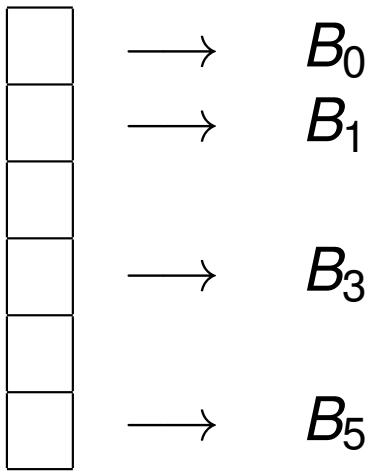
Tabelas de Dispersão (Hashtables)

Thiago Martins, Fabio Gagliardi Cozman

PMR2300 / PMR3201
Escola Politécnica da Universidade de São Paulo

Hashtables

- A ideia básica é espalhar os dados em um arranjo.
- Cada entrada do arranjo corresponde a um aglomerado "pequeno" de dados:



- Quando um dado é inserido, o aglomerado ("bucket") que o receberá é acessado e o dado é ali inserido.

Buckets e espalhamento

- Quando um dado é procurado, acessamos seu bucket e o procuramos ali.
- Como os buckets são “pequenos”, a procura é rápida.
- Para que isso funcione, temos que garantir que:
  1. os aglomerados sejam exaustivos, ou seja, um dado sempre é associado a um aglomerado;
  2. os aglomerados sejam mutuamente exclusivos, ou seja, só um aglomerado por dado;
  3. os dados sejam “bem espalhados” entre os aglomerados disponíveis.
- Precisamos assim de um esquema de *espalhamento* que produza o endereço de um bucket a partir de um dado (“função de hashing”).

Função usual: divisão Inteira

Bucket $h = \underbrace{K\%n}_{resto}$.

Exemplo: Suponha que temos 5 buckets, ou seja, $n = 5$:

- $54 \rightarrow 54\%5 = 4$;
- $41 \rightarrow 41\%5 = 1$;
- $4 \rightarrow 4\%5 = 4$.

Resultados teóricos indicam que melhor espalhamento é obtido para n primo.

Outros métodos

- Extração de dígitos:
  - 547790 $\rightarrow$ 577 (1^o, 3^o e 4^o dígitos);
  - 856430 $\rightarrow$ 85 (1^o e 2^o dígitos).
- Adição de dígitos:
  $123456789 \Rightarrow 123 + 456 + 789 = 1368$.
- Caso o dado não seja numérico, é preciso transformar para valor numérico. Por exemplo: cada caracter de uma String é substituído por seu código em Unicode.

Encadeamento Separado

- Até agora consideramos Hashtables onde os dados estão armazenados em uma estrutura auxiliar para cada bucket.
- Isso é denominado *encadeamento separado*.

Encadeamento Aberto: Estratégia Linear

- No *encadeamento aberto*, os dados estão armazenados diretamente em um arranjo único.
- Quando um dado é inserido, sua posição pode estar já ocupada (ocorre uma *colisão*).
- Considere a seguinte *estratégia linear*:
  - Se vamos inserir um dado com função de hashing por divisão inteira e o código $h = (K\%n)$ leva a uma posição no arranjo que que já está ocupada, tentamos $h+1, h+2, h+3, \ldots$, até encontrarmos uma posição vaga. É aqui que o dado será inserido.
  - Visitamos o arranjo de forma circular (se chegarmos ao seu final, retomamos no seu início).

Remoção e busca

- Para “remover” um item, simplesmente marcamos sua posição como “removida”.
- Para encontrarmos um dado, simplesmente fazemos os mesmos passos até que encontramos o dado ou encontrarmos uma célula vazia.

Estratégia quadrática

- A estratégia linear tende a agrupar dados em partes do arranjo, aumentando o número de colisões.

- Na estratégia quadrática usamos a sequência de endereços

$$(i + j^2)\%n$$

para $j = 0, 1, 2, \cdots$, até encontrar um elemento vago.

- Resultado: Se n é primo e o arranjo está com pelo menos metade dos elementos vazios, um elemento sempre é inserido, e nenhum elemento é visitado duas vezes durante a inserção.

Fator de Carga

- Fator de carga: razão entre número de dados e número de buckets.
- Normalmente deseja-se que uma Hashtable com encadeamento aberto opere com fator de carga bem menor que 1.

Rehashing

- Quando o fator de carga fica muito alto, o arranjo base de armazenamento precisa ter seu tamanho aumentado (um novo arranjo maior é criado, e os dados são copiados nele).
- Durante rehashing são também retiradas as posições marcadas como “removidas” (criadas por remoção de dados).

Custo para encadeamento aberto, estratégia linear

- Uma análise matemática detalhada revela que o número de acessos realizados em média para inserir um dado em uma Hashtable com encadeamento aberto linear é

$$M \approx \frac{1 + \frac{1}{(1-\lambda)^2}}{2}.$$

- Note:
  - para $\lambda = \frac{1}{2}$, temos $M \approx 2.5$;
  - para $\lambda = \frac{9}{10}$, temos $M \approx 50$.
- Esse custo vale para busca de um dado que não está na Hashtable. Se o dado está na Hashtable,

$$M \approx \frac{1 + \frac{1}{(1-\lambda)}}{2}.$$