int hash_aux(int k, int m)
```

```
int hash1(int k, int i, int m, int c1, int c2)
```

onde k é a chave a ser inserida, m é o tamanho da tabela hash, i é o número da tentativa, e c_1 e c_2 são 2 inteiros positivos.

Depois faça um programa que leia várias chaves e as insira em uma tabela de tamanho m , usando a sondagem quadrática.

Entrada

A entrada consiste várias linhas:

- a primeira linha terá 3 números inteiros, que correspondem ao tamanho m da tabela, C_1 e C_2 , respectivamente;
- depois seguem várias linhas com as chaves a serem inseridas na tabela. A entrada de dados termina quando for inserido o número -1, que não deve ser inserido na tabela.

Saída

Na saída, o programa deve mostrar a tabela preenchida, com valores iguais a -1 nas posições vazias.

Exemplos de entrada

```
11 1 3  
2 7 5 11 25 -1
```

Exemplos de saída

```
11 -1 2 25 -1 5 -1 7 -1 -1 -1
```

A solução está salva como: provaoguti.cpp

•

