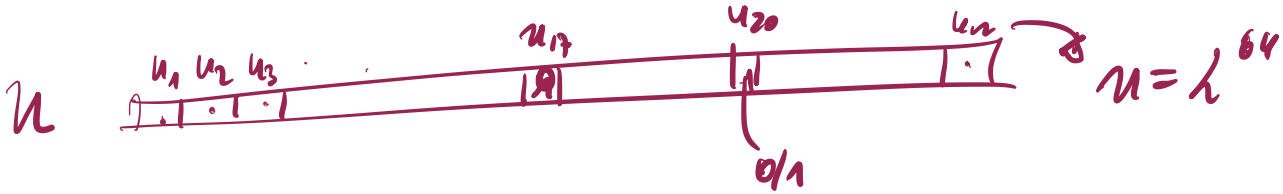

# HASHING 1

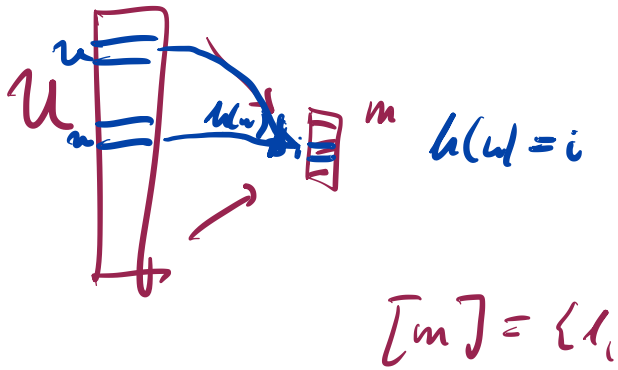
Marcus Ritt

# Outline


- *Universo* de chaves $|U|$, *tabela hash* $H$.
- Operações:
  - Inserção de uma chave $c \in U$: insert($c$,$H$)
  - Deleção de uma chave $c \in U$: delete($c$,$H$)
  - Teste da pertinência: Chave $c \in H$? lookup($c$,$H$)

O(1).





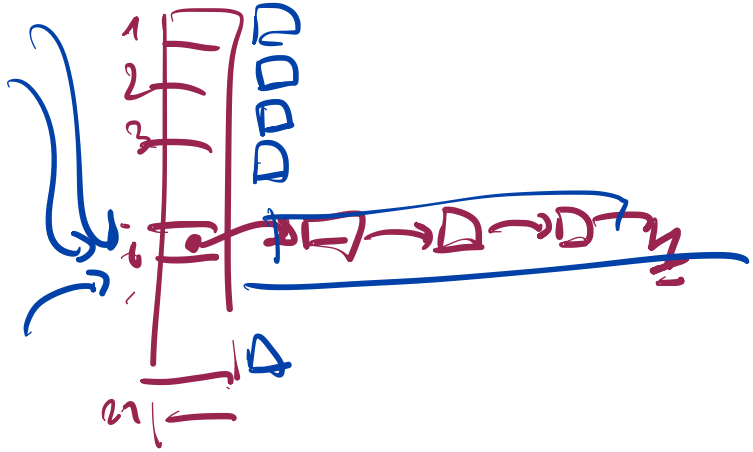

- Tabela de tamanho $m \ll |U|$
- Função hash: $h : U \to [m]$ para posições.
- Problema: *colisão* $h(u_1) = h(u_2)$

- Lista $l_i$ na posição $i$ com $n_i := |l_i|$ elementos

$insert(c,H) :=$
$\quad insert(c,l_{h(c)})$ $O(1)$

$lookup(c,H) :=$
$\quad lookup(c,l_{h(c)})$ $O(n_i)$

$delete(c,H) :=$
$\quad delete(c,l_{h(c)})$ $O(n_i)$



- Para função hash *simples* e *uniforme*:

$$\Pr(h(c) = i) = 1/m. \quad (1)$$

- Seja $c_{ji}$ a variável aleatória que indica se chave $j$ pertence a lista $i$
- Temos $\Pr(c_{ji} = 1) = \Pr(h(j) = i)$ e $n_i = \sum_{1 \le j \le n} c_{ji}$
- Logo

$$E[n_i] = E[\sum_{1 \le j \le n} c_{ji}] = \sum_{1 \le j \le n} E[c_{ji}] = \sum_{1 \le j \le n} \Pr(h(c_j) = i) = n/m.$$

- Onde $\alpha := n/m$ é o *fator de ocupação*.

**Teorema 2.1**
Uma busca sem sucesso precisa tempo esperado $\Theta(1+\alpha)$.

**Teorema 2.2**
Uma busca com sucesso precisa tempo esperado $\Theta(1+\alpha)$.

Demonstração Th. 2.1

- Chave c tem prob. $1/n$ valer lista i.
- Tempo esperado da busca: $\Theta(\alpha)$
- Tempo total: $\Theta(1+\alpha)$ ↑ hash

Th. 2.2 Hip.: chave $c$ é uma das chaves na tabela de prob. uniforme.

$\Rightarrow$ Prob. pertencer a lista $L_i$: $n_i/n$.

$\Rightarrow$ Custo: $\Theta(1 + E[p])$ ← posição na lista

Seja: $c_1, c_2, \ldots, c_n$ a ordem de inserção; assume: inserção é no início da lista.

$X_{ij} = \begin{cases} 1, & \text{chaves } c_i, c_j \text{ têm o mesmo valor hash} \\ 0, & \text{c.c.} \end{cases}$ $\quad E[X_{ij}] = 1/m$.

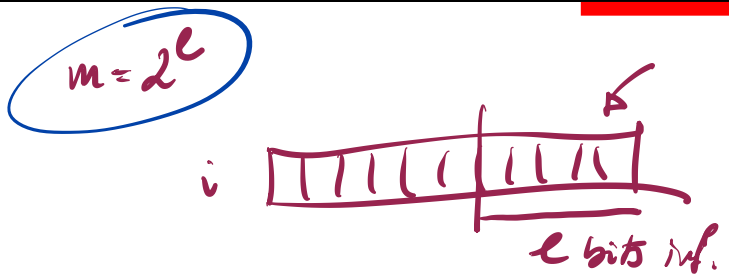
$p_i$ a posição da chave $c_i$ na sua lista.

$$E[p_i] = E\left[1 + \sum_{j:j>i} X_{ij}\right] = 1 + \sum_{j:j>i} E[X_{ij}] = 1 + (n-i)/m.$$

$$E[p] = \sum_{i \in [n]} \frac{1}{n} E[p_i] = \sum_{i \in [n]} \frac{1}{n}\left(1 + (n-i)/m\right) = 1 + \frac{n}{m} + (n+1)/2m$$

$$= 1 + \alpha/2 + \alpha/(2n).$$

$$\Theta(1 + E[p]) = \Theta(2 + \alpha/2 - \alpha/2n) = \Theta(1+\alpha)$$




Método de

- Divisão
  - $h(i) = i \mod m$ é simples e uniforme.
  - Prática: número primo "suficientemente" distante de uma potência de 2.
- Multiplicação

$$h(c) = \lfloor m \{Ac\} \rfloor .$$

$\{x\} = x - \lfloor x \rfloor$

  - Funciona para qualquer $m$, mas depende de um $A \in \mathbb{R}$ adequado.
  - Knuth: $A \approx (\sqrt{5} - 1)/2$.

- *Problema*: para função hash $h$ fixa, sempre existe um conjunto de chaves com muitas colisões
- Idéia: $\rightarrow$ escolhe uma *função hash aleatória* de uma família $\mathcal{H}$ de funções.
- Uma família é *universal* se

$$|\{h \in \mathcal{H} \mid h(c_1) = h(c_2)\}| = |\mathcal{H}|/m$$

ou equivalente

$$\Pr(h(c_1) = h(c_2)) = 1/m \qquad \text{sobre todos } h \in \mathcal{H}$$

para qualquer par de chaves $c_1, c_2$.

**Teorema 3.1**

Se escolhemos uma função hash $h \in \mathcal{H}$ uniformemente, para uma chave arbitrária $c$ o tamanho esperado de $l_{h(c)}$ é

- $\alpha$, caso $c \notin H$, e
- $1 + \alpha$, caso $c \in H$.

Dem. Fixa $c_1, c_2$; $X_{ij} = [h(c_1) = h(c_2)]$

$E[X_{ij}] = Pr[X_{ij} = 1] = Pr[h(c_1) = h(c_2)] = 1/m.$

Núm. de colisões de uma chave fora $c$, $Y_c$.

$$E[Y_c] = E\left[\sum_{\substack{c' \in H \\ c' \neq c}} X_{cc'}\right] = \sum_{\substack{c' \in H \\ c' \neq c}} E[X_{cc'}] \leq \sum_{\substack{c' \in H \\ c' \neq c}} 1/m.$$

$c \notin H$: tamanho $Y_c$ : tam. esp. $E[Y_c] \leq n/m = \alpha$

$c \in H$: tam. lista $Y_{c+1}$: $E[Y_{c+1}] = (n-1)/m$

$E[Y_{c+1}] = 1 + (n-1)/m = 1 + \alpha - 1/m < 1 + \alpha$

Exemplo: Chave $c = (c_0, c_1, \ldots, c_r)_m$ $c_i \in [0, m-1]$

← m base m

Escolhe $a = (a_0, a_1, \ldots a_r)_m$ aleatoriamente

$$h_a(c) = \left(\sum_{i \in [0,r]} a_i c_i\right) \bmod m.$$

↳ universal.

- *Idealmente*: sem colisões. $\rightarrow$ hashing perfeito.
- Garantia: somente com conhecimento das chaves.
- Função aleatória em $\mathcal{H}$: número esperado de colisões

$$E[\sum_{i \neq j} X_{ij}] = \sum_{i \neq j} E[X_{ij}] \leq n^2/m.$$

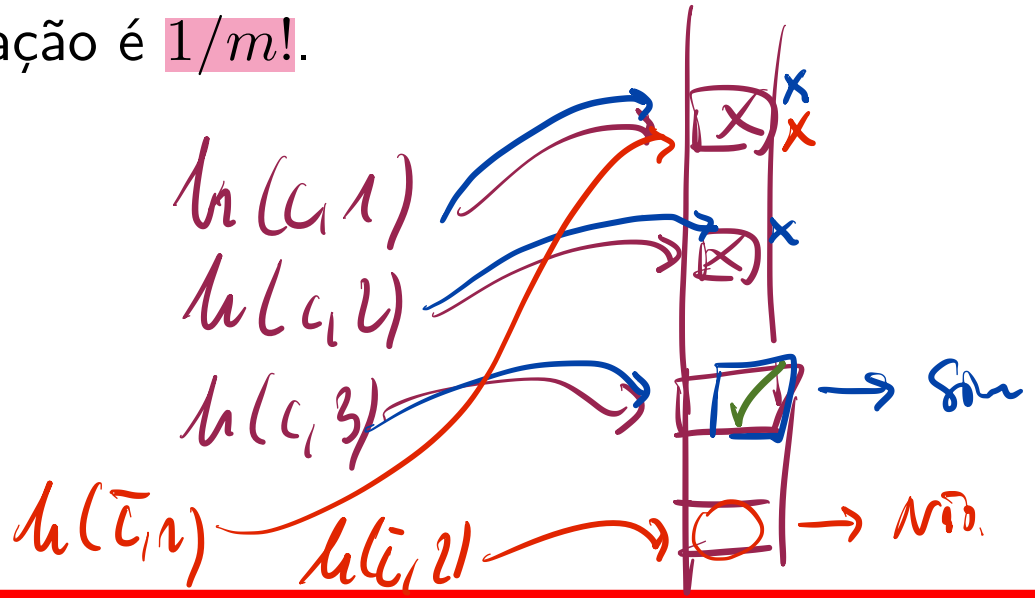
- Logo: número esperado de colisões para tabela de tamanho $m > n^2$ é $\leq 1$.
- Ainda: para $m > cn^2$ com $c > 1$ a probabilidade de uma colisão é

$$\Pr(\sum_{i \neq j} X_{ij} \geq 1) \leq E[\sum_{i \neq j} X_{ij}] \leq n^2/m < 1/c.$$

Pr[X ≥ a] ≤ E[X]/a. (Des. Markov)

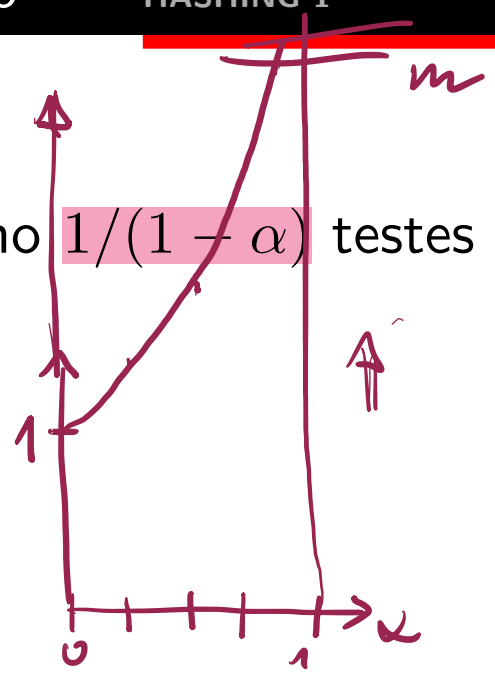
Open addressing.

- Colisão: $\rightarrow$ seleciona *outra posição*, visitando os elementos em alguma ordem $\pi$
- Concretamente: $h(c,1), \ldots, h(c,m)$ é uma *permutação* de $[m]$.
- Função $h(c,i)$ *uniforme*: probabilidade de uma chave aleatória gera dada permutação é $1/m!$.



```
insert(c,H) :=
    for i in [m]
        if H[h(c,i)] = free
            H[h(c,i)]=c
            return

lookup(c,H) :=
    for i in [m]
        if H[h(c,i)] = free
            return false
        if H[h(c,i)] = c
            return true
    return false
```




**Teorema 4.1**

As funções lookup e insert precisam no máximo $1/(1-\alpha)$ testes caso a chave não está na tabela.

Dem. $X$: núm. de testes até encontrar uma pos. livre.

$$E[X] = \sum_{i \geq 1} i \cdot Pr(X=i) = \sum_{i \geq 1} \sum_{j \geq i} Pr(X=i)$$

$$= \sum_{i \geq 1} Pr(X \geq i) \quad \|$$

$T_i$ : evento teste $i$ ocorre e a pos. $i$ ocupada.

$$Pr(X \geq i) = Pr(T_1 \cap T_2 \cap \dots \cap T_{i-1}) = Pr(T_1)\, Pr(T_2 | T_1)\, Pr(T_3 | T_1, T_2) \dots Pr(T_{i-1} | T_1, \dots, T_{i-2})$$

$Pr(T_1) = n/m = \alpha$ ; $Pr(T_2 | T_1) = (n-1)/(m-1)$ ;

$Pr(T_k | T_1 \dots, T_{k-1}) = (n-k+1)/(m-k+1) \leq n/m = \alpha$

$$Pr(X \geq i) \leq \alpha^{i-1}$$

$$E[X] = \sum_{i \geq 1} Pr(X \geq i) \leq \sum_{i \geq 1} \alpha^{i-1} = \sum_{i \geq 0} \alpha^{i}$$

$$= 1/(1-\alpha).$$

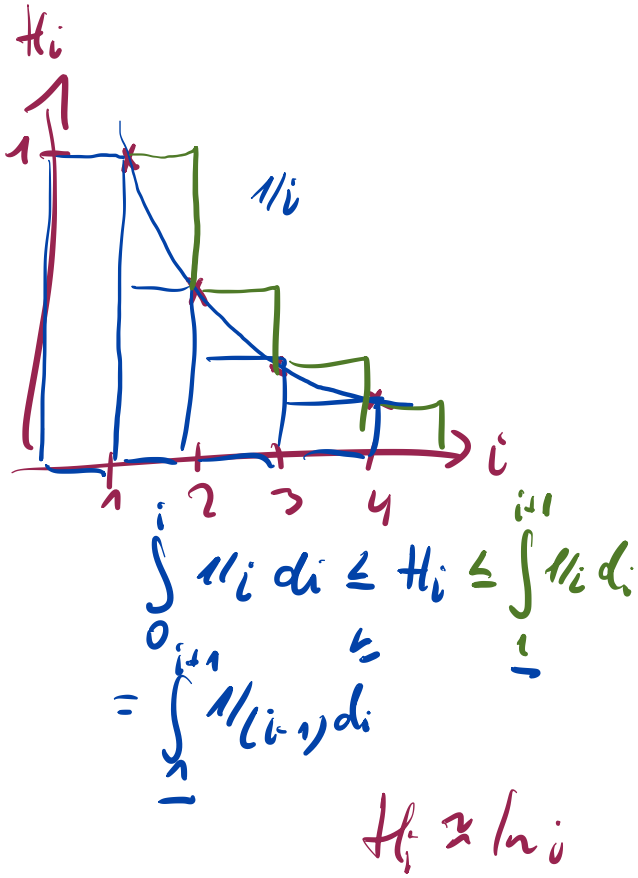
**Lema 4.1**

Para $i < j$, temos $H_i - H_j \leq \ln(i) - \ln(j)$.

$H_i = 1 + 1/2 + 1/3 + \dots + 1/i.$

$i < j : \quad H_i - H_j \leq \int_{i}^{i+1} \frac{1}{x-1}\,dx$

$= \int_{j}^{j+1} \frac{1}{x-1}\,dx = \ln(i) - \ln(j).$



**Teorema 4.2**

Caso $\alpha < 1$ a função lookup precisa esperadamente $1/\alpha \ln 1/(1-\alpha)$ testes caso a chave esteja na tabela, e cada chave tem a mesma probabilidade de ser procurada.

Dem. Chave $c_i$: $c$ foi a $i$-ésima chave inserida. $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$

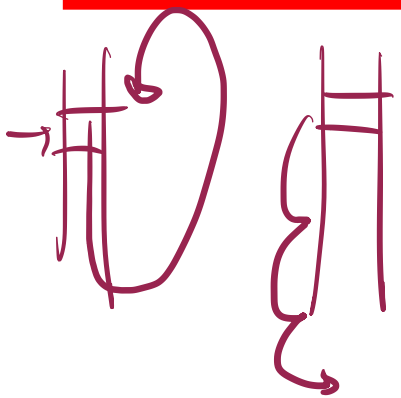
Na inserção: $i-1$ chaves; $\alpha = (i-1)/m$. $\Rightarrow$ # tedas esp. até uma posição livre.

$\hookrightarrow 1/(1-(i-1)/m) = m/(m-(i-1))$.

$$E[T] = \frac{1}{n} \sum_{i \in [n]} m/(m-(i-1)) = \frac{1}{\alpha} \sum_{i \in [0,n]} \frac{1}{m-i} = \frac{1}{\alpha} (H_m - H_{m-n})$$

$$< \frac{1}{\alpha} (\ln(m) - \ln(m-n)) = \frac{1}{\alpha} \ln \frac{1}{1-\alpha}.$$

# Funções hash para endereçamento aberto



- Linear: $h(c,i) = h(c) + i \mod m$
- Quadrática: $h(c,i) = h(c) + c_1 i + c_2 i^2 \mod m$
- Hashing duplo: $h(c,i) = h_1(c) + i h_2(c) \mod m$

(Nenhuma das funções é uniforme, mas o hashing duplo mostra um bom desempenho na prática.)