Espalhamento (hashing)

MC 202 EF

Fernando Vanini

IC - Unicamp

Klais Soluções

Busca por elementos

- Dado um conjunto de pares (chave,valor)
  - determinar se uma chave está no conjunto e o valor associado
  - inserir um novo par no conjunto
  - remover um par do conjunto
- Estruturas que podem ser usadas
  - Tabelas simples (vetores ou listas)
  - Árvores de busca
  - Tabelas de espalhamento

Busca binária

- A tabela é um vetor ordenado.
- A cada passo o algoritmo divide o espaço de busca pela metade com base numa comparação.
- Tempo de busca por um elemento: $O(\log_2 N)$
- Tempo de inserção de um elemento: $O(N)$
- Tempo de remoção de um elemento: $O(N)$

Busca binária

```
/*************************************
   versão recursiva
*************************************/
int binSearch(int a, int tab[],int i,int j){
  if(i > j) return -1;
  int k = (i+j)/2;
  if(a < tab[k]) return binSearch(a,tab,i,k-1);
  if(a > tab[k]) return binSearch(a,tab,k+1,j);
  return k;
}
```

Busca binária

```
/*****************************************
   versão iterativa
*****************************************/
int binSearch(int a, int tab[],int i,int j){
  while (i >= j) {
    k = (i+j)/2;
    if(a < tab[k]) j = k-1; else
    if(a > tab[k]) i = k+1; else
    return k;
  }
  return -1;
}
```

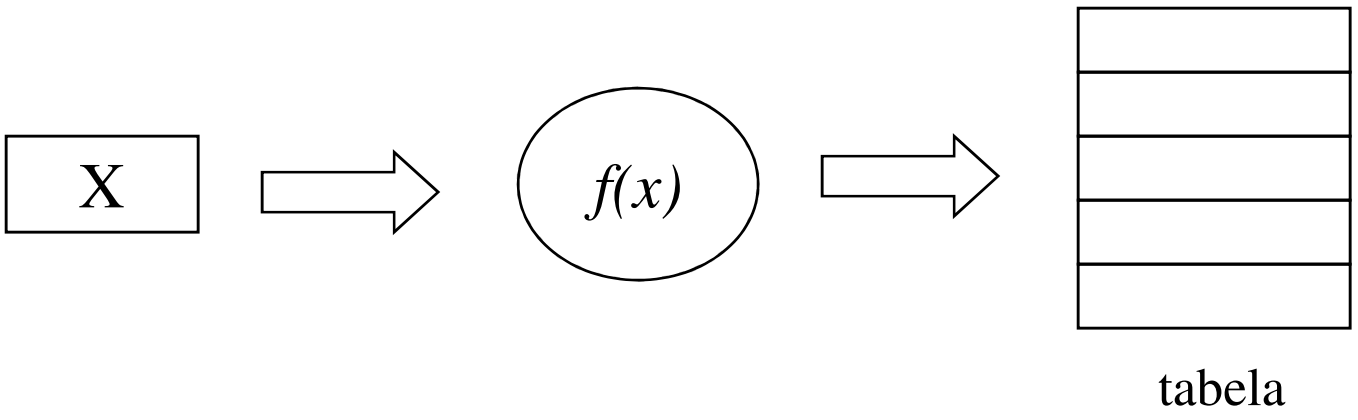
Estrutura geral do algoritmo

```
função busca(...){
  enquanto não achou {
      compare com uma chave da estrutura ;
      se achou devolva o par (chave,valor) encontrado;
      senão divida o espaço de busca com base no resultado
            da comparação;
  }
}
```

- esse processo é praticamente o mesmo em todos métodos de busca.
- se a divisão do espaço de busca for balanceada, o tempo de busca é O(log N)

Espalhamento (hashing)

- Idéia geral: usar uma função *f(x)* que associa as chaves do espaço de busca a um conjunto de inteiros entre 0 e k-1, correspondente ao índice numa tabela contendo os valores sendo procurados.



Espalhamento (*hashing*)

- Principais problemas
  - função de espalhamento adequada
  - que fazer em caso de colisão (chaves *x* e *y* tais que *f(x) = f(y)*) ?
- Função de espalhamento: características desejáveis:
  - eficiência
  - bom espalhamento

Um exemplo

```
/*********************************
  função de espalhamento exemplo
  devolve um inteiro entre 0 e 50
*********************************/
int f(char *s){
  int k = strlen(s);
  int i, j = 0;
  for(i=0; i<k; i++) j+=s[i];
  return j % 51;
}
```

Tratamento de colisões

• *resizing*: se a taxa de ocupação da tabela excede um limite, aumenta-se o tamanho da tabela.

•encadeamento: manter numa lista ligada as chaves que levam a um mesmo índice na tabela de *hashing*.

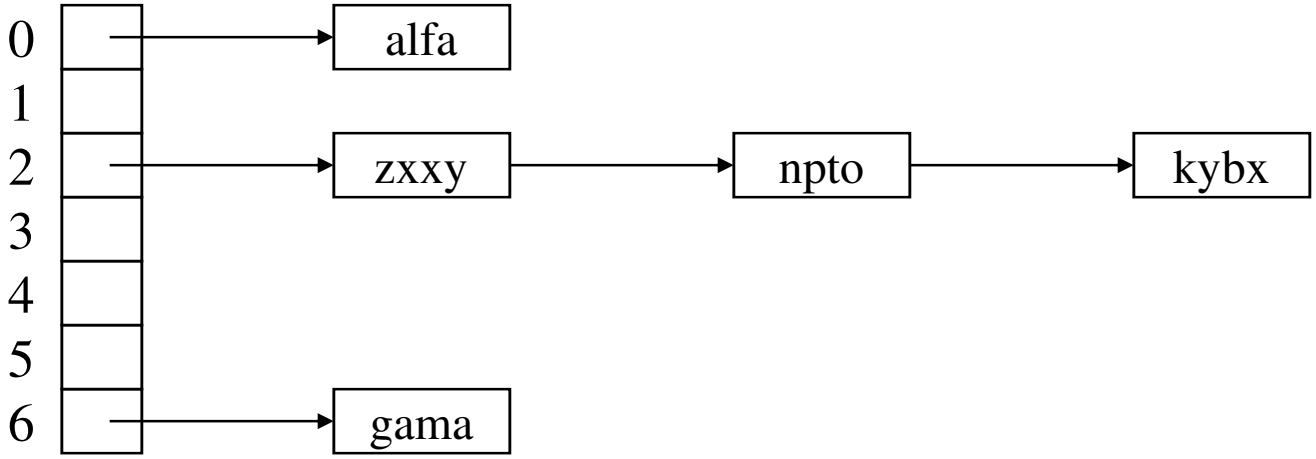


tabela de *hashing*

Hashing: vantagens e desvantagens

- Vantagens:
  - algoritmos simples e eficientes para inserção, retirada e busca
- Desvantagens:
  - nenhuma garantia de balanceamento
  - espaço sub-utilizado nas tabelas
  - o grau de espalhamento é sensível à função de *hashing* utilizada e ao tipo de informação usada como chave.

Referências na web

- http://en.wikipedia.org/wiki/Hash_table
- http://www.sparknotes.com/cs/searching/hashtables/section1.html
- http://www.cs.auckland.ac.nz/software/AlgAnim/hash_tables.html
- http://www.cs.pitt.edu/~kirk/cs1501/animations/Hashing.html

Espalhamento (hashing)

MC 202 EF

Fernando Vanini

IC - Unicamp

Klais Soluções

Busca por elementos

- Dado um conjunto de pares (chave,valor)
  - determinar se uma chave está no conjunto e o valor associado
  - inserir um novo par no conjunto
  - remover um par do conjunto
- Estruturas que podem ser usadas
  - Tabelas simples (vetores ou listas)
  - Árvores de busca
  - Tabelas de espalhamento

Busca binária

- A tabela é um vetor ordenado.
- A cada passo o algoritmo divide o espaço de busca pela metade com base numa comparação.
- Tempo de busca por um elemento: $O(\log_2 N)$
- Tempo de inserção de um elemento: $O(N)$
- Tempo de remoção de um elemento: $O(N)$

Busca binária

```
/*************************************
   versão recursiva
*************************************/
int binSearch(int a, int tab[],int i,int j){
  if(i > j) return -1;
  int k = (i+j)/2;
  if(a < tab[k]) return binSearch(a,tab,i,k-1);
  if(a > tab[k]) return binSearch(a,tab,k+1,j);
  return k;
}
```

Busca binária

```
/*****************************************
   versão iterativa
*****************************************/
int binSearch(int a, int tab[],int i,int j){
  while (i >= j) {
    k = (i+j)/2;
    if(a < tab[k]) j = k-1; else
    if(a > tab[k]) i = k+1; else
    return k;
  }
  return -1;
}
```

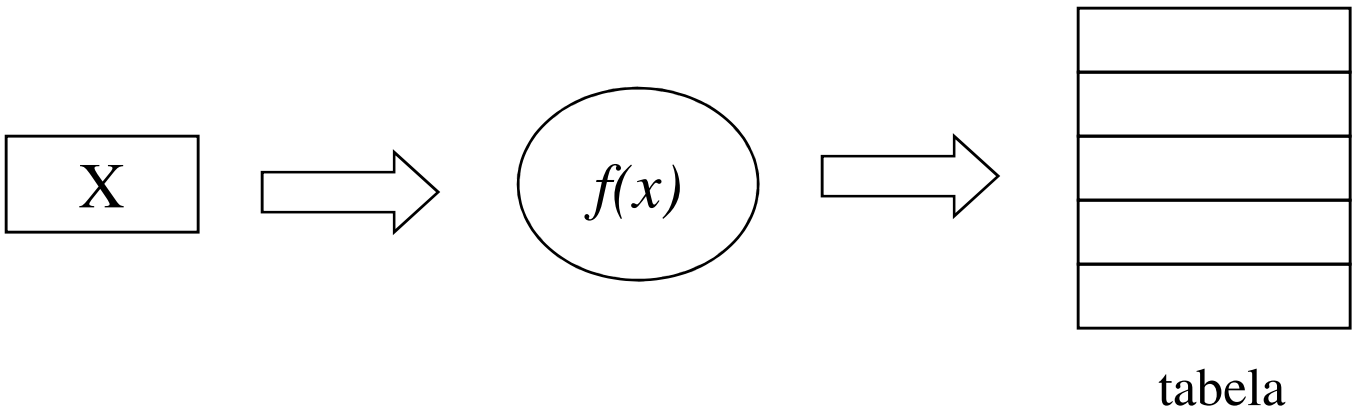
Estrutura geral do algoritmo

```
função busca(...){
  enquanto não achou {
      compare com uma chave da estrutura ;
      se achou devolva o par (chave,valor) encontrado;
      senão divida o espaço de busca com base no resultado
            da comparação;
  }
}
```

- esse processo é praticamente o mesmo em todos métodos de busca.
- se a divisão do espaço de busca for balanceada, o tempo de busca é O(log N)

Espalhamento (hashing)

- Idéia geral: usar uma função *f(x)* que associa as chaves do espaço de busca a um conjunto de inteiros entre 0 e k-1, correspondente ao índice numa tabela contendo os valores sendo procurados.



Espalhamento (*hashing*)

- Principais problemas
  - função de espalhamento adequada
  - que fazer em caso de colisão (chaves *x* e *y* tais que *f(x) = f(y)*) ?
- Função de espalhamento: características desejáveis:
  - eficiência
  - bom espalhamento

Um exemplo

```
/*********************************
  função de espalhamento exemplo
  devolve um inteiro entre 0 e 50
*********************************/
int f(char *s){
  int k = strlen(s);
  int i, j = 0;
  for(i=0; i<k; i++) j+=s[i];
  return j % 51;
}
```

Tratamento de colisões

• *resizing*: se a taxa de ocupação da tabela excede um limite, aumenta-se o tamanho da tabela.

•encadeamento: manter numa lista ligada as chaves que levam a um mesmo índice na tabela de *hashing*.

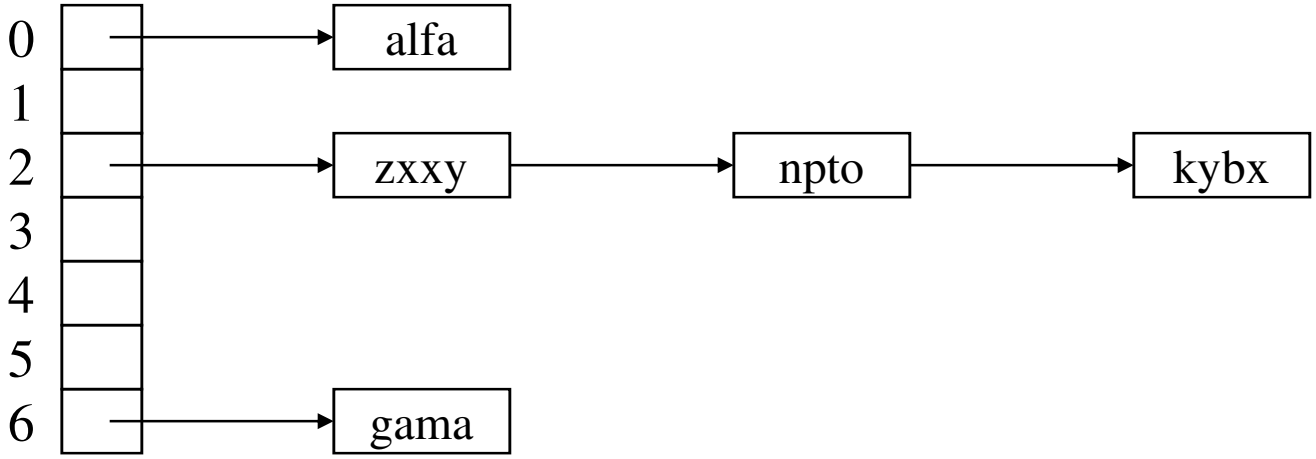


tabela de *hashing*

Hashing: vantagens e desvantagens

- Vantagens:
  - algoritmos simples e eficientes para inserção, retirada e busca
- Desvantagens:
  - nenhuma garantia de balanceamento
  - espaço sub-utilizado nas tabelas
  - o grau de espalhamento é sensível à função de *hashing* utilizada e ao tipo de informação usada como chave.

Referências na web

- http://en.wikipedia.org/wiki/Hash_table
- http://www.sparknotes.com/cs/searching/hashtables/section1.html
- http://www.cs.auckland.ac.nz/software/AlgAnim/hash_tables.html
- http://www.cs.pitt.edu/~kirk/cs1501/animations/Hashing.html