

Pesquisa de Dados em Tabelas

Aula 12

Funções de Cálculo de Endereço e Tratamento de Colisões

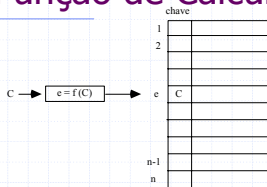
UFRGS

INF01124

Cálculo de Endereço (*hashing*)

- ◆ Método de cálculo de endereço
 - Aleatorização
 - Randomização
 - *Hashing*
- ◆ Não é apenas um método de pesquisa, mas também um método de organização física de tabelas
- ◆ Armazenamento de cada entrada em um *endereço calculado* pela aplicação de uma *função sobre o valor da chave da entrada*

Função de Cálculo de endereço



- ◆ Executam a transformação do valor de uma chave em um endereço, pela aplicação de operações aritméticas e/ou lógicas
 - f: $C \rightarrow E$
 - f: função de cálculo de endereço
 - C: espaço de valores da chave (domínio de f)
 - E: espaço de endereçamento (contradomínio de f)

- Os valores da chave podem ser numéricos, alfabéticos ou alfanuméricos

Eficiência

- ◆ A eficiência da pesquisa depende muito da função de cálculo de endereço adotada.
- ◆ *função ideal*: gera um endereço diferente (entre 1 e n) para cada um dos diferentes valores da chave presentes na tabela.
- ◆ Normalmente acontecem *colisões*: funções que atribuem o mesmo endereço a entradas com diferentes valores de chave.
 - se chaves C1, C2 e função f
 - colisão: $C1 \neq C2$ e $f(C1) = f(C2)$

Exemplo de colisões

- ◆ tabela com 23 entradas
- ◆ campo chave C com valores no intervalo [1...1000].
- ◆ função de cálculo de endereço:

$$f(C) = (C \bmod 23) + 1$$
- ◆ argumento de f(C): um valor entre 1 e 1000
- ◆ resultado de f(C): um endereço entre 1 e 23
- ◆ para cada endereço corresponde uma média de
 $1000/23 = 43,47$ valores possíveis da chave

C	383	487	235	527	510	320	203	108	563	500	646	103	063
E	16	05	06	22	05	22	20	17	12	18	03	12	18

Sinônimos: valores de chave para as quais é calculado mesmo endereço

Exercício:

- ◆ Calcule os endereços gerados pela função $f(C) = (\text{int})((C/1000) * 22,9 + 1)$ para os valores de chave do exercício anterior.

[illegible]

Tratamento de colisões

- ◆ Estabelecimento de uma disciplina para a escolha de uma entrada disponível onde será armazenada cada chave que vier a colidir durante o processo de inserção
- ◆ Principais Técnicas de Tratamento de Colisões:
 - Endereçamento Aberto
 - Com Encadeamento
 - Alocação de Blocos
 - Tabela Estendível

Endereçamento Aberto

- ◆ *Técnica:*
Armazenamento da chave que colidiu na primeira entrada livre a partir do endereço calculado
- ◆ Variantes do método: definidas pela sequência segundo a qual os endereços são examinados
 - com busca linear
 - com reateorização

Endereçamento Aberto com Busca Linear

- ◆ *Técnica:*
 - Procura-se uma entrada livre nos endereços da tabela, a partir do endereço calculado (e), na sequência e, e+1, ... n, 1, 2, ... e-1.
 - O primeiro endereço livre encontrado na sequência é utilizado para o armazenamento da nova entrada.
- ◆ *Exemplo:*
 - Inserir as 13 chaves abaixo em uma tabela de 23 entradas, usando a função

$$f(C) = (C \bmod 23) + 1 = e$$

C	383	487	235	527	510	320	203	108	563	500	646	103	063
E	16	05	06	22	05	22	20	17	12	18	03	12	18

Endereçamento Aberto com Busca Linear

Endereço	Chave	Informação	Ocupado	Usado
1			F	V
2			F	F
3	646		V	V
4			F	F
5	487		V	V
6	235		V	V
7	510		V	V
8			F	V
9			F	V
10			F	F
11			F	F
12	563		V	V
13	103		V	V
14			F	V
15			F	F
16	383		V	V
17	108		V	V
18	500		V	V
19	063		V	V
20	203		V	V
21			F	F
22	527		V	V
23	320		V	V

C	383	487	235	527	510	320	203	108	563	500	646	103	063
E	16	05	06	22	05	22	20	17	12	18	03	12	18

Endereçamento Aberto com Busca Linear

- ◆ *Campos OCUPADO e USADO:*
 - OCUPADO:
 - V nos endereços ocupados
 - F nos endereços livres da tabela
 - USADO:
 - V nos endereços da tabela que estão ou já estiveram ocupados desde a criação da tabela
 - F nos demais endereços
- ◆ *Exclusão de entrada da tabela:*
 - campo OCUPADO passa de V para F
 - campo USADO permanece com valor V
- ◆ *Consulta de entrada da tabela:*
A entrada procurada não está na tabela se for encontrado um endereço ainda não usado desde a criação da tabela (usado = F).

Endereçamento Aberto com Busca Linear

```

procedure pesquisa_eabl ( tab: tabela; arg: chave; e: int);
{ tab: tabela em que será feita a busca
  arg: argumento de pesquisa (chave)
  e: endereço da chave procurada, se e = 0, a chave procurada está ausente }
var endl, endc, n : int;
begin
  e:=0; n := #(tab);
  endc:= (arg mod n)+1;      {endereço calculado}
  endl:=endc;                {endereço livre}
  repeat
    if tab[endl].usado
    then if tab[endl].ocupado and tab[endl].chave=arg
      then e := endl;
      else if endl=n {incrementa endereço}
        then endl:=1
      else endl:= endl+1;
      else escape {força o final do loop}
    until (endl=endc) or (e<>0)
  end { pesquisa_eabl };
    
```

Endereçamento Aberto com Busca Linear

```
procedure insere_absl (tab: tabela; ent: entrada; e, marca: int);
{ tab: tabela em que será feita a inserção
  ent: entrada a ser inserida }
var endi, endc, n, marca: int;  arg: chave;
begin
  marca:=0; e:=0; n:=#(tab); arg:=ent.chave;
  endc:= (arg mod n)+1;      (endereço calculado)
  endi:=endc;                (endereço livre)
  repeat
    if tab[endi] usado
    then if tab[endi].ocupado
    then if tab[endi].chave = arg
    then e:=endi (Chave presente. Encerra)
    else endi:= (endi mod n)+1;
    else begin
      if marca=0 then marca:=endi;
      endi:= (endi mod n) + 1
    end
  else begin
    if marca=0 then marca:=endi;
    escape      (força fim do loop)
  end
until (endi=endc) or (e<>0) (Final da pesquisa. Verifica se é possível inserir)
if (e=0) and (marca>0)
then begin
  tab[marca].:=entrada;      (insere no endereço marca)
  tab[marca].usado := True;
  tab[marca].ocupado := True;
end
end; (insere_absl)
```

Exercício:

- ◆ Simule o funcionamento dos algoritmos de pesquisa e inserção para um exemplo:
 - Observe que, nos dois procedimentos, é usada a função de cálculo de endereço dada por $f(c)=(c \bmod n)+1$

Endereçamento Aberto com Realeatorização

- ◆ **Busca Linear:** Se o endereço e estiver ocupado tenta no endereço $e+1, e+2, \dots, n, 1, 2, \dots, e-1$
- ◆ **Com Realeatorização:** Se o endereço e estiver ocupado tenta nos endereços $e+i, e+2i, \dots$ onde $i=f_2(C)$
 f_2 : função de realeatorização
- ◆ **Vantagem:** diminuição da probabilidade de formação de longas seqüências de endereços ocupados, responsáveis pela degradação da eficiência da pesquisa

Endereçamento Aberto com Realeatorização

- ◆ **Regra Fundamental:**
os incrementos i e o tamanho da tabela n devem ser primos entre si, ou seja, não devem ter nenhum divisor inteiro comum, exceto a unidade
- ◆ **Exemplo:**
 $n=20$
 $endi=10$
 $i=2 \rightarrow$ analisa somente metade da tabela
 $i=3 \rightarrow$ analisa toda a tabela
 - Na prática: adoção de um número primo como valor de n é suficiente. Neste caso, qualquer incremento $i < n$ é satisfatório.

Exercício:

- ◆ Modifique os algoritmos de pesquisa e inserção para o caso de endereçamento aberto com realeatorização.

Desempenho do Método de Endereçamento Aberto

- ◆ **Fatores de que depende o desempenho:**
 - Taxa de ocupação t_o da tabela:
$$t_o = \frac{\text{número de endereços ocupados}}{\text{número total de endereços}}$$

Quanto maior o valor de t_o , maior o número médio de endereços examinados durante a busca.
 - Método de escolha do endereço livre a ser utilizado:
busca linear ou realeatorização

Análise: Endereçamento Aberto

Busca Linear X Realeatorização

- ◆ O desempenho do método de realeatorização é melhor por causa da maior dispersão que ele provoca na ocupação das entradas
- ◆ O método de busca linear tende a gerar seqüências longas de entradas ocupadas
- ◆ Este fenômeno é tão mais sensível quanto maior a taxa de ocupação da tabela