Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

SIN 343
Desafios de Programação

João Batista Ribeiro

joao42lbatista@gmail.com

Slides baseados no material do prof. Guilherme C. Pena

Universidade Federal de Viçosa
Campus Rio Paranaíba
Instituto de Ciências Exatas e Tecnológicas

Aula de Hoje

Estruturas de Dados Elementares

Estruturas de Dados Elementares

Listas Encadeadas:

Em uma lista encadeada, os objetos estão organizados em uma ordem linear.

Diferentemente de um arranjo onde a ordem é dada pelos índices, na lista encadeada a ordem é determinada por um ponteiro em cada objeto.

A lista possui um atributo ***L.head*** que aponta para o primeiro elemento da lista. Caso ***L.head*** **== NULL**, a lista está vazia.

Estruturas de Dados Elementares

Listas Encadeadas (Variantes):

- Lista Simplesmente Encadeda

Onde um elemento **x** possui além da ***key***, um ponteiro para o ***next*** elemento da lista.

- Lista Duplamente Encadeda

Onde um elemento **x** possui além da ***key***, dois ponteiros, um para o ***next*** elemento da lista e um para o elemento ***prev***.

Estruturas de Dados Elementares

Listas Encadeadas (Variantes):

- Lista Circular

No caso das variantes anteriores a lista pode ser ***circular*** da seguinte forma:

-Na lista simplesmente encadeada, o ponteiro ***next*** do último elemento aponta para o primeiro elemento.

-Na lista duplamente encadeada, o ponteiro **next** do último elemento aponta para o primeiro elemento, e o ponteiro ***prev*** do primeiro elemento aponta para o último elemento.

Estruturas de Dados Elementares

Listas Encadeadas (Variantes):

- Lista Ordenada

No caso das listas encadeadas serem ***ordenadas***, a ordem linear corresponde à ordem linear de ***chaves*** armazenadas em elementos da lista, de forma crescente ou decrescente.

No caso de listas ***não-ordenadas*** os elementos podem aparecer em qualquer ordem.

Estruturas de Dados Elementares

Vamos supor uma implementação de uma ***lista simplesmente encadeada não-ordenada:***

List-Search(L, k)

```
1   x = L.head
2   while x != NULL e x.key != k
3      x = x.next
4   return x
```

Estruturas de Dados Elementares

Vamos supor uma implementação de uma ***lista simplesmente encadeada não-ordenada:***

List-Insert(L, x) Dado um elemento **x** cujo atributo **chave** já foi definido*:*

```
1   x.next = L.head
2   L.head = x
```

Neste caso Insere x sempre no início da lista.

Estruturas de Dados Elementares

Vamos supor uma implementação de uma ***lista simplesmente encadeada não-ordenada:***

List-Delete(L, k) Dado uma **chave** k*:*

```
1   x = List-Search(L, k)
2   if x != NULL
3      if x == L.head
4         L.head = x.proximo
5      else
6         y = L.head
7         while y.proximo != x
8            y = y.proximo
9         y.proximo = x.proximo
```

Estruturas de Dados Elementares

Listas (C++):

#include <list> std::list

C++ traz uma implementação de uma **lista duplamente encadeada.**

http://www.cplusplus.com/reference/list/list/

Estruturas de Dados Elementares

Listas (C++):

#include <forward_list> std::forward_list

C++ também traz uma implementação de uma **lista simplesmente encadeada.**

http://www.cplusplus.com/reference/forward_list/forward_list/

Estruturas de Dados Elementares

Listas (C++):

```
#include<iostream>
#include<list>
using namespace std;
int main() {
    list<int> lista;
    list<int>::iterator it;
    for(int i=1; i<=5; ++i)
        lista.push_back(i); // 1 2 3 4 5
    cout << "Lista contem:";
    for(it=lista.begin(); it!=lista.end(); it++)
        cout << ' ' << *it;
    cout << '\n';
    return 0;
}
```

Estruturas de Dados Elementares

Listas: (Problema do URI)

URI Online Judge | 1119

A Fila de Desempregados

Em resumo, trata-se de uma lista circular onde os elementos vão sendo retirados a cada vez, seguindo algumas regras próprias do problema.

https://www.urionlinejudge.com.br/judge/pt/problems/view/1119

Estruturas de Dados Elementares

Vectors (C++):

#include <vector> std::vector

http://www.cplusplus.com/reference/vector/vector/

Vectors são estruturas implementadas no C++ que representam um arranjo dinâmico, que pode alterar de tamanho. No entando, sempre de forma **contígua** na memória.

A manipulação de tamanho é feita de forma **automática** pela estrutura usando alocação dinâmica.

Estruturas de Dados Elementares

Vectors (C++):

A operação de inserção de um novo elemento em arrays exige alocar um novo array e mover todos os elementos antigos para o novo espaço.

Esta é uma operação muito cara em termos de eficiência e os ***vectors*** usam uma estratégia de sempre deixar um espaço maior do que o necessário na inserção de novos elementos.

Estruturas de Dados Elementares

Vectors (C++):

Portanto, comparados com arrays, os ***vectors*** consomem **mais memória** em troca da capacidade de gerenciar o **armazenamento** e o **crescimento dinâmico** de uma forma **mais eficiente**.

Comparando com outras estruturas de sequência (decks, listas), **vectors** são **muito eficientes** ao **acessar elementos** e relativamente eficientes adicionando ou removendo **elementos do fim**. Já para fazer o mesmo com **outras posições** eles **não são tão eficientes**.

Estruturas de Dados Elementares

Pairs (C++): #include <utility>

A classe ***Pair*** une um par de valores, que podem ser de tipos iguais ou diferentes. Os valores individuais podem ser acessados através dos membros públicos ***first*** e ***second***.

Dependendo da aplicação, ***pairs*** são usados juntos com estruturas: ***vectors, maps, sets***..

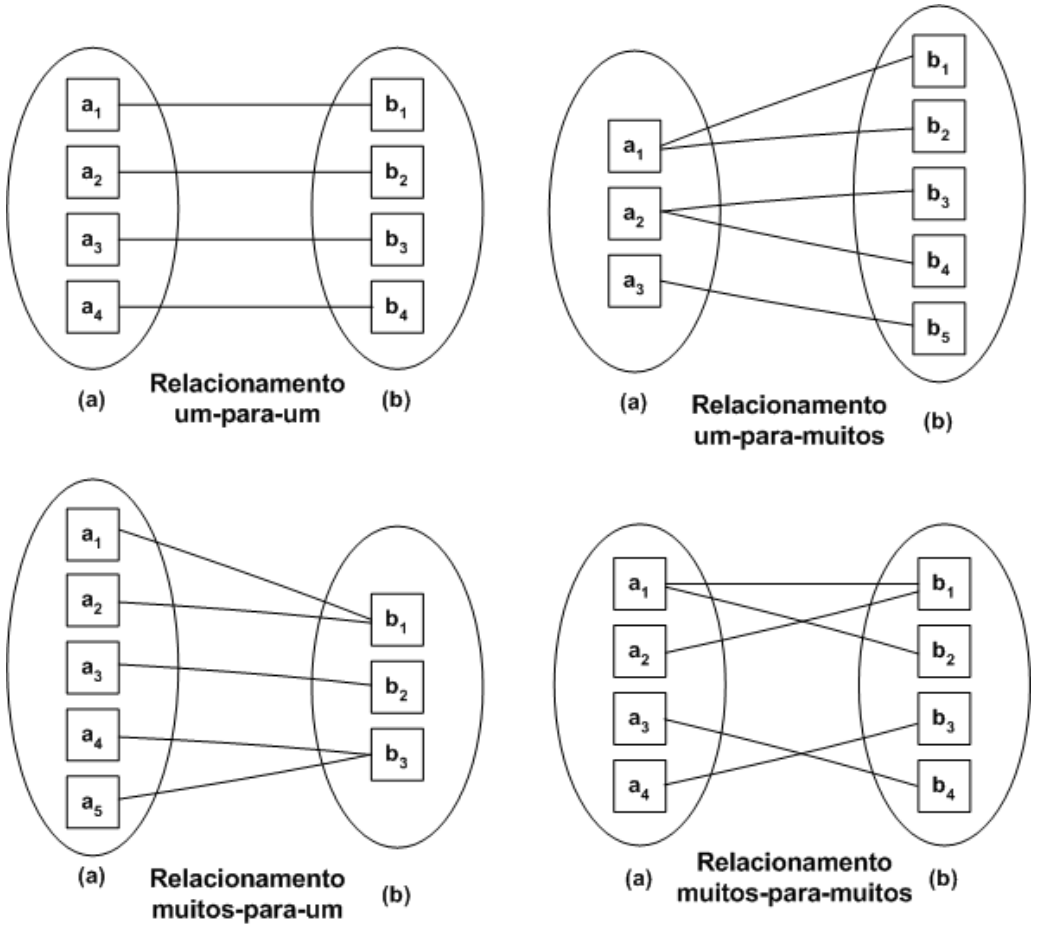
```
#include <utility>
#include <iostream>
using namespace std;
int main () {
  pair <int,int> x, y;
  x = make_pair(10,20);
  y = x;
  cout << y.first << ” “ << y.second << '\n';
  return 0;
}
```

http://www.cplusplus.com/reference/utility/pair/

Estruturas de Dados Elementares

Maps (C++): #include <map>

Maps são estruturas associativas que armazenam elementos formados por uma combinação de chave e um valor mapeado, seguindo uma ordem específica



http://www.cplusplus.com/reference/map/map/

Estruturas de Dados Elementares

Maps (C++): #include <map>

No ***map***, as chaves são usadas para ordenar e identificar de forma única os elementos, enquanto que os valores mapeados guardam o conteúdo associado àquela chave.

A ordenação das chaves do ***map*** é feita de forma interna. E os valores mapeados podem ser acessados diretamente pelo operador **[]**.

Estruturas de Dados Elementares

Maps (C++): #include <map>

Maps são tipicamente implementados como árvores binárias.

E nas implementações de C++ (11), existem também algumas variações da estrutura:

- ***multimap***
- **unordered_map**
- ***unordered_multimap***

Estruturas de Dados Elementares

Maps (C++): #include <map>

```
int main() {
    map<int,vector<int> > mapa;
    map<int,vector<int> >::iterator it;
    vector<int>::iterator itv;
    vector<int> vet, vet2;
    for(int i=0; i<10; i++)
        vet.push_back(i);
    mapa.insert(pair<int, vector<int> >(0, vet));
    vet2.push_back(2);
    vet2.push_back(10);
    mapa[1] = vet2;
    cout << "map contains:" << endl;
    for(it = mapa.begin(); it != mapa.end(); it++) {
        cout << ' ' << it->first << " => ";// << it->second[0] << endl;
        for(itv=it->second.begin(); itv!=it->second.end(); itv++)
            cout << " " << *itv;
        cout << endl;
    }
    cout << '\n';
    return 0;
}
```

```
0 =>  0 1 2 3 4 5 6 7 8 9
1 =>  2 10
```

Estruturas de Dados Elementares

Sets (C++): #include <set>

Sets são estruturas que armazenam elementos **únicos**, seguindo uma ordem específica. Elementos repetidos não serão inseridos.

No ***set***, o valor de um elemento o identifica e cada valor deve ser único. Os valores dos elementos não podem ser modificados dentro da estrutura (são sempre ***const***), no entanto podem ser inseridos ou removidos.

http://www.cplusplus.com/reference/set/set/

Estruturas de Dados Elementares

Sets (C++): #include <set>

Sets também são tipicamente implementados como árvores binárias.

E nas implementações de C++ (11), existem também algumas variações da estrutura:

- *multiset*

- unordered_set

- *unordered_multiset*

Estruturas de Dados Elementares

Sets (C++): #include <set>

```
#include <iostream>
#include <set>
using namespace std;
int main() {
    set<int> conjunto;
    set<int>::iterator it;
    for(int i=0; i<10; i++)
        conjunto.insert(i%2);
    cout << "set contains:" << endl;
    for(it=conjunto.begin(); it!=conjunto.end(); it++)
        cout << ' ' << *it;
    cout << '\n';
    return 0;
}
```

```
set contains:
 0 1
```

Estruturas de Dados Elementares

Listas: (Problema do URI)

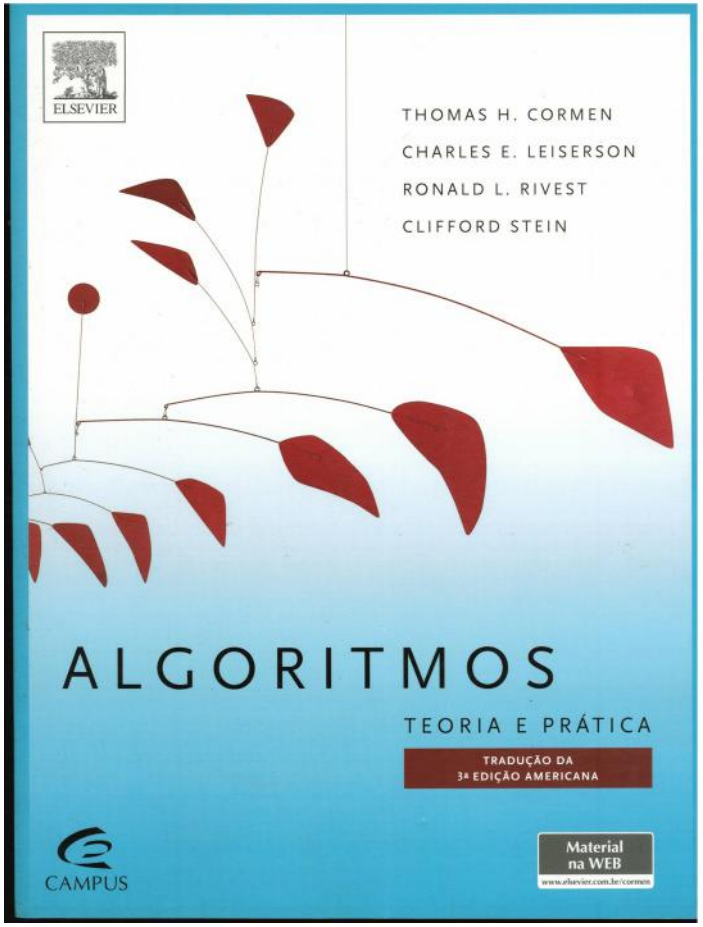
URI Online Judge | 1256
Tabelas Hash

Esse problema basicamente é a implementação de uma tabela hash contendo chaves e valores inteiros.

https://www.urionlinejudge.com.br/judge/pt/problems/view/1256

Referência Bibliográfica

- CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: teoria e prática. Tradução da 2. ed. Americana. Rio de Janeiro: Campus, 2002.



Referência Bibliográfica

- SKIENA, S.S. REVILLA, M. A. Programming challenges: the programming contest training manual. Springer, 2003.