Tabela Espalhamento (Hash function)

Rafael Alves da Costa

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
FATEC Carapicuíba

Aula 8 - Estrutura de dados

09/2025

Sumário

Hashing

Hashing

- Quando tem-se uma **lista ordenada** pode-se melhorar uma busca nela utilizando-se uma busca binária que é em **tempo logarítmico**[1].
- Agora vamos construir uma **estrutura de dados** cuja busca terá **tempo constante**. Isto é, vamos aprender o conceito de hashing[1].
- Uma **tabela de dispersão** (ou **hash table**) é uma **coleção de itens** que são **armazenados** de maneira a serem **encontrados** com facilidade mais tarde. Cada **posição** da tabela de dispersão, geralmente **denominada índice** (ou **slot**), pode **guardar um item** e possui um **rótulo inteiro** começando a partir de **0**. [1]

Tabela de Dispersão com 11 Índices Vazios

IME-USP

[1] Miller, B. N.; Ranum, D. L. Problem solving with algorithms and data structures using python, 2Ed. Franklin, Beedle and Associates Inc., 2011.

Hashing

- O **mapeamento** entre a **chave** e o **índice** ao qual ela pertence na tabela é conhecido como a **função de espalhamento** (ou **hash function**)[1].
- A função de espalhamento receber **qualquer item na coleção** e irá retornar um **inteiro dentro do intervalo dos índices**, isto é, entre 0 e m-1.[1]
- Suponha, o conjunto de inteiros formado por **54, 26, 93, 17, 77 e 31**.
- Função de espalhamento do exemplo mostrado na tabela:

$$h(item) = item\%11$$

Item	Valor de Espalhamento
54	10
26	4
93	5
17	6
77	0
31	9

Função Simples de Espalhamento Usando o Resto da Divisão
IME-USP

obs:. Observe que o método do resto (artimética modular) tipicamente estará presente de algum modo em todas as funções de espalhamento[1].

Hashing

- Uma vez que os valores de espalhamento tenham sido computados, podemos inserir cada item na tabela de dispersão na posição designada[1].
- Nota-se que 6 dos 11 índices estão agora ocupados. Chama-se essa razão como **fator de carga (λ)** e é comumente denotada por $\boxed{\lambda = \text{Número de itens/Tamanho da tabela}}$ ou $\boxed{\lambda = 6/11}$.

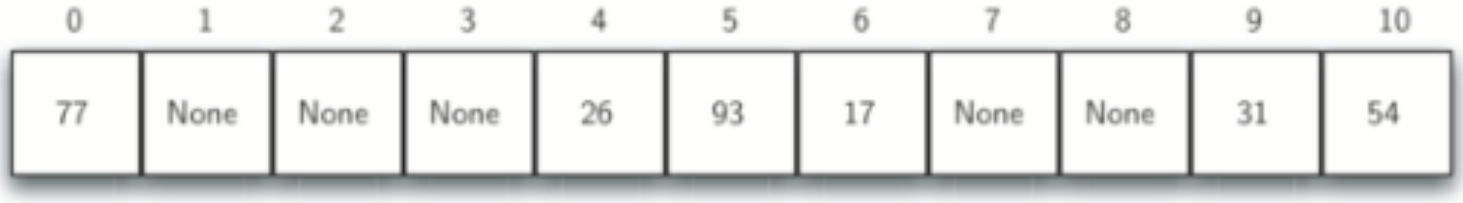


Tabela de Dispersão com Seis Itens (do exemplo do slide anterior)

IME-USP

- Temos um problema chamado de **colisão**!
- Por exemplo, se o item 44 fosse a próxima chave na nossa coleção, ele teria um valor de espalhamento de 0 (44%11==0). Como 77 também possui 0 como valor de espalhamento, teríamos um problema. Vamos ver como resolver isso mais adiante!

Hashing

- **Método de folding:** Considere esse número $436 - 555 - 4601$. Extrai-se os dígitos e dividi-se em grupos de 2 (43, 65, 55, 46, 01). Soma-se tudo (43+65+55+46+01) tem-se 210. Se a tabela de dispersão for de 11 . Então, 210% 11 é 1. Ou seja, o número de telefone 436-555-4601 seria mapeado para o índice 1. **Obs:.** Alguns métodos de folding vão além e trocam a ordem dos pedaços antes de somá-los.[1]
- **Método do quadrado:** Considera-se que o primeiro item é igual 44. Eleva-se ao quadrado obtendo-se $44^2 = 1936$. Extrai-se os dois dígitos do meio, 93, e realiza-se o passo de pegar o resto da divisão, obtendo-se o número 5 (93 % 11).[1]

Hashing

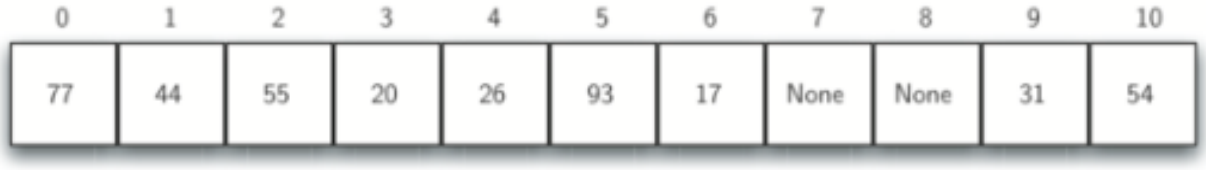
Problema das Colisões:

- Quando dois itens são levados à mesma posição pela função de espalhamento.
- É necessário um método para posicionar o segundo item.

Métodos de Resolução:

- **Endereçamento Aberto**: Procura a próxima entrada aberta. Vamos ver em detalhes adiante!

  Sondagem Linear: Move sequencialmente pelas entradas até encontrar uma posição vazia.



Resolução de Colisões com Sondagem Linear[1]

- Entretanto esse método gera um problema chamade de **aglutinação**, e para solucionar isso aplica-se pequenas modificações como a sondagem quadrática ou o encadeamento veja em Problem solving with algorithms and data structures using python.

O que é Endereçamento Aberto?

- Técnica de resolução de colisões em tabelas hash.
- Os elementos são armazenados diretamente na tabela.
- Quando ocorre uma colisão, o algoritmo procura uma nova posição seguindo uma estratégia de sondagem.

Estratégias de Sondagem

- **Sondagem Linear:** Procura sequencialmente a próxima posição vazia.
- **Sondagem Quadrática:** Utiliza uma função quadrática para determinar a próxima posição.

Exemplo: Sondagem Linear

- Considere uma tabela com 7 posições (índices 0 a 6) e uma função hash simples:

$$h(\text{chave}) = \text{chave} \mod 7$$

- **Inserção de 10:**

$$h(10) = 10 \mod 7 = 3 \quad \rightarrow \quad \text{Insere em 3 (se estiver vazia)}$$

- **Inserção de 3:**

$$h(3) = 3 \mod 7 = 3 \quad \rightarrow \quad \text{Verifica 3 (ocupado), tenta 4}$$

- **Inserção de 17:**

$$h(17) = 17 \mod 7 = 3 \quad \rightarrow \quad \text{Verifica 3, 4 (ocupados), insere em 5 (se vazio)}$$

- **Inserção de 24:**

$$h(24) = 24 \mod 7 = 3 \quad \rightarrow \quad \text{Verifica 3, 4, 5 (ocupados), insere em 6 (se vazio)}$$

Exemplo: Sondagem Quadrática

- Utiliza a fórmula:

$$h(i) = h(\text{chave}) + c_1 \cdot i + c_2 \cdot i^2$$

- **Exemplo:** Supondo $c_1 = 0$ e $c_2 = 1$ e $h(\text{chave}) = 3$:
  - 1^{a} tentativa: $3 + 1^2 = 4$
  - 2^{a} tentativa: $3 + 2^2 = 7$ (ajustado conforme o tamanho da tabela)
  - 3^{a} tentativa: $3 + 3^2 = 12$, e assim por diante.
- **Vantagem:** Reduz o clustering primário.
- **Desvantagem:** Pode ocorrer clustering secundário.

Eficiência em Tabelas Hash

Operation	List	Hash Table	
		expected	worst case
__getitem__	$O(n)$	$O(1)$	$O(n)$
__setitem__	$O(n)$	$O(1)$	$O(n)$
__delitem__	$O(n)$	$O(1)$	$O(n)$
__len__	$O(1)$	$O(1)$	$O(1)$
__iter__	$O(n)$	$O(n)$	$O(n)$

Comparação dos tempos de execução dos métodos de uma tabela hash.[2]

[2] Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). Data structures and algorithms in Python. John Wiley & Sons Ltd.

Considerações

Considerações

- **Recordando!!!**
  - Hashing.
  - Exercícios.