MC202 – Estruturas de Dados

Aula 19 - Tabelas de Espalhamento

12 de maio de 2017

Emilio Francesquini – UNICAMP
francesquini@ic.unicamp.br

Disclaimer

- Esses slides foram preparados pelo professor Orlando Lee para o curso de Estrutura de Dados ministrado na UNICAMP.
- Este material pode ser usado livremente desde que sejam mantido os créditos dos autores e da instituição.
- Muitos dos exemplos apresentados aqui foram retirados de uma apostila ou slides preparados pelo Prof. Tomasz Kowaltowski da UNICAMP. Outros exemplos foram retirados do livro "Algoritmos em Linguagem C" de Paulo Feofiloff, Editora Campus.

Referências


T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein (2001),
"Introduction to Algorithms", 2nd edition, The MIT Press, pp. 434–454.

Tabelas de espalhamento/dispersão (hash tables)

- Muitas aplicações requerem um conjunto dinâmico com as operações de busca, inserção e remoção.

Tabelas de espalhamento/dispersão (hash tables)

- Muitas aplicações requerem um conjunto dinâmico com as operações de busca, inserção e remoção.
- Por exemplo, um compilador que traduz uma linguagem de programação mantém uma tabela de símbolos, na qual as chaves dos elementos são strings que correspondem a identificadores na liguagem.

Tabelas de espalhamento/dispersão (hash tables)

- Muitas aplicações requerem um conjunto dinâmico com as operações de busca, inserção e remoção.
- Por exemplo, um compilador que traduz uma linguagem de programação mantém uma tabela de símbolos, na qual as chaves dos elementos são strings que correspondem a identificadores na liguagem.
- Tabela de espalhamento (hash table) é uma alternativa bastante atraente se comparada com implementações vistas:
  - é mais simples de implementar, e
  - sob hipóteses razoáveis, o tempo médio de busca é sempre constante (não depende do número de elementos na tabela).

Problema típico

- Temos um conjunto de elementos no qual cada um deles está associado a uma chave de algum conjunto universo U. O conjunto U pode ser formado por inteiros, strings ou tipos mais complicados.

Problema típico

- Temos um conjunto de elementos no qual cada um deles está associado a uma chave de algum conjunto universo U. O conjunto U pode ser formado por inteiros, strings ou tipos mais complicados.
- Dependendo da aplicação, o conjunto de elementos pode ser dinâmico (muda ao longo do tempo), mas o conjunto de universo é fixo.

Problema típico

- Temos um conjunto de elementos no qual cada um deles está associado a uma chave de algum conjunto universo U. O conjunto U pode ser formado por inteiros, strings ou tipos mais complicados.
- Dependendo da aplicação, o conjunto de elementos pode ser dinâmico (muda ao longo do tempo), mas o conjunto de universo é fixo.
- Queremos armazenar este conjunto de elementos de modo que suporte as operações de busca, inserção e remoção de modo eficiente.

Representações de conjuntos dinâmicos

N é o número de chaves (elementos).

Implementação	Pior caso			Caso médio		
	busca	inserção	remoção	busca	inserção	remoção
vetor não-ordenado	N	N	N	$N/2$	N	$N/2$
vetor ordenado	$\lg N$	N	N	$\lg N$	$N/2$	$N/2$
árvore de busca	N	N	N	$1.39N$	$1.39N$	$\sqrt{N}$
splay trees	N	N	N	$O(\lg N)$	$O(\lg N)$	$O(\lg N)$
árvore AVL e rubro-negra	$O(\lg N)$	$O(\lg N)$	$O(\lg N)$	$O(\lg N)$	$O(\lg N)$	$O(\lg N)$

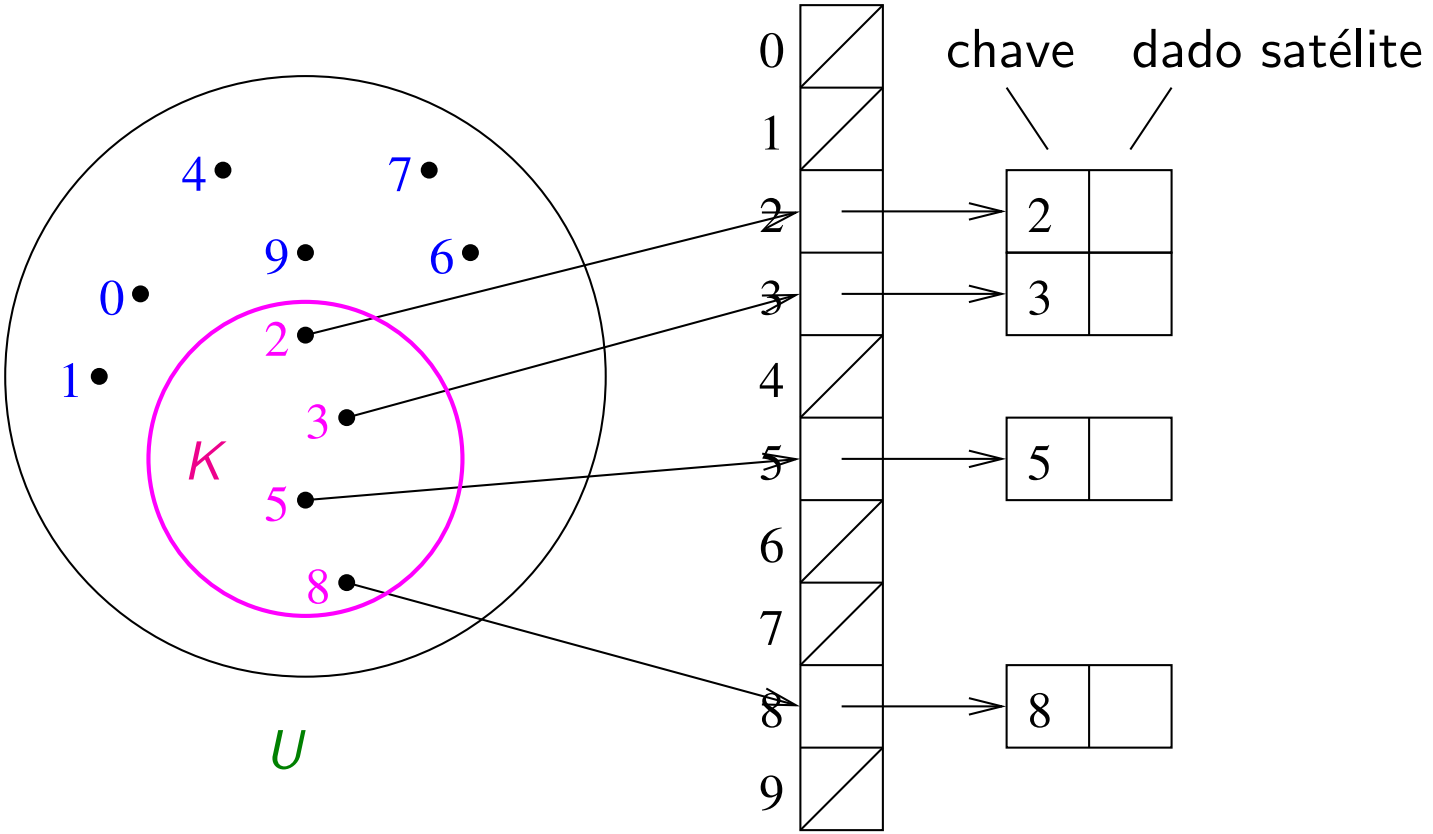
Em uma tabela de espalhamento, o tempo de pior caso de busca ou inserção pode ser $O(N)$, mas o tempo médio esperado é $O(1)$, ou seja, é uma constante.

Tabelas de endereçamento direto

- Endereçamento direto (direct addressing) é uma técnica simples que funciona bem quando o universo U de chaves é pequeno.
- Suponha que uma aplicação precisa de um conjunto dinâmico na qual cada elemento tem uma chave do universo $U = \{0, \ldots, M-1\}$ onde M não é muuuito grande.
- Suponha que não haja dois elementos com a mesma chave.

Tabelas de endereçamento direto

Para representar o conjunto dinâmico usamos uma tabela de endereçamento direto (direct-address table) $T[0\,..\,M-1]$ na qual cada posição/índice corresponde a uma chave.



Tabelas de endereçamento direto

A posição k da tabela aponta para um elemento com chave k. Se não existir elemento com chave k, então $T[k] = \text{NULL}$.

A implementação das operações busca, inserção e remoção é trivial.

TED-BUSCA(T, k)
1. **return** $T[k]$

TED-INSERE(T, x)
1. $T[x.chave] = x$

TED-REMOVE(T, x)
1. $T[x.chave] = \text{NULL}$

Desvantagens do endereçamento direto

- se o universo U for grande, armazenar uma tabela T de tamanho $|U|$ pode ser impraticável ou mesmo impossível;

Desvantagens do endereçamento direto

- se o universo U for grande, armazenar uma tabela T de tamanho $|U|$ pode ser impraticável ou mesmo impossível;
- o conjunto K das chaves que são de fato armazenadas pode ser muito menor que $|U|$ (todas as possíveis chaves).
  Isto implica em perda de espaço muito grande;

Desvantagens do endereçamento direto

- se o universo U for grande, armazenar uma tabela T de tamanho $|U|$ pode ser impraticável ou mesmo impossível;
- o conjunto K das chaves que são de fato armazenadas pode ser muito menor que $|U|$ (todas as possíveis chaves).
  Isto implica em perda de espaço muito grande;
- tabelas de espalhamento exigem muito menos espaço que tabelas de endereçamento direto: tamanho da tabela é $O(|K|)$ (tipicamente um múltiplo pequeno de $|K|$).

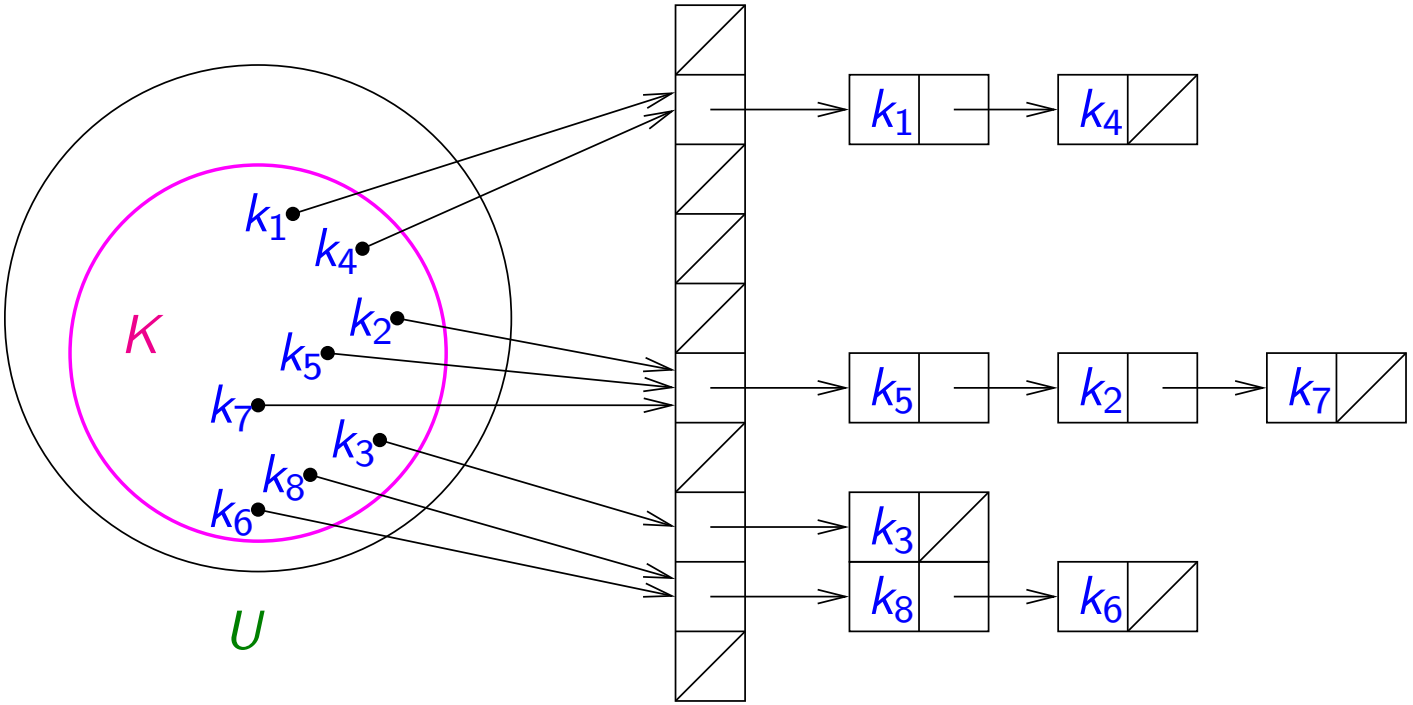
Tabelas de espalhamento/dispersão (hash tables)

- Endereçamento direto: elemento com chave k vai para a posição k da tabela.
- Espalhamento (hashing): elemento com chave k vai para a posição $h(k)$ da tabela.

A função h é chamada função de espalhamento (hash function). Ela mapeia cada chave do universo U em uma posição da tabela $T[0\,..\,M-1]$:

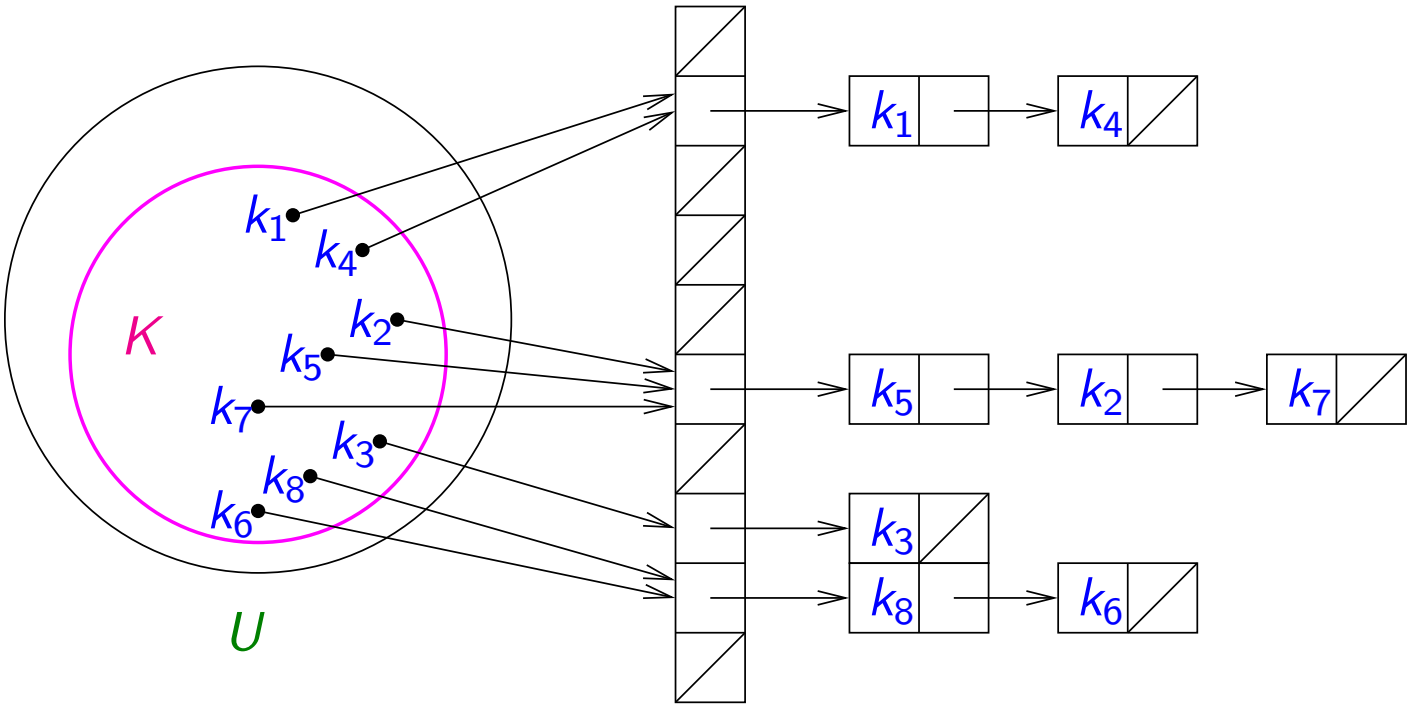
$h : U \mapsto \{0, \ldots, M-1\}$.

Tabelas de espalhamento/dispersão (hash tables)



Tipicamente $|U| >> |M|$. A função de espalhamento h reduz o conjunto de índices disponíveis. Por isto a tabela tem tamanho M em vez de $|U|$.

Tabelas de espalhamento/dispersão (hash tables)



Há um porém: duas chaves podem ser mapeadas ao mesmo índice pela função h. Isto é chamado de colisão. Felizmente, há várias técnicas para lidar com conflitos causados por colisão.

Funções de espalhamento

Há duas características desejáveis de uma função de espalhamento:

- ser eficientemente computável
- qualquer índice da tabela é igualmente provável para cada chave (uniform hashing).

O segundo requisito é bem mais difícil de atender apesar de muita pesquisa ter sido feita.

Além disso, uma estratégia que é boa para um certo conjunto de chaves, pode ser ruim para outro conjunto.

Exemplos de funções de espalhamento (hash functions)

- Método da divisão. Suponha que $U = \{0, \ldots, N-1\}$.Uma função de espalhamento bem simples seria:

  $h(k) = k \bmod M$.
- Um problema importante é como escolher M. Uma escolha ruim seria tomar uma potência de 2 como $M = 2^p$ pois $h(k)$ nada mais é do que tomaros p bits menos significativos de k.
- Uma escolha boa é tomar M como um número primo que não esteja muito próximo de uma potência de 2.
- Este método é frequentemente combinado com outros métodos.

Exemplos de funções de espalhamento (hash functions)

- Chaves arbitrárias. suponha que U seja um conjunto de strings, digamos nomes de pessoas. Suponha que as letras são minúsculas para simplificar.
- Interprete $a = 0, b = 1, c = 2$, etc. Usamos $M = 26$.
  Uma função de espalhamento bem simples é usar esta a primeira letra do nome:

  $h(\text{ana}) = 0$, $h(\text{lucas}) = 11$, $h(\text{tiago}) = 19$.

- Evidentemente, esta é uma escolha ruim de h pois há muitas colisões.

Exemplos de funções de espalhamento (hash functions)

- Uma ideia mais sofisiticada é interpretar o nome como um número em uma certa base b e usar o método da divisão. Por exemplo, podemos pensar que cada caractere é um dígito na base $b = 26$. Por exemplo,

$$\begin{aligned} \text{carlos} &= 1 \times 26^5 + 0 \times 26^4 + 17 \times 26^3 + 11 \times 26^2 + \\ &\quad 14 \times 26^1 + 18 \times 26^0 \\ &= 24.069.362. \end{aligned}$$

Se $M = 51$ então $f(\text{carlos}) = 24.069.362 \bmod 51 = 14$.

Exemplos de funções de espalhamento (hash functions)

- No exemplo anterior, os números representados pela string podem ser muito grandes. Para evitar isto, é melhor aplicar o módulo para resultados intermediários.
- Lembre-se que
  $(a + b) \bmod M = ((a \bmod M) + (b \bmod M)) \bmod M$ e
  $(a \times b) \bmod M = ((a \bmod M) \times (b \bmod M)) \bmod M$

Exemplos de funções de espalhamento (hash functions)

- No exemplo anterior, os números representados pela string podem ser muito grandes. Para evitar isto, é melhor aplicar o módulo para resultados intermediários.
- Lembre-se que
  $(a + b) \bmod M = ((a \bmod M) + (b \bmod M)) \bmod M$ e
  $(a \times b) \bmod M = ((a \bmod M) \times (b \bmod M)) \bmod M$
- De fato, podemos interpretar qualquer conjunto universo U como um conjunto de inteiros (possivelmente muito grande). Isto por sua vez permite usar o método da divisão e o método de multiplicação que veremos a seguir.

Exemplos de funções de espalhamento (hash functions)

- Método da multiplicação. Ela consiste de dois passos.
- Multiplicamos a chave k por uma constante A, com $0 < A < 1$ e extraímos a parte fracionária de kA. Então multiplicamos este valor por M e arredondamos para baixo.
- Em linguagem matemática:

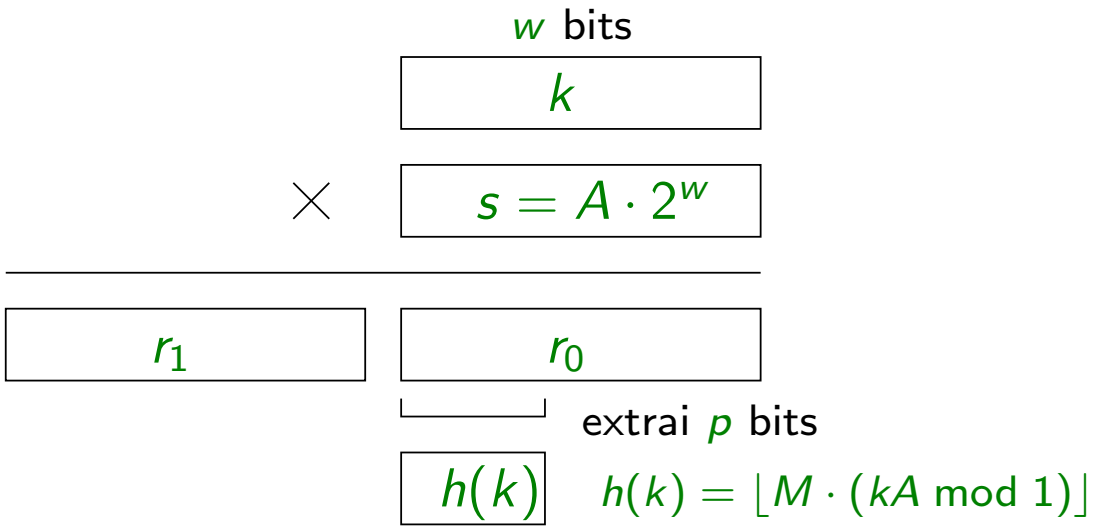
  $h(k) = \lfloor M \cdot (kA \bmod 1) \rfloor$,

  onde $kA \bmod 1$ é a parte fracionária de de kA (ou seja, $kA - \lfloor kA \rfloor$).

Exemplos de funções de espalhamento (hash functions)

- Uma vantagem do método da multiplicação é que o valor de M não é crítico.
- Tipicamente, escolhe-se $M = 2^p$ pois o método é mais fácil de implementar como mostramos a seguir.
- Suponha que o tamanho de uma palavra da máquina usada seja de w bits e que k cabe em uma palavra. Escolhemos A como uma fração da forma $s/2^w$ onde s é um inteiro no intervalo $0 < s < 2^w$.

Exemplos de funções de espalhamento (hash functions)



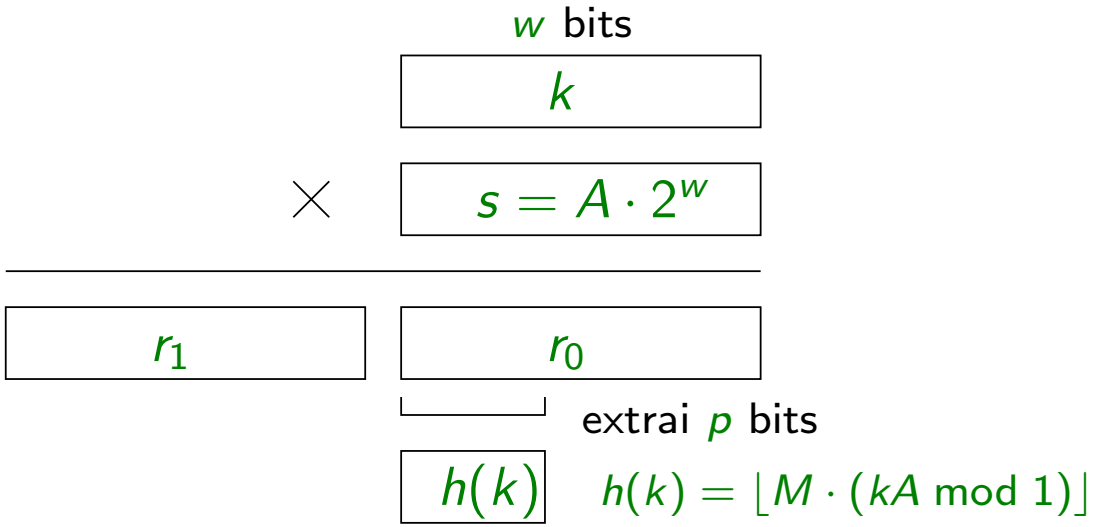
- Multiplicando k por $s = A \cdot 2^w$ obtemos um número de $2w$ bits da forma $r_1 2^w + r_0$.
- A função de espalhamento consiste em extrair os p bits mais significativos de r_0. (Explicado nos próximos slides)

Números em representação binária

Antes de explicar a ideia, é preciso lembrar/aprender algumas coisas.

- Para exemplo, tome $n = 139$. A representação de n em bits é $(10001011)_2$ com $w = 8$ bits.
- Multiplicar n por 2 equivale a fazer um shift de um bit para a esquerda. Logo $2n = 278 = (100010110)_2$. Multiplicar por 2^p equivale a fazer um shift de p bits para a esquerda. Por exemplo, se $p = 3$ então $2^p n = 8n = (10001011000)_2$.
- Dividir n por 2 e arredondar para baixo equivale a fazer um shift de um bit para a direita. Logo, $\lfloor n/2 \rfloor = 69 = (1000101)_2$. Dividir por 2^p e arredondar para baixo equivale a fazer um shift de p bits para a direita. Por exemplo, se $p = 3$ então $\lfloor n/2^p \rfloor = \lfloor n/8 \rfloor = 17 = (10001)_2$.

Exemplos de funções de espalhamento (hash functions)



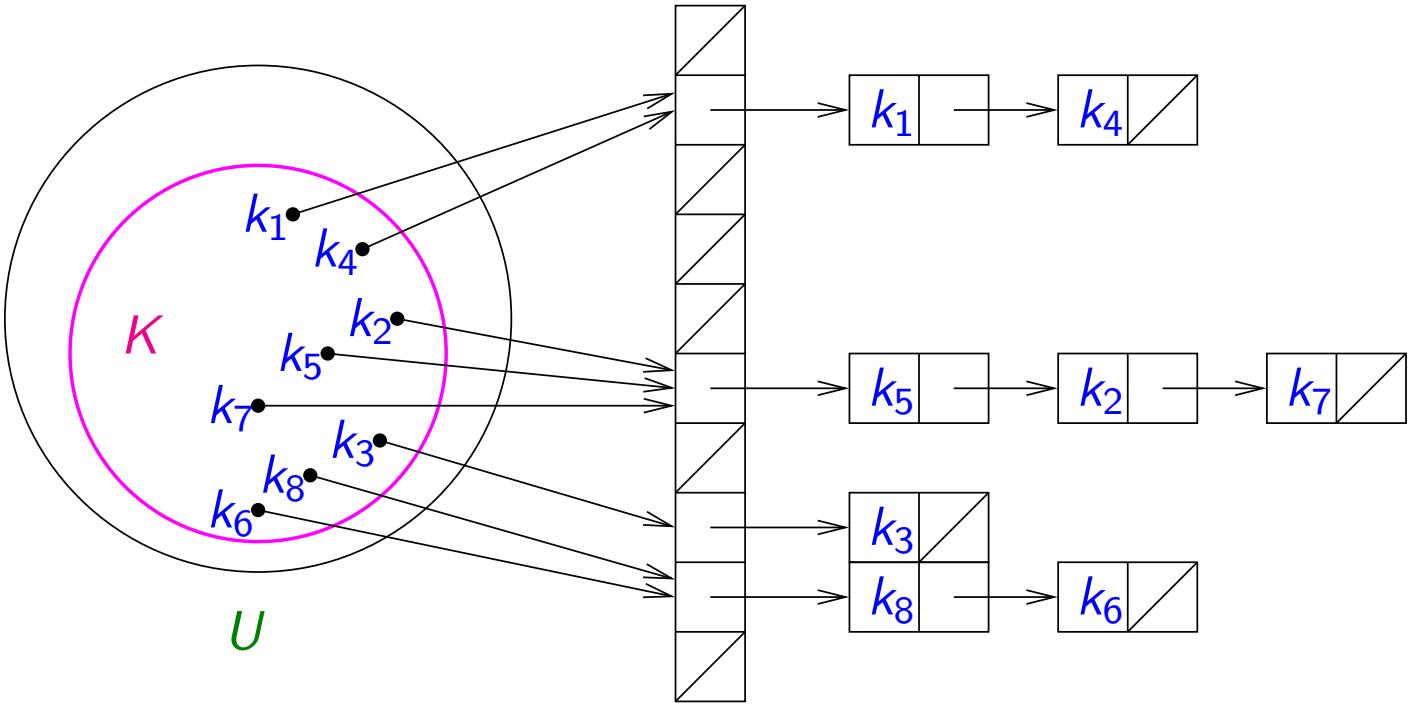
- Note que $h(k) = \lfloor M \cdot (kA \bmod 1) \rfloor = \lfloor 2^p \cdot (\frac{ks}{2^w} \bmod 1) \rfloor$.
- Além disso, $ks = r_1 2^w + r_0$. Logo, $\frac{ks}{2^w} = \frac{r_0}{2^w}$.
- Finalmente, $h(k) = \lfloor 2^p \cdot \frac{r_0}{2^w} \rfloor = \lfloor \frac{r_0}{2^{w-p}} \rfloor$ o que equivale a pegar os p bits mais significativos de r_0.

Tratamento de colisões

- Idealmente a função h deveria mapear cada chave de K (chaves de fato armazenadas) em um único índice da tabela de espalhamento. Isto permitiria uma consulta extremamente rápida para qualquer chave.
- Infelizmente, isto nem sempre é possível de se obter pois à medida que novas chaves surgem, a chance de colisão aumenta. Assim, é necessário criar estratégias para lidar com colisões.
- Há duas estratégias clássicas para lidar com colisões:
  - encadeamento e
  - endereçamento aberto.

Encadeamento

Ideia: manter as chaves com mesmo valor de função de espalhamento em uma lista ligada (simples ou duplamente ligada).



Encadeamento

Ideia: manter as chaves com mesmo valor de função de espalhamento em uma lista ligada simples ou duplamente ligada (cadeia).

CADEIA-INSERE(T, x)
1. insira x no início da lista $T[h(x.chave)]$

CADEIA-BUSCA(T, k)
1. procure por um elemento com chave k na lista $T[h(k)]$

CADEIA-REMOVE(T, x)
1. remova x da lista $T[h(x.chave)]$

Encadeamento

- A eficiência desta implementação depende do tamanho M da tabela e do número N de chaves existentes na tabela.
- O valor $\alpha = N/M$ é o fator de carga da tabela.
- O valor α também é o tamanho médio de uma lista ligada. Assim, o tempo médio de busca é $\approx 1 + \alpha/2$ (supondo uma função de espalhamento uniforme).
- O ideal seria que α fosse constante.

Encadeamento

Veja a tabela abaixo para $M = 1000$. Lembre que $\alpha = N/M$.

N	$1 + \alpha/2$
100	1,05
200	1,10
400	1,20
500	1,25
1000	1,50
2000	2,00

Eis um resumo sobre tratamento de colisões por encadeamento:

- fácil de implementar;
- o fator de carga aumenta lentamente;
- a tabela nunca fica cheia.

Endereçamento aberto

- Na técnica de endereçamento aberto cada posição da tabela de espalhamento pode conter um elemento do conjunto dinâmico ou NULL.

Endereçamento aberto

- Na técnica de endereçamento aberto cada posição da tabela de espalhamento pode conter um elemento do conjunto dinâmico ou NULL.
- Não há listas ligadas nem elementos fora da tabela de espalhamento.

Endereçamento aberto

- Na técnica de endereçamento aberto cada posição da tabela de espalhamento pode conter um elemento do conjunto dinâmico ou NULL.
- Não há listas ligadas nem elementos fora da tabela de espalhamento.
- Assim, uma tabela pode ficar cheia quando usamos endereçamento aberto (logo $\alpha = N/M \leq 1$).

Endereçamento aberto

- Na técnica de endereçamento aberto cada posição da tabela de espalhamento pode conter um elemento do conjunto dinâmico ou NULL.
- Não há listas ligadas nem elementos fora da tabela de espalhamento.
- Assim, uma tabela pode ficar cheia quando usamos endereçamento aberto (logo $\alpha = N/M \leq 1$).
- Quando há uma colisão, o algoritmo procura de modo sistemático uma posição disponível para inserir o novo elemento. Isto deve ser feito de modo que seja possível encontrá-lo ou removê-lo em outra consulta.

Sondagem linear (linear probing)

- Suponha que k seja uma chave e ao tentarmos inserir k na posição $h(k)$, descobrimos que ela já está ocupada.
- O método de sondagem linear (linear probing) consiste em examinas as posições:

  $$h(k)+1, h(k)+2, \ldots, M-1, 0, 1, \ldots h(k)-1$$

  até encontrar uma posição livre. Em outras palavras, o algoritmo sonda as posições $(h(k)+i) \bmod M$ para $i = 1, 2, \ldots$

Sondagem linear (linear probing)

- Suponha que k seja uma chave e ao tentarmos inserir k na posição $h(k)$, descobrimos que ela já está ocupada.
- O método de sondagem linear (linear probing) consiste em examinas as posições:

  $$h(k)+1, h(k)+2, \ldots, M-1, 0, 1, \ldots h(k)-1$$

  até encontrar uma posição livre. Em outras palavras, o algoritmo sonda as posições $(h(k)+i) \bmod M$ para $i = 1, 2, \ldots$
- Um problema deste método é a formação de clusters longos (blocos de posições consecutivas ocupadas) na tabela.

Exemplo de sondagem linear

0	ana
1	
2	caio
3	
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem linear

0	ana
1	
2	caio
3	david
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem linear

0	ana
1	adão
2	caio
3	david
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem linear

0	ana
1	adão
2	caio
3	david
4	césar
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem linear

0	ana
1	adão
2	caio
3	david
4	césar
5	amélia
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem linear

0	ana
1	adão
2	caio
3	david
4	césar
5	amélia
6	arnaldo
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Sondagem quadrática

- Suponha que k seja uma chave e ao tentarmos inserir k na posição $h(k)$, descobrimos que ela já está ocupada.
- O método de sondagem quadrática (quadratic probing) consiste em sondar as posições $(h(k) + i^2) \bmod M$ para $i = 1, 2, \ldots$

Sondagem quadrática

- Suponha que k seja uma chave e ao tentarmos inserir k na posição $h(k)$, descobrimos que ela já está ocupada.
- O método de sondagem quadrática (quadratic probing) consiste em sondar as posições $(h(k) + i^2) \bmod M$ para $i = 1, 2, \ldots$
- Diminui a formação de clusters longos.
- Em alguns textos usa-se a seguinte definição. Escolhe-se a priori constantes $c_1, c_2 \geq 0$.
  O algoritmo sonda as posições $h(k) + c_1 i + c_2 i^2 \bmod M$ para $i = 1, 2, \ldots$

  Quando $c_1 = 0$ e $c_2 = 1$ temos o método descrito acima.

Exemplo de sondagem quadrática

0	ana
1	
2	caio
3	
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem quadrática

0	ana
1	
2	caio
3	david
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem quadrática

0	ana
1	adão
2	caio
3	david
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem quadrática

0	ana
1	adão
2	caio
3	david
4	
5	
6	césar
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem quadrática

0	ana
1	adão
2	caio
3	david
4	amélia
5	
6	césar
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de sondagem quadrática

0	ana
1	adão
2	caio
3	david
4	amélia
5	
6	césar
7	
8	
9	arnaldo
	...
25	

Suponha que $h(k)$ é a primeira letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Espalhamento duplo (double hashing)

- Neste método temos uma segunda função de espalhamento g. Em caso de colisão (a posição $h(k)$ está ocupada), o algoritmo sonda as posições $(h(k) + i \cdot g(k)) \bmod M$ para $i = 1, 2, \ldots$
- Para ter certeza que a busca vai examinar todas as posições da tabela, é necessário que o valor $g(k)$ seja relativamente primo com o tamanho da tabela M.
- Um modo de garantir isto é tomar M primo e projetar g de modo que $g(k) < M$. Por exemplo,

  $h(k) = k \bmod M$

  $g(k) = 1 + (k \bmod M')$

  onde M' é um pouco menor que M (digamos, $M' = M - 1$).

Exemplo de espalhamento duplo

0	ana
1	
2	caio
3	
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k e $g(k) = 1 + (c \bmod 5)$ onde c é a segunda letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de espalhamento duplo

0	ana
1	
2	caio
3	david
4	
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k e $g(k) = 1 + (c \bmod 5)$ onde c é a segunda letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de espalhamento duplo

0	ana
1	
2	caio
3	david
4	adão
5	
6	
7	
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k e $g(k) = 1 + (c \bmod 5)$ onde c é a segunda letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de espalhamento duplo

0	ana
1	
2	caio
3	david
4	adão
5	
6	
7	césar
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k e $g(k) = 1 + (c \bmod 5)$ onde c é a segunda letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de espalhamento duplo

0	ana
1	
2	caio
3	david
4	adão
5	
6	amélia
7	césar
8	
9	
	...
25	

Suponha que $h(k)$ é a primeira letra de k e $g(k) = 1 + (c \bmod 5)$ onde c é a segunda letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Exemplo de espalhamento duplo

0	ana
1	
2	caio
3	david
4	adão
5	
6	amélia
7	césar
8	
9	arnaldo
	...
25	

Suponha que $h(k)$ é a primeira letra de k e $g(k) = 1 + (c \bmod 5)$ onde c é a segunda letra de k. Considere a inserção de: david, adão, césar, amélia, arnaldo.

Remoção em endereçamento aberto

- A remoção em uma tabela de espalhamento com endereçamento aberto é um pouco mais complicada. Suponha que queremos remover o elemento de chave k da tabela.

Remoção em endereçamento aberto

- A remoção em uma tabela de espalhamento com endereçamento aberto é um pouco mais complicada. Suponha que queremos remover o elemento de chave k da tabela.
- A partir da posição $h(k)$ temos que procurar k usando a sequência de sondagem para descobrir a posição verdadeira deste. Suponha que ele esteja na posição j. Não podemos simplesmente colocar $T[j] = \text{NULL}$. (por quê?)

Remoção em endereçamento aberto

- A remoção em uma tabela de espalhamento com endereçamento aberto é um pouco mais complicada. Suponha que queremos remover o elemento de chave k da tabela.
- A partir da posição $h(k)$ temos que procurar k usando a sequência de sondagem para descobrir a posição verdadeira deste. Suponha que ele esteja na posição j. Não podemos simplesmente colocar $T[j] = \text{NULL}$. (por quê?)
- Podemos resolver isto colocando um marcador REMOVIDO em vez de NULL. Isto indica que uma busca não deve parar neste ponto. Além disso, para a inserção, esta corresponderia a uma posição livre.

Endereçamento aberto

Os seguintes resultados são conhecidos e não-triviais de provar. Lembre que $\alpha = N/M < 1$ é o fator de carga da tabela. A análise supõe que vale espalhamento uniforme e que não há remoção.

- O número médio de sondagens em uma busca sem sucesso ou uma inserção é $\frac{1}{1-\alpha}$.
- O médio de sondagens em uma busca com sucesso é $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$.
- Quando há remoção, a análise não depende apenas de α. Em geral, neste caso prefere-se usar encadeamento.

Espalhamento

- Vários resultados teóricos dependerem da hipótese de espalhamento uniforme.
- Na prática, usa-se heurísticas ou conhecimento sobre as chaves para se projetar boas funções de espalhamento.
- Do ponto de vista prático, seu desempenho não é pior do que as implementações com garantia de pior caso.
- Tabelas de espalhamento são usadas em várias bibliotecas por causa de duas características: facilidade de implementação e flexibilidade.