

Hash

Definição e endereçamento aberto

Prof. Edson Alves - UnB/FGA

2019

1. Definição de hash
2. Exemplos de funções *hash*
3. Endereçamento aberto

Definição de hash

Motivação para a criação da hash

- Estruturas lineares permitem N armazenar elementos sem que o valor de N seja conhecido *a priori*, em tempo de compilação
- Estas estruturas, porém, não são eficientes na busca do elementos (complexidade $O(N)$)
- As árvores de busca, como as estruturas lineares, também permitem o armazenamento de um número arbitrário de elementos (limitado somente pela memória disponível)
- Em árvores busca perfeitamente balanceadas a ordem de complexidade da busca é $O(\log N)$
- Porém a organização de memória das árvores não é contígua, levando à perda de eficiência em relação à *cache*
- A idéia da *hash* é deduzir o índice de um elemento em um vetor a partir apenas da informação armazenada pelo elemento, reduzindo a ordem de complexidade busca para $O(1)$

Definição

Uma função h é uma função de *hash* se ela transforma uma chave K no índice do elemento que contém K na tabela. Se h transforma chaves distintas em índices distintos, ela é uma função de *hash* perfeita.

- Para se criar uma função h de *hash* perfeita, a tabela deve conter, no mínimo, o número de elementos cujas chaves serão transformadas pela função h
- Uma colisão ocorre se duas chaves distintas K_1 e K_2 gerarem o mesmo índice, isto é, se $h(K_1) = h(K_2)$ com $K_1 \neq K_2$
- A idéia é encontrar uma função h que gere o mínimo de colisões mas que não seja sofisticada ao ponto de seu cálculo interferir na performance do programa

Exemplos de funções *hash*

Soma dos elementos

Chave	Uma string K
Algoritmo	Atribui-se um código numérico para cada um dos caracteres que aparecem na string K e somam-se estes valores
Nível de colisão	Alto

```
1 int h(const string& K)
2 {
3     int v = 0;
4
5     for (const auto& c : K)
6         v += c;
7
8     return v;
9 }
```

Resto da divisão

Chave	Um inteiro K
Algoritmo	Obtêm-se o resto da divisão da chave K pelo tamanho T da tabela. De preferência, T deve ser um número primo
Nível de colisão	Inversamente proporcional a T

```
1 unsigned long h(unsigned long K, size_t T)
2 {
3     return K % T;
4 }
```


Enlaçamento deslocado

Chave	Uma string K
Algoritmo	Divide-se a string em N partes de, no máximo, m caracteres, e computa-se a chave aplicando-se a operação XOR em todas as partes
Nível de colisão	Médio

```
1 unsigned long h(const string& S) {  
2     unsigned long v = 0, p = 0, m = 4, i = 0;  
3  
4     for (const auto& c : S) {  
5         p |= (c << i++);  
6  
7         i == m ? (v ^= p) : (i = p = 0);  
8     }  
9  
10    return v ^ p;  
11 }
```

Enlaçamento no limite

Chave	Uma string K
Algoritmo	Variante do enlaçamento deslocado. Divide-se a chave em 3 partes, e se enlaça os extremos com a parte do meio invertida
Nível de colisão	Médio

```
1 unsigned long h(const string& S) {  
2     string s(S);  
3  
4     while (s.size() % 3) s.push_back(0);  
5  
6     unsigned long v = 0, N = s.size(), M = N/3;  
7  
8     for (size_t i = 0; i < M; ++i)  
9         v ^= (s[i] ^ s[2*M - 1 - i] ^ s[2*M + i]);  
10  
11     return v;  
12 }
```

Meio quadrado

Chave	Um inteiro K
Algoritmo	Eleva-se K ao quadrado e toma-se a parte central do resultado
Nível de colisão	Médio

```
1 unsigned long h(int K)
2 {
3     auto s = K*K;
4
5     return (s & 0x00FFFF00) >> 8;
6 }
```

Polinomial

Chave	Uma string K com N caracteres
Algoritmo	$g(x)$ é um polinômio de grau $N - 1$ com coeficientes $a_i = K[i]$ e $h(K) = g(p) \pmod{T}$, onde T é o tamanho da tabela e $p \neq T$ é primo
Nível de colisão	Baixo

```
1 unsigned long h(const string& K, size_t p, size_t T)
2 {
3     unsigned long h = 0;
4
5     for (int i = K.size() - 1; i >= 0; --i)
6     {
7         h = (h * p) % T;
8         h = (h + K[i]) % T;
9     }
10
11     return h;
12 }
```

Endereçamento aberto

Problemas com a colisão

- Como visto, as funções de *hash* podem gerar colisões, isto é, um mesmo índice para duas chaves distintas
- Naturalmente surge o seguinte questionamento: como inserir duas chaves que colidem em uma mesma tabela, e como resgatá-las em uma busca?
- Uma alternativa para o tratamento de colisões é o endereçamento aberto

Definição

Se a chave K for mapeada para uma posição já ocupada da tabela, o endereçamento aberto utiliza a sequência de sondagem

$$N(h(K) + p(1)), N(h(K) + p(2)), \dots, N(h(K) + p(i)), \dots$$

onde p é a função de sondagem, i é o índice de sondagem e N a função de normalização, até que

1. se encontre uma posição desocupada
2. $N(h(K) + p(j)) = N(h(K))$
3. se verifique que a tabela está cheia

Sondagem linear

- Na sondagem linear, temos a função de sondagem é a identidade, isto é, $p(i) = i, \forall i$
- A função de normalização faz com que o índice resultante esteja dentro dos limites da tabela, usando o resto da divisão:

$$N(K) = K(\text{mod } T),$$

onde T é o tamanho da tabela

- Se uma posição $N(h(K) + p(i))$ já estiver ocupada, tenta-se o próximo índice de sondagem $(i + 1)$ até que se encontre um espaço vago ou ocorra uma das outras condições
- Esta estratégia tende a formação de agrupamentos de chaves, com pontos de acumulação na tabela e intervalos contíguos não ocupados

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 51

0	1	2	3	4	5	6	7	8	9	10

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 51

$$h(51) = 51 \pmod{11} = 7$$

0	1	2	3	4	5	6	7	8	9	10
							51			

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 16

0	1	2	3	4	5	6	7	8	9	10
							51			

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 16

$$h(16) = 16 \pmod{11} = 5$$

0	1	2	3	4	5	6	7	8	9	10
					16		51			

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 76

0	1	2	3	4	5	6	7	8	9	10
					16		51			

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 76

$$h(76) = 76 \pmod{11} = 10$$

0	1	2	3	4	5	6	7	8	9	10
					16		51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 35

0	1	2	3	4	5	6	7	8	9	10
					16		51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 35

$$h(35) = 35 \pmod{11} = 2$$

0	1	2	3	4	5	6	7	8	9	10
		35			16		51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: -6

0	1	2	3	4	5	6	7	8	9	10
		35			16		51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: -6

$$h(-6) = -6 \pmod{11} = 5$$

0	1	2	3	4	5	6	7	8	9	10
		35			16		51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: -6

$$N(h(-6) + 1) = (5 + 1) \pmod{11} = 6$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 49

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 49

$$h(49) = 49 \pmod{11} = 5$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 49

$$N(h(49) + 1) = (5 + 1) \pmod{11} = 6$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 49

$$N(h(49) + 2) = (5 + 2) \pmod{11} = 7$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem linear

Sondagem linear, $T = 11$

Elemento a ser inserido: 49

$$N(h(49) + 3) = (5 + 3) \pmod{11} = 8$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51	49		76

Implementação da sondagem linear

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 template<typename I, size_t T>
6 class HashSet {
7 private:
8     size_t mod(const I& a, int b)
9     {
10         return ((a % b) + b) % b;
11     }
12
13     size_t h(const I& K) { return mod(K, T); }
14
15     size_t N(const I& K, size_t i) { return mod(h(K) + i, T); }
16
17     vector<I> xs;
18     bitset<T> used;
19
20 public:
21     HashSet() : xs(T) {}
```

Implementação da sondagem linear

```
22
23  bool insert(const I& K)
24  {
25      if (used.count() == T)
26          return false;
27
28      for (size_t i = 0; i < T; ++i)
29      {
30          auto pos = N(K, i);
31
32          if (not used[pos])
33          {
34              xs[pos] = K;
35              used[pos] = true;
36              break;
37          }
38      }
39
40      return true;
41  }
```

Sondagem quadrática

- Na sondagem quadrática, a função de sondagem é dada por

$$p(i) = h(K) + (-1)^{i-1} \left\lfloor \frac{i+1}{2} \right\rfloor^2,$$

para $i = 1, 2, \dots, T-1$

- A função de normalização é dada por $N(K) = K \pmod{T}$, onde T é o tamanho da tabela
- A sondagem quadrática pode ser interpretada como a sequência

$$h(K) + i^2, h(K) - i^2, h(K) + (i+1)^2, h(K) - (i+1)^2, \dots$$

para $i = 1, 2, \dots, (T-1)/2$

- Se T for um número primo da forma $4k+3$, a sequência acima passa por todas as posições da tabela (Radke, 1970)

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 51

0	1	2	3	4	5	6	7	8	9	10

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 51

$$h(51) = 51 \pmod{11} = 7$$

0	1	2	3	4	5	6	7	8	9	10
							51			

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 16

0	1	2	3	4	5	6	7	8	9	10
							51			

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 16

$$h(16) = 16 \pmod{11} = 5$$

0	1	2	3	4	5	6	7	8	9	10
					16		51			

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 76

0	1	2	3	4	5	6	7	8	9	10
					16		51			

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 76

$$h(76) = 76 \pmod{11} = 10$$

0	1	2	3	4	5	6	7	8	9	10
					16		51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 35

0	1	2	3	4	5	6	7	8	9	10
					16		51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 35

$$h(35) = 35 \pmod{11} = 2$$

0	1	2	3	4	5	6	7	8	9	10
		35			16		51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: -6

0	1	2	3	4	5	6	7	8	9	10
		35			16		51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: -6

$$h(-6) = -6 \pmod{11} = 5$$

0	1	2	3	4	5	6	7	8	9	10
		35			16		51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: -6

$$N(h(-6) + 1^2) = (5 + 1) \pmod{11} = 6$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 49

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 49

$$h(49) = 49 \pmod{11} = 5$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 49

$$N(h(49) + 1^2) = (5 + 1) \pmod{11} = 6$$

0	1	2	3	4	5	6	7	8	9	10
		35			16	-6	51			76

Exemplo de inserção utilizando sondagem quadrática

Sondagem quadrática, $T = 11$

Elemento a ser inserido: 49

$$N(h(49) - 1^2) = (5 - 1) \pmod{11} = 4$$

0	1	2	3	4	5	6	7	8	9	10
		35		49	16	-6	51			76

Implementação da sondagem quadrática

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 template<typename I, size_t T>
6 class HashSet {
7 private:
8     size_t mod(const I& a, int b) { return ((a % b) + b) % b; }
9
10    size_t h(const I& K) { return mod(K, T); }
11
12    size_t N(const I& K, size_t i)
13    {
14        auto sign = i % 2 ? 1 : -1;
15        auto j = (i + 1)/2;
16
17        return mod(h(K) + sign * j * j, T);
18    }
19
20    vector<I> xs;
21    bitset<T> used;
```

Implementação da sondagem quadrática

```
23 public:
24     HashSet() : xs(T) {}
25
26     bool insert(const I& K)
27     {
28         if (used.count() == T) return false;
29
30         for (size_t i = 0; i < T; ++i)
31         {
32             auto pos = N(K, i);
33
34             if (not used[pos])
35             {
36                 xs[pos] = K;
37                 used[pos] = true;
38                 break;
39             }
40         }
41
42         return true;
43     }
```

1. **DROZDEK**, Adam. *Algoritmos e Estruturas de Dados em C++*, 2002.
2. **KERNIGHAN**, Bryan; **RITCHIE**, Dennis. *The C Programming Language*, 1978.
3. **RADKE**, Charles E. *The Use of Quadratic Residue Research*, Communications of the ACM, volume 13, issue 2, pg 103–105, 1970¹.
4. **STROUSTROUP**, Bjarne. *The C++ Programming Language*, 2013.
5. C++ Reference².

¹<https://dl.acm.org/citation.cfm?id=362036>

²<https://en.cppreference.com/w/>