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

Aula de Hoje

Estruturas de Dados Elementares

Estruturas de Dados Elementares

Conjuntos (coleções de dados) são fundamentais para a Matemática e para a Ciência da Computação.

Enquanto conjuntos matemáticos são invariáveis, os **conjuntos manipulados por algoritmos** podem **crescer**, **encolher** ou **sofrer outras mudanças** ao longo do tempo.

Tais conjuntos podem ser chamados de ***conjuntos dinâmicos***.

Estruturas de Dados Elementares

Em uma implementação típica de um conjunto dinâmico, cada elemento é representado por um objeto cujos atributos podem ser examinados e manipulados através de um ponteiro.

Alguns tipos de conjuntos dinâmicos consideram que um dos atributos do objeto é uma ***chave*** de identificação.

Números, letras, palavras, são exemplos de chave que podem definir uma ordenação aos elementos do conjunto.

Estruturas de Dados Elementares

Operações:

As operações em conjuntos dinâmicos podem ser agrupadas em duas categorias:

- ***consultas***, que simplesmente retornam informações sobre o conjunto;

- ***operações modificadoras***, que alteram o conjunto.

Estruturas de Dados Elementares

Operações:

- Search(S, k)

Uma consulta que, dado um conjunto S e um valor de chave k, retorna um ponteiro para um elemento de S tal que x.chave = k ou NULL se nenhum elemento desse tipo for encontrado.

- Insert(S, x)

Uma operação modificadora que aumenta o conjunto S com o elemento apontado por x. Normalmente considera-se que os atributos de x já foram inicializados.

Estruturas de Dados Elementares

Operações:

- Delete(S, x)

Uma operação modificadora que, dado um ponteiro x para um elemento no conjunto S, remove x de S.

- Minimum(S)

Uma consulta em um conjunto totalmente ordenado S que retorna um ponteiro para o elemento de S que tenha a menor chave.

- Maximum(S)

Uma consulta em um conjunto totalmente ordenado S que retorna um ponteiro para o elemento de S que tenha a maior chave.

Estruturas de Dados Elementares

Operações:

- Sucessor(S, x)

Uma consulta que, dado um elemento x cuja chave é de um conjunto totalmente ordenado S, retorna um ponteiro para o elemento maior seguinte em S ou NULL se x é o elemento máximo.

- Predecessor(S, x)

Uma consulta que, dado um elemento x cuja chave é de um conjunto totalmente ordenado S, retorna um ponteiro para o elemento menor seguinte em S ou NULL se x é o elemento mínimo.

Estruturas de Dados Elementares

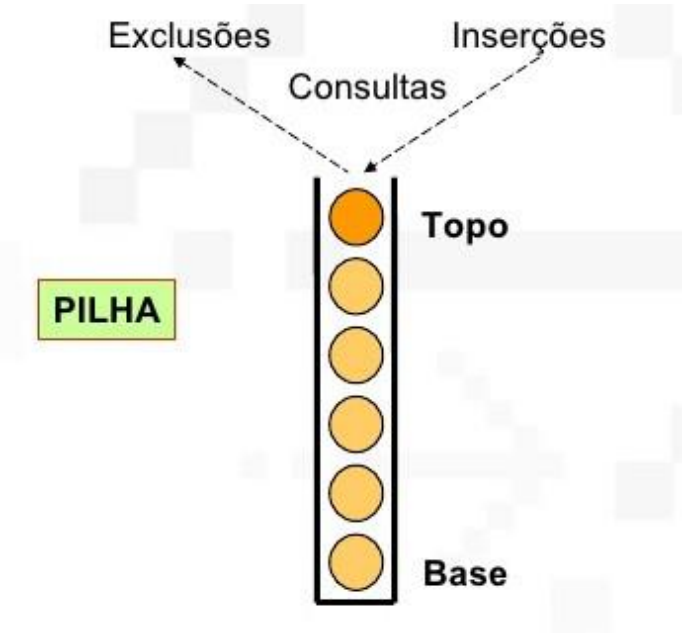
Pilhas e Filas:

Pilhas e Filas são conjuntos dinâmicos nos quais o elemento removido do conjunto pela operação ***Delete*** é especificado previamente.

Estruturas de Dados Elementares

Pilhas e Filas:

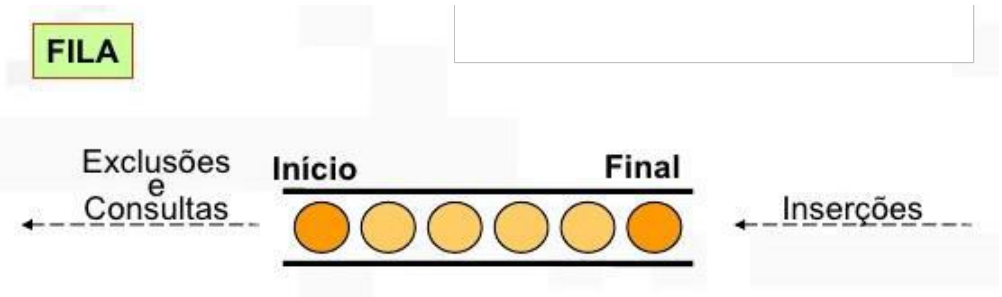
Em uma ***pilha***, o elemento eliminado do conjunto é o mais recentemente inserido: a pilha implementa uma política de ***último a entrar, primeiro a sair*** ou ***LIFO*** (last-in, first-out).



Estruturas de Dados Elementares

Pilhas e Filas:

Em uma ***fila***, o elemento eliminado é sempre o que estava no conjunto a mais tempo: a fila implementa uma política de ***primeiro a entrar, primeiro a sair*** ou ***FIFO*** (first-in, first-out).



Estruturas de Dados Elementares

Pilhas:

A operação ***Insert*** em uma pilha é frequentemente denominada ***Push***, e a operação ***Delete*** é frequentemente denominada ***Pop***.

Podemos implementar uma pilha de no máximo n elementos com um arranjo simples **S[1 .. n]**.

O arranjo tem um atributo ***S.top*** que indexa o elemento mais recentemente inserido.

Estruturas de Dados Elementares

Pilhas:

A pilha consiste nos elementos **S[1 .. S.top]**, onde **S[1]** é o elemento na parte inferior da pilha e **S[S.top]** é o elemento na parte superior.

Quando S.topo = 0, a pilha não contém nenhum elemento e está ***vazia***.

Estruturas de Dados Elementares

Pilhas:

Stack-Empty(S)

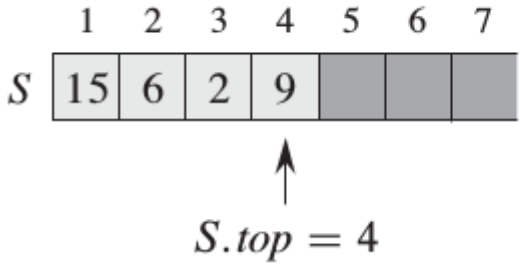
```
1  if S.topo == 0
2     return true
3  else return false
```

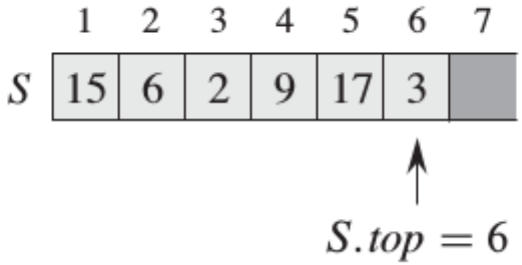
Push(S, x)

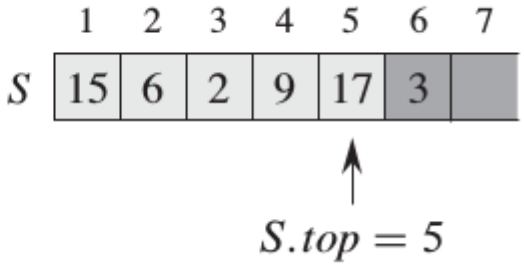
```
1  S.topo = S.top + 1
2  S[S.top] = x
```

Pop(S)

```
1  if Stack-Empty(S)
2     error "underflow"
3  else
4     S.top = S.top -1
5     return S[S.top+1]
```







Estruturas de Dados Elementares

Pilhas (C++):

#include <stack>
std::stack

Funções Principais (existem outras):

(constructor)	Constroi Pilha
empty	Testa se está vazia
size	Retorna o tamanho
top	Consulta o topo
push	Insere um elemento
pop	Retorna o topo

Estruturas de Dados Elementares

Pilhas (C++):

```
#include <iostream>          // std::cout
#include <stack>             // std::stack

using namespace std;

int main (){
   stack<int> pilha;
   for (int i=0; i<5; ++i)
       pilha.push(i);

   cout << "Removendo elementos...";
   while (!pilha.empty()) {
      cout << ' ' << pilha.top();
      pilha.pop();
   }

   cout << '\n';
   return 0;
}
```

Estruturas de Dados Elementares

Pilhas: (Balanceamento de Parenteses)

Em uma string, você deve avaliar se os elementos parênteses, colchetes e chaves estão balanceados. Diz que uma string é balanceada quando cada elemento que abre tem seu elemento equivalente que fecha corretamente.

Por exemplo, a string: **“(()[]{})”** está balanceada;
Enquanto as strings: **“(()([])” “(()())”** e **“){}(“** não estão.

3 **{[()]}** **{[(])}** **{{[[()]]}}**	**YES** **NO** **YES**

Estruturas de Dados Elementares

Pilhas: (Balanceamento de Parenteses)

https://www.hackerrank.com/challenges/balanced-brackets

Estruturas de Dados Elementares

Filas:

A operação ***Insert*** em uma fila é frequentemente denominada ***Enqueue***, e a operação ***Delete*** é frequentemente denominada ***Dequeue***.

A fila funciona como uma fileira de pessoas em uma caixa registradora. A fila tem um início (ou ***head***) e um fim (ou ***tail***). Quando um elemento é inserido na fila, ocupa seu lugar no fim dela.

Estruturas de Dados Elementares

Filas:

O elemento retirado da fila é sempre aquele que está no início dela.

Podemos implementar uma pilha de no máximo n-1 elementos com um arranjo simples **Q[1 .. n]**.

A fila tem um atributo ***Q.head*** que indexa ou aponta para seu início. O atributo ***Q.tail*** que indexa a próxima posição onde um elemento recém-chegado ocupará.

Estruturas de Dados Elementares

Filas:

Os elementos de uma fila seguem uma ordem circular, isto é, quando ***Q.tail ou Q.head == Q.length*** eles passam para a primeira posição novamente.

Quando ***Q.head == Q.tail***, a fila está vazia.

Inicialmente ***Q.head == Q.tail == 1*** e não existem elementos para desenfileirar.

Quando ***Q.head == Q.tail + 1*** ou simultaneamente ***Q.head == 1 e Q.tail == Q.length***, a fila está cheia e não podemos inserir mais elementos.

Estruturas de Dados Elementares

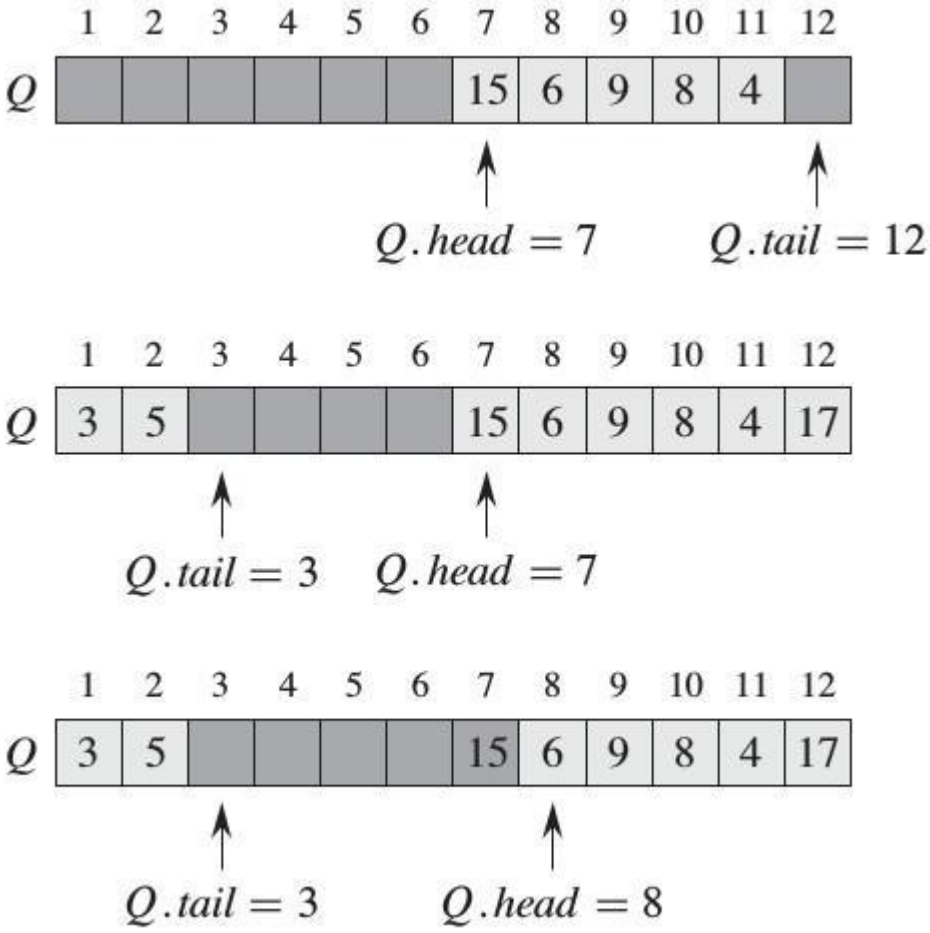
Filas:

Enqueue(Q,x)

```
1   Q[Q.tail] = x
2   if Q.tail == Q.length
3      Q.tail = 1
4   else Q.tail = Q.tail + 1
```

Dequeue(Q)

```
1   x = Q[Q.head]
2   if Q.head == Q.length
3      Q.head = 1
4   else Q.head = Q.head + 1
5   return x
```



Estruturas de Dados Elementares

Filas (C++):

#include <queue>
std::queue

Funções Principais (existem outras):

(constructor)	Constroi Pilha
empty	Testa se está vazia
size	Retorna o tamanho
front	Acessa o primeiro elemento atual
back	Acessa o ultimo elemento atual
push	Insere um elemento
pop	Remove o primeiro elemento atual

Estruturas de Dados Elementares

Filas (C++):

```
#include <iostream>
#include <queue>
using namespace std;
int main (){
  queue<int> fila;
  int num;
  cout << "Digite valores(0 - sair):\n";
  do {
    cin >> num;
    fila.push (num);
  } while (num);
  cout << "Valores: ";

  while (!fila.empty()){
    cout << ' ' << fila.front();
    fila.pop();
  }
  cout << '\n';
  return 0;
}
```

Estruturas de Dados Elementares

Filas (Distancia entre cidades):

A ideia de fila aparece naturalmente no cálculo de distâncias em um grafo. Imagine N cidades numeradas de 0 a N-1 e interligadas por estradas de mão única. As ligações entre as cidades são representadas por uma matriz A da seguinte maneira:

A[i][j] vale 1 se existe estrada de i para j

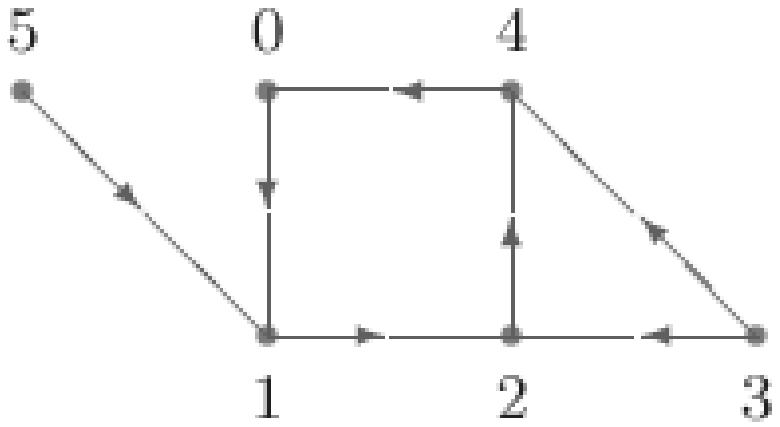
e vale 0 em caso contrário.

Estruturas de Dados Elementares

Filas (Distancia entre cidades):

Segue um exemplo em que N vale 6:

	0	1	2	3	4	5
0	0	1	0	0	0	0
1	0	0	1	0	0	0
2	0	0	0	0	1	0
3	0	0	1	0	1	0
4	1	0	0	0	0	0
5	0	1	0	0	0	0



Exemplo de entrada:

6 3
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 0 1 0
0 0 1 0 1 0
1 0 0 0 0 0
0 1 0 0 0 0

Saída

2 3 1 0 1 6

Estruturas de Dados Elementares

Filas (Distancia entre cidades):

A distância de uma cidade c a uma cidade j é o **menor número de estradas** que preciso percorrer para ir de c a j. (A distância de c a j é, em geral, diferente da distância de j a c.)

O problema:

Dada uma cidade c e a matriz de estradas, determinar a distância de c a cada uma das demais cidades.

Exemplo de entrada:

```
6 4
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 0 1 0
0 0 1 0 1 0
1 0 0 0 0 0
0 1 0 0 0 0
```

Saída:

```
1 2 3 6 0 6
```

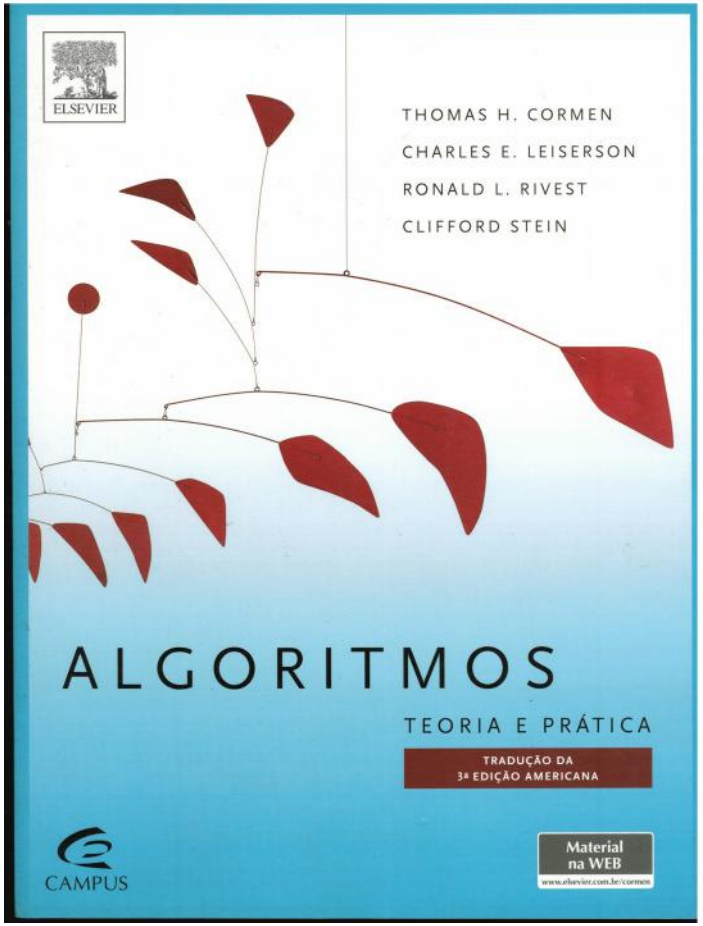
Estruturas de Dados Elementares

Filas (Problema do castelo na matriz):

https://www.hackerrank.com/challenges/castle-on-the-grid

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.