Estrutura de Dados Avançada

Daniel de Sousa Moraes
danielmoraes14@gmail.com

Tabelas Hash (Espalhamento)

- Algumas aplicações precisam de um conjunto dinâmico que suportam apenas operações de de dicionário:
  - INSERT
  - SEARCH
  - DELETE

Tabelas Hash (Espalhamento)

- Ex: Um compilador que traduz uma linguagem de programação possui uma tabela de símbolos, onde as chaves dos elementos são cadeias de caracteres correspondentes aos identificadores da linguagem.

Tabelas Hash (Espalhamento)

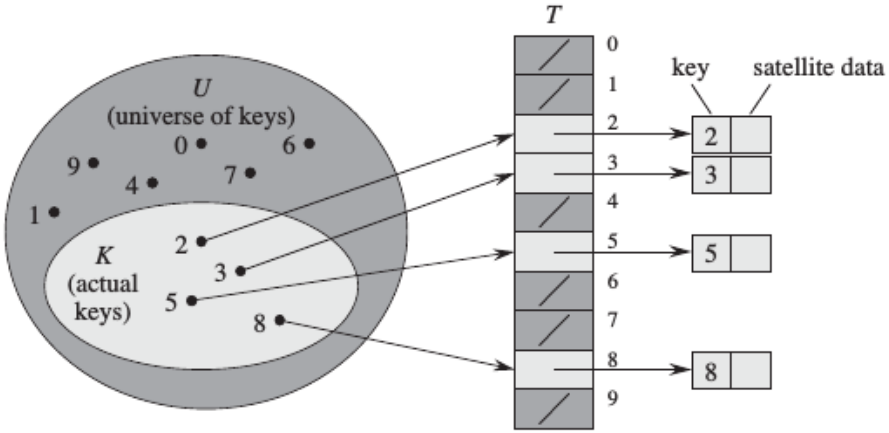
- Tabela Hash
  - É uma generalização da noção de um vetor simples.
  - É como um vetor cujas chaves **não** são apenas inteiros e os valores **não** estão necessariamente consecutivos na memória
  - É uma estrutura eficaz para implementação de dicionários

Tabelas de Acesso direto

- Acesso direto é uma técnica simples que funciona bem no seguinte universo:
  - Considere um universo *U* de *chaves,* razoavelmente pequeno
  - Considere uma aplicação que precise de um conjunto dinâmico onde cada elemento possui uma *chave k* do universo $K = \{k_1, k_2, \ldots, k_m\}$ onde *m* não é tão grande
  - Para cada chave usada do universo *U*, pode haver um único respectivo dado satélite (elemento), ou seja, dois elementos não podem ter a mesma chave.

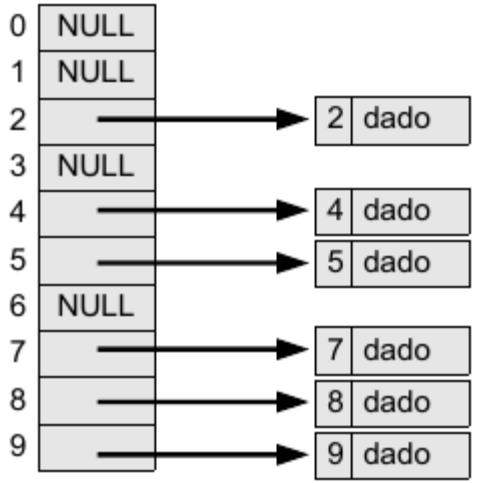
Tabelas de Acesso direto

- Para representar o conjunto dinâmico, usamos um vetor, ou **tabela de acesso direto**, representada por *T [0… m* -1], onde cada posição (ou *slot*) corresponde a uma única *chave k* do universo *U*.



Tabelas de Acesso direto

- *Slot k* aponta para um elemento no conjunto com a chave *k*.
- Se não há elementos no conjunto com a chave *k*, então *T[k] = NULL;*



Tabelas de Acesso direto

- Primitivas
  - Insert(T , x)

    T[key[x]] = x
  - Search (T, k)

    return T [k]
  - Delete (T, x)

    T[key[x]] = null

Tabelas de Acesso direto

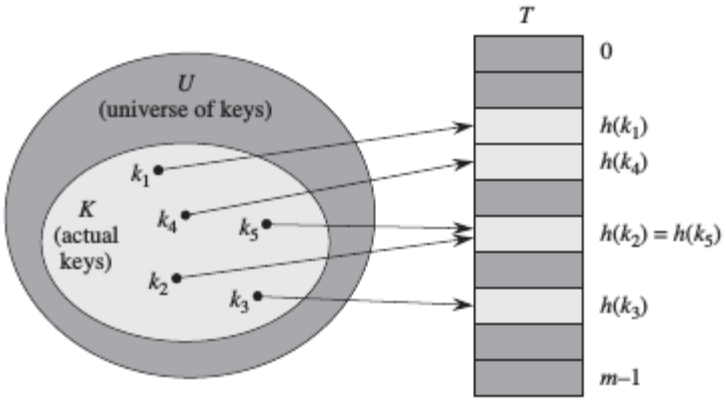
- Problemas
  - Se o universo U for muito grande, armazenar uma tabela T de tamanho $|U|$ poder ser impráticavel e até impossível dada a memória de um computador típico.
    - Suponha as chaves sendo os nomes de pessoas
  - E se $|K|$ for pequeno e $|U|$ for grande, a maioria dos espaços alocados por T seriam inutéis.

Tabelas Hash

- Quando o conjunto de chaves utilizadas, *K,* é bem menor que o universo *U* de todas as possíveis chaves, uma **tabela hash** é mais eficaz pois demanda menos espaço de armazenamento que uma tabela de acesso direto

Tabelas Hash

- No acesso direto, um elemento de chave *k* é armazenado no *slot k*. Na hash, esse elemento é armazenado no *slot h(k)*, ou seja, usa-se uma **função de hash (dispersão) *h*** para mapeiar o *slot* a partir da chave *k*.



Tabelas Hash

- Podem haver **colisões**
  - Duas chaves podem ser mapeadas pro mesmo *slot*

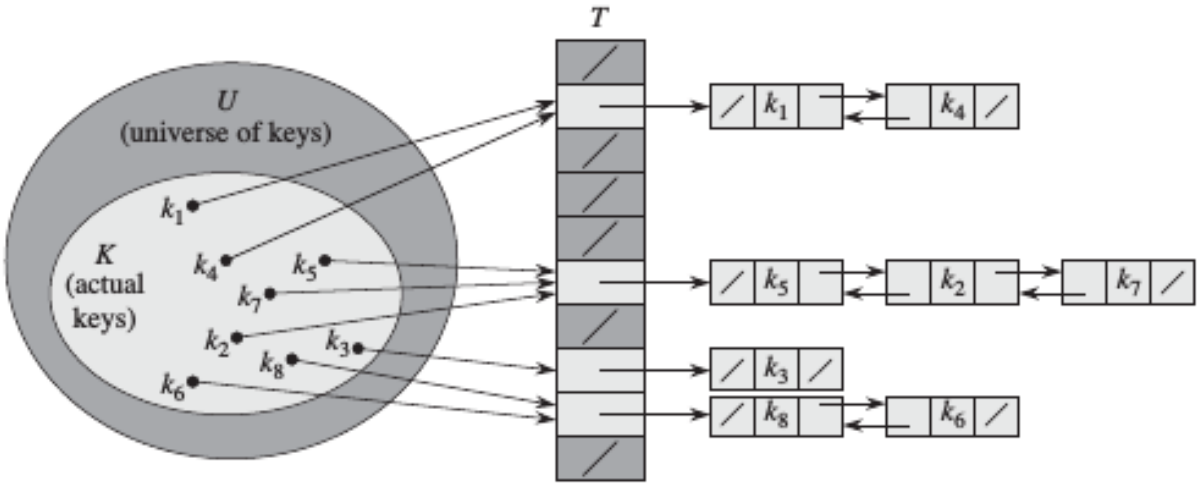
Tabelas Hash

- Formas de evitar colisões
  - Evitar completamente tendo uma função *h* aleatória
    - Mas como |U| > m, sempre devem haver pelo menos duas chaves com mesmo valor hash. Evitar completamente as colisões é impossível.

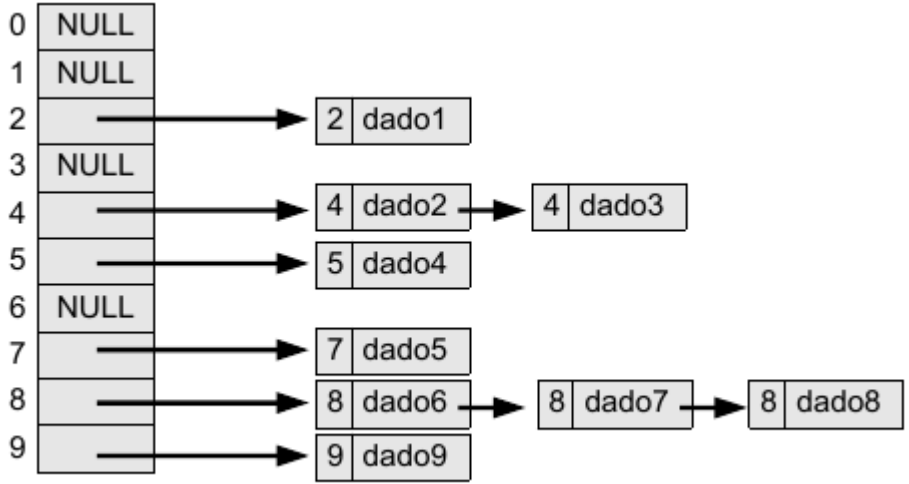
Evitando colisões com listas encadeadas

- Usa-se listas encadeadas para posicionar pacotes com mapeiados para o mesmo *slot*
- O *slot j* contêm um ponteiro para a cabeça da lista com todos os elementos mapeiados para o hash *j*.
- Se não houver tais elementos, *slots i* contêm NULL

Tabelas hash com Listas



Tabelas hash com Listas



Evitando colisões com listas encadeadas

- Primitivas
  - insert(T, x) insere na cabeça da lista T[h (ḱey[k])]
  - search(T, k) procura x na lista T T[h (ḱey[k])]
  - delete(T, x) deleta x da lista T[h (ḱey[k])]

Bibliografia

Cormen, Thomas H. et al. Algoritmos.; [tradução Arlete Simille]. 3ª ed - Rio de Janeiro - Elsevier, 2011.

Carlos de Salles Soares Neto – Notas de Aula da Disciplina de ED II - UFMA