

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

7

utad
Pedro Melo-Pinto 2022

os dados podem organizar-se de diferentes formas que correspondem a diferentes conceitos de agrupamento; embora alguns destes conceitos tenham surgido em disciplinas como a matemática ou a álgebra, interessam-nos somente o ponto de vista dos sistemas de computação.

Algoritmia (e Estruturas de Dados)

ED00

ESTRUTURAS DE DADOS

DADOS E ABSTRACÇÃO

8

utad
Pedro Melo-Pinto 2022

An abstract orange background featuring a complex network of black lines and dots, resembling a digital or mathematical landscape. The lines form a series of peaks and valleys, creating a sense of depth and structure.

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

9

utad
Pedro Melo-Pinto 2022

PARADIGMA. ABSTRACÇÃO

"uma abstracção é uma descrição ou uma especificação simplificada de uma entidade que dá ênfase a certas propriedades dessa entidade e ignora outras"

João Pavão Martins, 2012. Programação em Python

funções

separa as propriedades lógicas de uma accão dos detalhes da sua implementação
separa o modo como uma função deve ser utilizada, da forma com ela é desenvolvida e implementada

dados

separa as propriedades lógicas dos dados dos detalhes da sua representação
separa o modo como os dados são utilizados, da forma com eles são definidos e construídos

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

10

utad
Pedro Melo-Pinto 2022

PARADIGMA. ABSTRACÇÃO

A abstracção de dados permite-nos lidar com tipos de dados mais complexos, mantendo a independência entre a sua definição e a sua representação

A especificação de um Tipo Abstracto de Dados conterá um conjunto de dados e de operações sobre esses dados

As abstrações procedimentais e as abstrações de dados estão fortemente ligadas: as operações de um Tipo Abstrato de Dados são abstrações procedimentais.

Um tipo abstrato de dados engloba ambas – o conjunto de operações é definido para qualquer tipo de dados que possa assumir um conjunto de valores.

PARADIGMA. TIPOS ABSTRACTOS DE DADOS

Na utilização ... (de) tipos abstractos de dados devem ser seguidos quatro passos sequenciais:

- 1 a identificação das operações básicas;
- 2 a axiomatização das operações básicas;
- 3 a escolha de uma representação para os elementos do tipo; e
- 4 a concretização das operações básicas para a representação escolhida.

João Pavão Martins, 2012. Programação em Python

11

utad

Pedro Melo-Pinto 2022

As estruturas de dados estabelecem distintas formas de relacionamento entre os dados.

Estas relações são habitualmente divididas em lineares ou não lineares.

os elementos estão organizados sequencialmente

cada elemento pode estar ligado a mais do que um outro elemento, de modo a reflectir uma relação entre eles

12

utad

Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

13

utad
Pedro Melo-Pinto 2022

RELACOES ENTRE OS DADOS

conjunto de dados	relação linear	relação não linear	relação não linear (mais geral)
	lista	árvore	grafo

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

14

utad
Pedro Melo-Pinto 2022

as estruturas de dados são implementações de colecções

Definition. As the term is used more precisely in computer science, a data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data.

Wegner P., Reilly, E.D., 2003
Data Structures in Encyclopedia of Computer Science, 507-512.

Definition. A collection is an object that serves as a repository for other objects. They may be implemented in a variety of ways (which is why they are often referred to as abstract data types - ADTs), determined by the data structure.

Smyth, T., 2007
Simon Fraser University, Canada



Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

15 utad Pedro Melo-Pinto 2022

CONCEITO TIPO ABSTRACTO DE DADOS

COLECCÃO

Grupo de dados de tamanho variável (0+ membros)

Os dados:

- partilham significado no âmbito do problema em causa
- são geralmente do mesmo tipo ou de tipos derivados
(em linguagens que suportem o conceito de derivação/hierarquia)

(o número de elementos de um *array* é fixo)

Um *array* não é habitualmente considerado uma coleção
mas desempenha um papel na implementação de coleções



Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

16 utad Pedro Melo-Pinto 2022

CONCEITO TIPO ABSTRACTO DE DADOS

COLECCÃO

Grupo de dados de tamanho variável (0+ membros)

algumas coleções permitem duplicados, outras não

algumas coleções são ordenadas, outras não

operações típicas: acrescentar ou remover um elemento, remover todos os elementos, encontrar um elemento (ou saber se ele pertence à coleção), obter o tamanho da coleção

a maioria das coleções são construídas para tipos de dados em particular ou tendo em conta operações específicas

exemplos de coleções:
conjuntos (*sets* e *multisets*), listas, árvores e grafos

17

Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADOS INTRO

CONCEITO TIPO ABSTRACTO DE DADOS

COLECCAO: CONJUNTO

Colecção de objectos diferentes
contém elementos não repetidos e sem uma ordem estabelecida

versão computacional do conceito matemático de conjunto finito
(conjunto com um número finito de elementos)

exemplo:
 $\{3, -2, 5, 0, 11\}$

Pedro Melo-Pinto 2022

utad

18

Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADOS INTRO

CONCEITO TIPO ABSTRACTO DE DADOS

COLECCAO: DICIONÁRIO

Colecção de chaves únicas + coleção de valores

- cada chave se encontra-se associada a um valor ou conjunto de valores
- correspondência unívoca
 - (um valor está associado a uma única chave, uma chave pode corresponder a diferentes valores)

a operação de encontrar o valor associado a uma chave é designado por indexação (*indexing* ou *lookup*)

fortemente relacionados com o conceito matemático de função de domínio finito (são utilizados nos processos de *memoization*)

Pedro Melo-Pinto 2022

utad

19

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

CONCEITO: TIPO ABSTRACTO DE DADOS

COLECCAO: LISTA

Colecção sequencial

- pode conter elementos repetidos
- os elementos são identificados por índices

representação computacional do conceito matemático de tuplo
sucessão ordenada finita (*tuple*)

exemplo:
 $\langle 3, -2, 5, 0, 11, 5 \rangle$

Pedro Melo-Pinto 2022

utad

20

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS INTRO

CONCEITO: TIPO ABSTRACTO DE DADOS

COLECCAO: LISTA

Colecção sequencial

- não quer dizer que os elementos estão ordenados pelo seu valor, mas sim que os elementos têm uma posição (sequencial) na lista

exemplo:
 $\langle 3, -2, 5, 0, 11, 5 \rangle \neq \langle 11, 5, 0, -2, 3, 5 \rangle$

Pedro Melo-Pinto 2022

utad



Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADOS LISTAS

TIPO ABSTRACTO DE DADOS

LISTA

Colecção sequencial

- pode conter elementos repetidos
- os elementos são identificados por índices

representação computacional do conceito matemático de tuplo
sucessão ordenada finita (*tuple*)

exemplo: $(3, -2, 5, 0, 11, 5)$

```
graph LR; A[3] --> B[-2]; B --> C[5]; C --> D[0]; D --> E[11]; E --> F[5]
```

nota

$(3, -2, 5, 0, 11, 5) \neq (11, 5, 0, -2, 3, 5)$

23

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS LISTAS

TIPO ABSTRACTO DE DADOS

LISTA COLEÇÃO SEQUENCIAL

definições:

- sequência dinâmica de dados $\langle a_0, a_1, \dots, a_{n-1} \rangle$ de tamanho n
pode mudar ao longo do tempo
- o primeiro elemento de uma lista é a_0 e o último é a_{n-1}
- o elemento a_{i+1} sucede ao elemento a_i (numa lista não vazia)
- o elemento a_i precede o elemento a_{i+1} (numa lista não vazia)
- não se define o elemento que precede a_0 , nem o elemento que sucede a a_{n-1}
- a posição do elemento a_i na lista é $i+1$ (sendo a posição inicial 0) ou i (se a posição inicial é 1)
- o comprimento/tamanho de uma lista indica quantos elementos tem a lista
- uma lista de tamanho $n = 0$ designa-se por lista vazia

Pedro Melo-Pinto 2022

utad

24

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS LISTAS

TIPO ABSTRACTO DE DADOS

LISTA COLEÇÃO SEQUENCIAL

características:

- cada elemento da lista deve ter (associado) algum tipo de dados
- geralmente todos os elementos da lista têm o mesmo tipo de dados
embora não exista objecção conceptual a listas cujos elementos tenham tipos de dados diferentes
- o início da lista (primeiro elemento) designa-se habitualmente por cabeça (*head*)
- o final da lista (último elemento) designa-se habitualmente por cauda (*tail*)

duas listas consideram-se iguais se os seus elementos (considerados sequencialmente) forem iguais

Pedro Melo-Pinto 2022

utad

25

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS LISTAS

TIPO ABSTRACTO DE DADOS

LISTA COLEÇÃO SEQUENCIAL

propriedades:

- uma lista pode ter zero ou mais elementos
- um novo elemento pode ser adicionado a uma lista em qualquer posição
- qualquer elemento de uma lista pode ser eliminado
- qualquer elemento de uma lista pode ser acedido
- uma lista pode ser percorrida (i.e., todos os elementos são acedidos, um de cada vez)

Pedro Melo-Pinto 2022

utad

26

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS LISTAS

TIPO ABSTRACTO DE DADOS

LISTA COLEÇÃO SEQUENCIAL

um tipo abstrato de dados tem associado um conjunto de operações “universais”
(operações que não dependem do contexto em que são usadas)

correspondem a quatro categorias de acções:

- Ⓐ operações | funções de criação (*creators*)
criam novos objectos deste TAD
- Ⓑ operações | funções de produção (*producers*)
criam novos objectos a partir de objectos existentes deste TAD
- Ⓒ operações | funções de acesso (*query; observers*)
obtêm informação sobre objectos deste TAD
- Ⓓ operações | funções de comando (*command; mutators*)
modificam instâncias deste TAD

Pedro Melo-Pinto 2022

utad

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS LISTAS

27

utad
Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA COLEÇÃO SEQUENCIAL

$a_0 \rightarrow a_1 \rightarrow \dots \rightarrow a_i \rightarrow \dots \rightarrow a_{n-2} \rightarrow a_{n-1}$

operações habituais:

- criação de uma lista (vazia)
- inserção de elemento(s)
- eliminação de elemento(s)
- percorrer a lista
- união de duas ou mais listas
- divisão de uma lista em duas ou mais
- ordenar os elementos de uma lista
- procurar determinado elemento numa lista

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS LISTAS

28

utad
Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA METODOLOGIAS DE IMPLEMENTAÇÃO

1 arrays

VANTAGENS	
Acesso directo a qualquer elemento. Facilidade em percorrer a lista.	
DESVANTAGENS	
Operação de inserção (ou remoção) lenta. <i>Uso ineficaz dos recursos de memória.</i> mesmo com alocação dinâmica	

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | ARRAYS

29

utad
Pedro Melo-Pinto 2022

lista L com $n = 9$ elementos

ENTRADA DE ELEMENTO NO INÍCIO DA LISTA

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | ARRAYS

30

utad
Pedro Melo-Pinto 2022

data newelement

i = n = 9

```
STATUS insertStartIntList(int *L, int n, int data)
{
    int i;
    for(i=n; i>=1; i--)
        L[i] = L[i-1];
    L[0] = data;
    return OK;
}
```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | ARRAYS

31 utad Pedro Melo-Pinto 2022

data newelement

```
i = 8
STATUS insertStartIntList(int *L, int n, int data)
{
    int i;
    for(i=n-1; i>=1; i--)
        L[i] = L[i-1];
    L[0] = data;
    return OK;
}
```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | ARRAYS

32 utad Pedro Melo-Pinto 2022

data newelement

```
i = 7 to 1
STATUS insertStartIntList(int *L, int n, int data)
{
    int i;
    for(i=n-1; i>=1; i--)
        L[i] = L[i-1];
    L[0] = data;
    return OK;
}
```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | ARRAYS

33

utad
Pedro Melo-Pinto 2022

```
STATUS insertStartIntList(int *L, int n, int data)
{
    int i;
    for(i=n-1; i>=1; i--)
        L[i] = L[i-1];
    L[0] = data;
    return OK;
}
```

Complexidade temporal: $O(n)$

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | ARRAYS

34

utad
Pedro Melo-Pinto 2022

```
STATUS insertEndIntList(int *L, int n, int data)
{
    L[n] = data;
    return OK;
}
```

Qual é a complexidade temporal de inserir um elemento no final da lista?

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS LISTAS

35 utad
Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA METODOLOGIAS DE IMPLEMENTAÇÃO

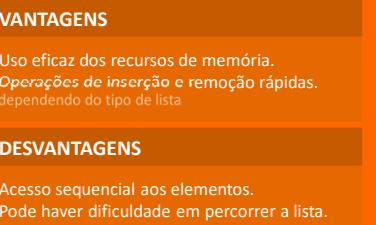
② listas encadeadas (*linked lists*)

VANTAGENS

Uso eficaz dos recursos de memória.
Operações de inserção e remoção rápidas.
dependendo do tipo de lista

DESVANTAGENS

Acesso sequencial aos elementos.
Pode haver dificuldade em percorrer a lista.



Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

36 utad
Pedro Melo-Pinto 2022

LISTA LISTAS ENCADEADAS

Uma lista encadeada é uma sequência de nós

A lista é accedida através de uma ligação para o início (*list head*).



Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Uma lista encadeada é uma sequência de nós

Cada nó contém :

- A os dados (podem ser de qualquer tipo)
- B uma ligação ao elemento seguinte

Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Uma lista encadeada é uma sequência de nós

O último nó aponta para NULL.

Uma lista encadeada pode conter, ou não, um elemento cabeçalho (*header node*).

This special node is usually used to store the number of nodes present in the linked list.

Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Uma lista encadeada é uma sequência de nós

```
graph LR; L((L)) --> N1[11]; N1 --> N2[22]; N2 --> N3[33]; N3 --> N4[44]; N4 --> N5[55]; N5 --> N6[66]; N6 --> N7[77]; N7 --> NULL((NULL)); T((T)) --> N7;
```

O último nó aponta para NULL.
Uma lista encadeada pode ter uma ligação para o último elemento.

Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Uma lista encadeada é uma sequência de nós

```
graph LR; L((L)) --> N1[11]; N1 --> N2[22]; N2 --> N3[33]; N3 --> N4[44]; N4 --> N5[55]; N5 --> N6[66]; N6 --> N7[77]; N7 --> NULL((NULL));
```

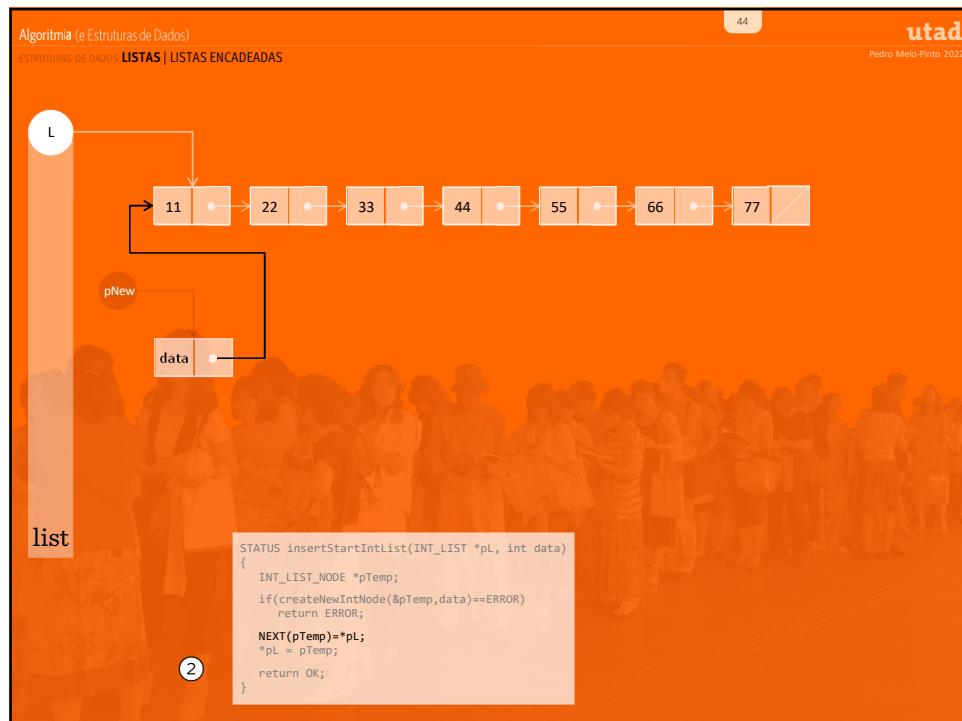
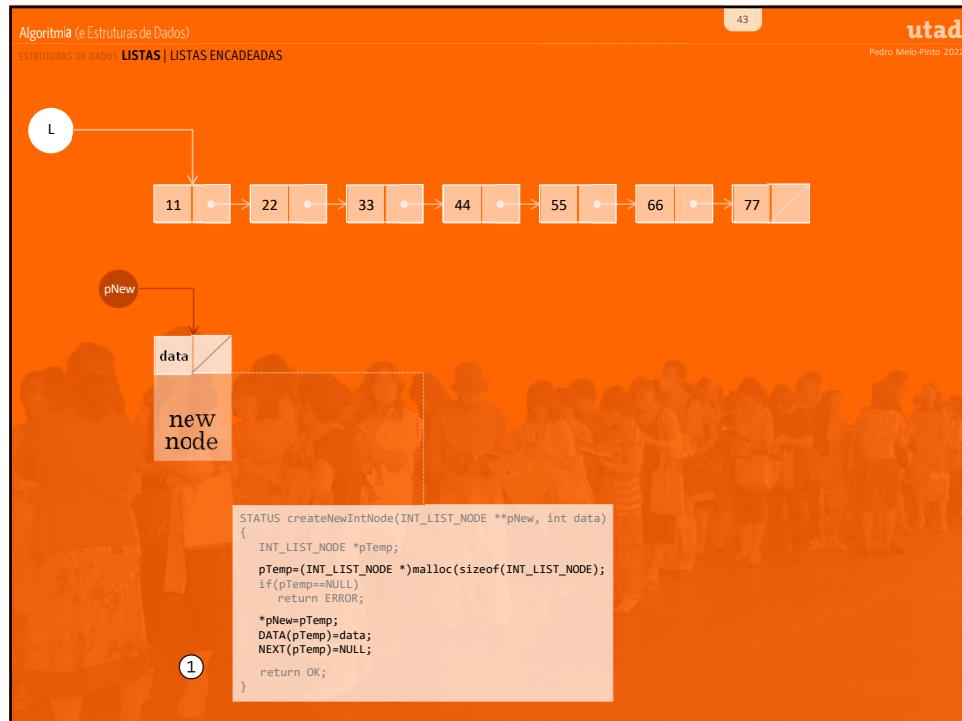
O **sucessor** de um nó é o próximo nó na sequência.
(o último nó não tem successor)

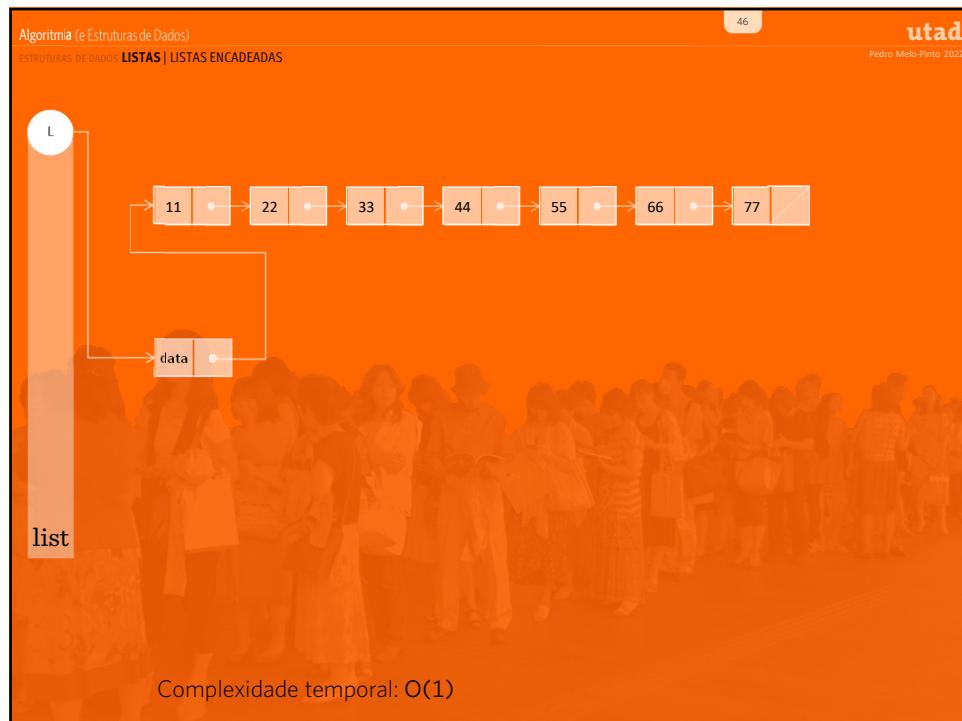
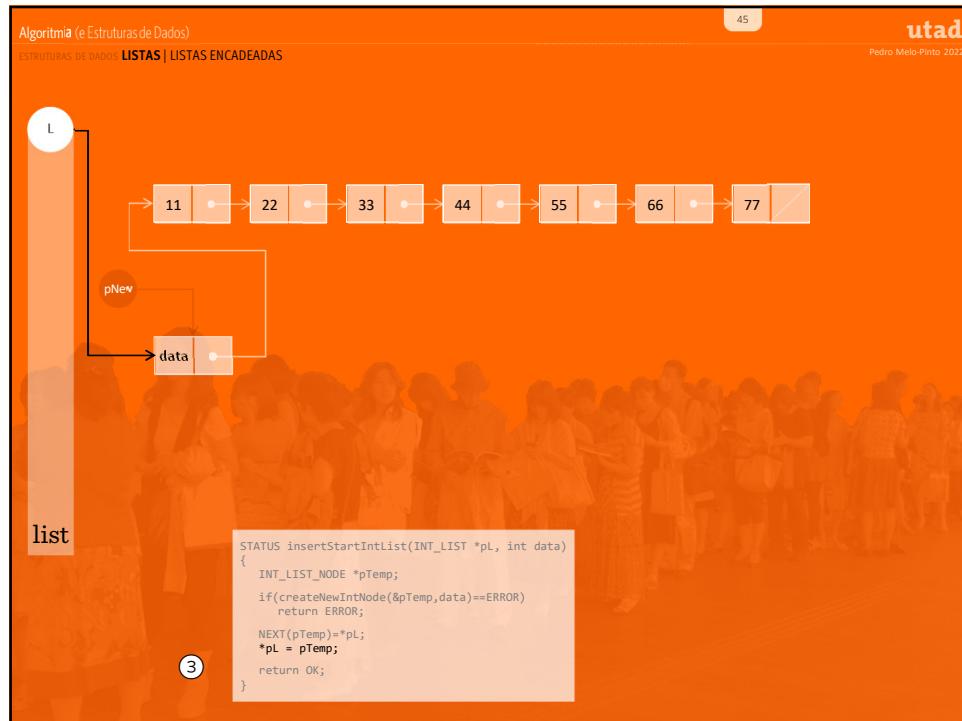
O **antecessor** de um nó é o nó anterior na sequência.
(o primeiro nó não tem antecessor)

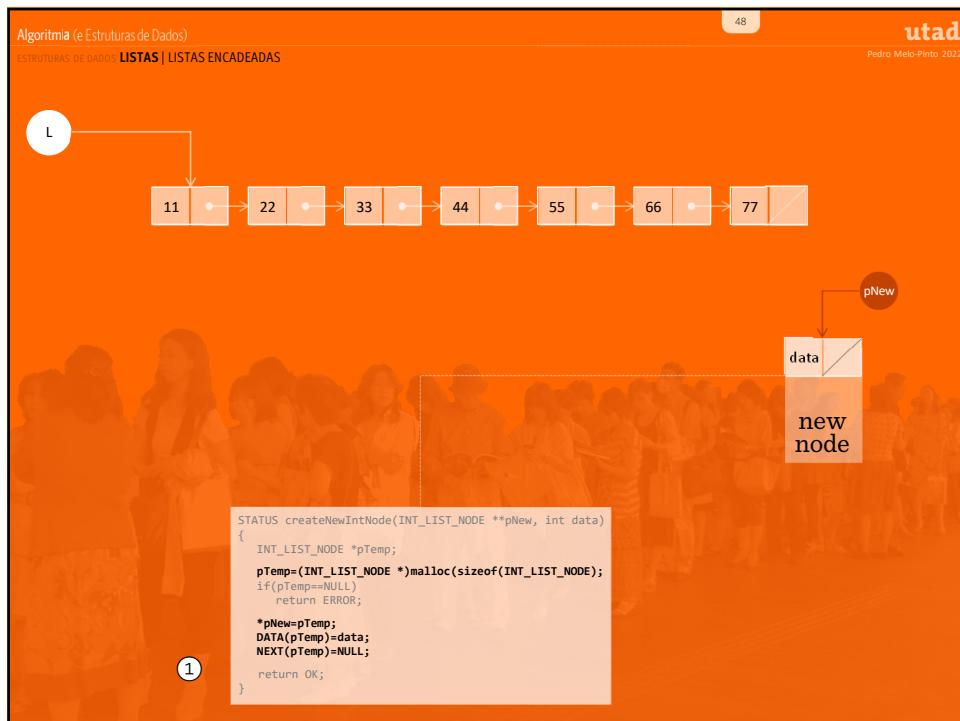
O comprimento de uma lista encadeada é o seu número de elementos.
(lista vazia se não tiver elementos)

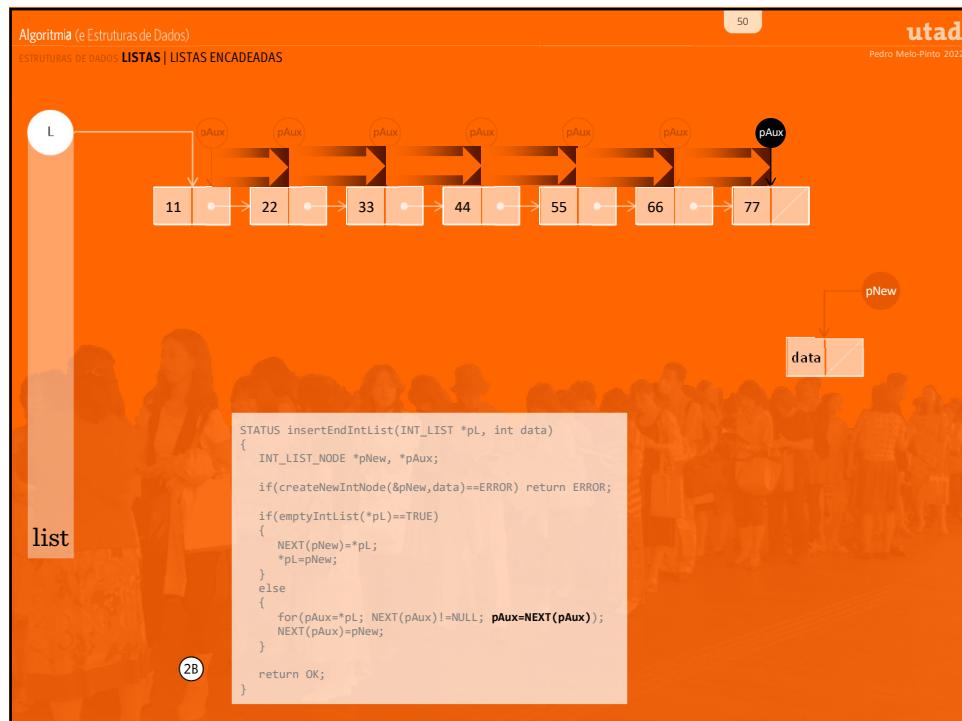
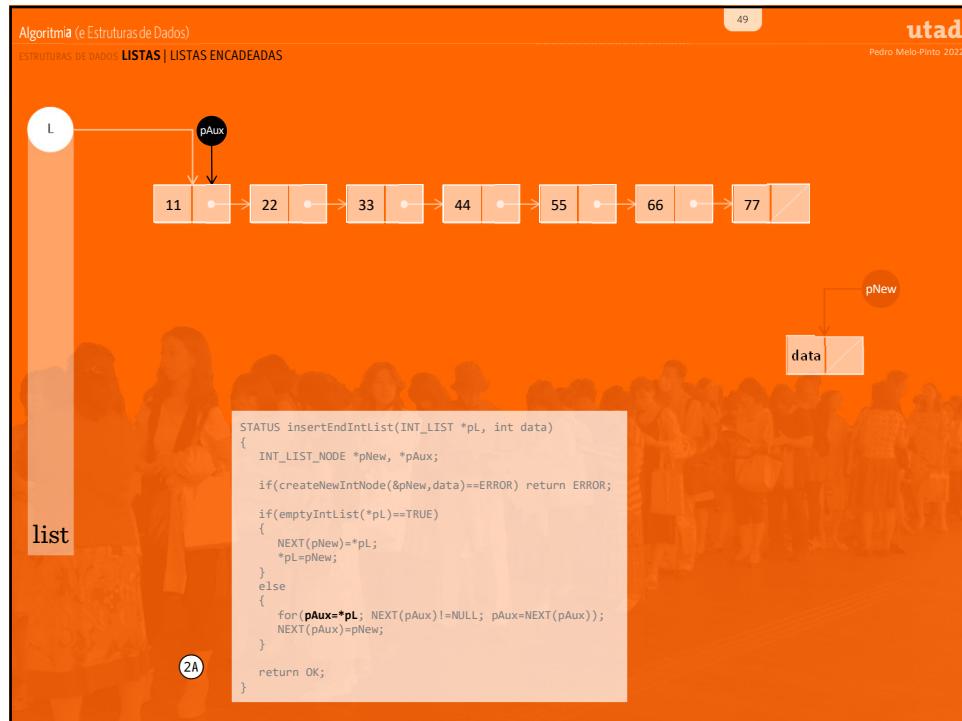
Pedro Melo-Pinto 2022

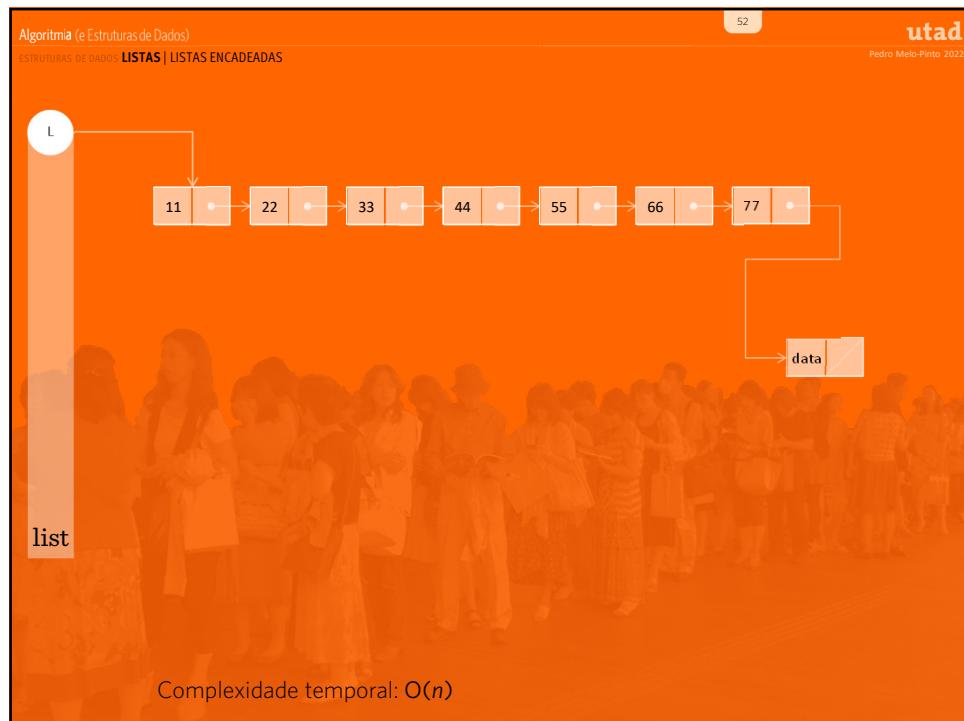
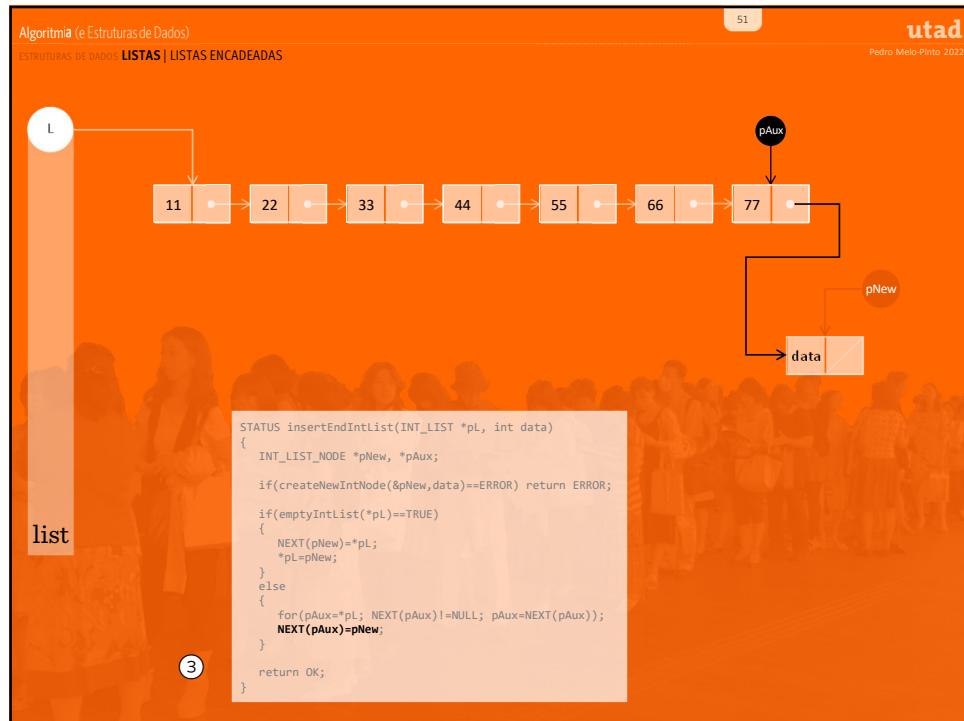












Algoritmia (e Estruturas de Dados)

ALGORITMOS ESTRUTURAS LINEARES DE DADOS | LISTAS

53

utad

Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA

Colecção sequencial

Existem implementações do **TAD Lista** em várias linguagens:

The Java™ Tutorials

[List Implementations](#)

[General-Purpose List Implementations](#)

[There are two general-purpose List implementations — ArrayList and LinkedList. Most of the time, you'll probably use ArrayList, which offers constant-time positional access and is just plain fast. It does not have to allocate a node object for each element in the List, and it can take advantage of System.arraycopy when it has to move multiple elements at the same time.](#)

Home Page > Collections > Implementations

Algoritmia (e Estruturas de Dados)

ALGORITMOS ESTRUTURAS LINEARES DE DADOS | LISTAS

54

utad

Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA

Colecção sequencial

Existem implementações do **TAD Lista** em várias linguagens:

5. Data Structures

5.1. More on Lists

5.1.2. Using Lists as Stacks

5.1.3. List Comprehensions

5.1.4. Nested List Comprehensions

5.2. The del statement

5.3. Tuples and Other Sequences

5.4. Sets

5.5. Dictionaries

5.6. Looping Techniques

5.7. More on Conditions

5.8. Comparing Sequences and Other Types

5.1. More on Lists

The list data type has some more methods. Here are all of the methods of list objects.

```
list.append(x)
Add an item to the end of the list. Equivalent to a[len(a):] = [x].
```

```
list.extend(iterable)
Extend the list by appending all the items from the iterable. Equivalent to a[len(a):] = iterable.
```

```
list.insert(i,x)
Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0, x) inserts at the front of the list, and a.insert(len(a), x) is equivalent to a.append(x).
```

Algoritmia (e Estruturas de Dados)
ALGORITMOS ESTRUTURAS LINEARES DE DADOS | LISTAS

55

utad
Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA

Colecção sequencial

Algumas aplicações:

- verificação de expressões aritméticas
- chamada de funções
- ordenação (p.ex. *radix sort*)
- simulação de eventos
- gestão de processos baseados na premissa *first come first served*
- pesquisa (tabelas de dispersão - *hashing*)

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

56

utad
Pedro Melo-Pinto 2022

LISTA LISTAS ENCADEADAS

Existem variantes das listas encadeadas, que tentam ultrapassar algumas das suas limitações a nível da eficácia das operações de entrada e saída de elementos ou do percurso da lista.

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Lista biencadeada

Cada nó contém :

- A os dados (podem ser de qualquer tipo)
- B uma ligação ao elemento seguinte
- C uma ligação ao elemento anterior



Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Lista biencadeada



Cada nó tem uma ligação para o nó anterior e para o nó seguinte.
A ligação para o nó anterior do primeiro nó aponta para NULL.
A ligação para o nó seguinte do último nó aponta para NULL.

Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Lista biencadeada

```
graph LR; L((L)) --> N1[11]; N1 --> N2[22]; N2 --> N3[33]; N3 --> N4[44]; N4 --> N5[55]; N5 --> NULL((NULL));
```

Cada nó tem uma ligação para o nó anterior e para o nó seguinte.
A ligação para o nó anterior do primeiro nó aponta para NULL.
A ligação para o nó seguinte do último nó aponta para NULL.

Quais as vantagens?

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Lista circular

```
graph LR; L((L)) --> N1[11]; N1 --> N2[22]; N2 --> N3[33]; N3 --> N4[44]; N4 --> N5[55]; N5 --> N6[66]; N6 --> N7[77]; N7 --> N1;
```

O último nó aponta para o primeiro.

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Lista circular

O último nó aponta para o primeiro.

Uma lista circular é normalmente acedida através de uma ligação para o último nó.

QUAL A RAZÃO ?

61 utad Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | LISTAS ENCADEADAS

LISTA LISTAS ENCADEADAS

Lista circular biencadeada

Cada nó tem uma ligação para o nó anterior e para o nó seguinte.
A ligação para o nó anterior do primeiro nó aponta para o último.
A ligação para o nó seguinte do último nó aponta para o primeiro.

62 utad Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADOS | LISTAS | LISTAS ENCADEADAS

63

utad

Pedro Melo-Pinto 2022

LISTA LISTAS ENCADEADAS

Lista circular biencadeada

Cada nó tem uma ligação para o nó anterior e para o nó seguinte.
A ligação para o nó anterior do primeiro nó aponta para o último.
A ligação para o nó seguinte do último nó aponta para o primeiro.

Vantagens?

Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADOS | LISTAS

64

utad

Pedro Melo-Pinto 2022

LISTA COMPLEXIDADE DAS OPERAÇÕES

List Operations: Run time analysis			
Operation	Array-Based	Linked List	Double Linked List
add head	$O(N)$	$O(1)$	$O(1)$
remove head	$O(N)$	$O(1)$	$O(1)$
retrieve head	$O(1)$	$O(1)$	$O(1)$
add tail	$O(1) / O(N)$	$O(1) / O(N)$	$O(1)$
retrieve tail	$O(1)$	$O(1) / O(N)$	$O(1)$
remove tail	$O(1)$	$O(N)$	$O(1)$
access an item by position	$O(1)$	$O(N)$	$O(N)$
access an item by its value	$O(N)$	$O(N)$	$O(N)$
Length	$O(1)$	$O(N)$	$O(N)$
Last	$O(1)$	$O(N)$	$O(N)$
k^{th} -element	$O(1)$	$O(N)$	$O(N)$
First	$O(1)$	$O(1)$	$O(1)$
FindNext	$O(1)$	$O(1)$	$O(1)$
Find	$O(N)$	$O(N)$	$O(N)$
Delete	$O(N)$	$O(N)$	$O(N)$
FindPrev	$O(1)$	$O(N)$	$O(N)$
Insert	$O(N)$	$O(1)$	$O(1)$
isEmpty	$O(1)$	$O(1)$	$O(1)$

depending if the array is full or not; depending on there's a tail reference or not
depending on there's a tail reference or not

Algoritmia (e Estruturas de Dados) 65 utad
Pedro Melo-Pinto 2022

ED01B

LISTAS: STACKS



Algoritmia (e Estruturas de Dados) 66 utad
ESTRUTURAS DE DADO: LISTAS | STACKS
Pedro Melo-Pinto 2022

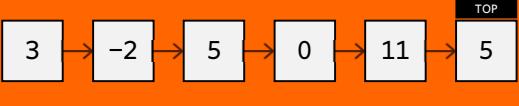
TIPO ABSTRACTO DE DADOS

LISTA STACKS

Subconjunto das listas Restrições na entrada/saída de elementos

Operações de entrada e saída de elementos
são efectuadas por um dos extremos da lista.

normalmente designado por **TOP**



```
graph LR; 3[3] --> -2[-2]; -2 --> 5[5]; 5 --> 0[0]; 0 --> 11[11]; 11 --> 5[5];
```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

67

utad
Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA STACKS

$a_0 \rightarrow a_1 \rightarrow \dots \rightarrow a_i \rightarrow \dots \rightarrow a_{n-2} \rightarrow a_{n-1}$

definições:

listas que apresentam restrições nas operações de entrada|saída de elementos
estas operações são efetuadas somente por uma das extremidades da lista, normalmente designada por **top**
a operação de entrada de dados é designada por **push**
a operação de saída de dados é designada por **pop**

uma *stack* diz-se de tipo LIFO (*last in first out*)
o último elemento a entrar é o primeiro a sair

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

68

utad
Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA STACKS

$a_0 \rightarrow a_1 \rightarrow \dots \rightarrow a_i \rightarrow \dots \rightarrow a_{n-2} \rightarrow a_{n-1}$

propriedades:

pode ter zero ou mais elementos
o elemento posicionado numa das extremidades designa-se por TOP
só o elemento TOP deve ser acedido
um novo elemento é adicionado a seguir ao elemento TOP
em cada instante, só pode ser eliminado o elemento TOP

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

TIPO ABSTRACTO DE DADOS

LISTA STACKS

$a_0 \rightarrow a_1 \rightarrow \dots \rightarrow a_i \rightarrow \dots \rightarrow a_{n-2} \rightarrow a_{n-1}$

operações habituais:

- criação de uma stack (vazia)
- inserção de elemento(s)
- eliminação de elemento(s)
- determinar se a stack está vazia
- determinar o número de elementos da stack

69

utad
Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

TIPO ABSTRACTO DE DADOS

LISTA STACKS METODOLOGIAS DE IMPLEMENTAÇÃO

1 arrays

VANTAGENS	DESVANTAGENS
Entrada/saída de dados rápida. Facilidade de utilização.	<i>Uso ineficaz dos recursos de memória.</i> mesmo com alocação dinâmica convém que o TOP não se aproxime do limite do array

2 listas encadeadas

VANTAGENS	DESVANTAGENS
Uso eficaz dos recursos de memória. Operações de inserção e remoção rápidas. Operações de busca eficientes.	Acesso sequencial aos elementos. Pode haver dificuldade em percorrer a lista.

70

utad
Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

71 utad
Pedro Melo-Pinto 2022

stack S com 9 elementos

ENTRADA DE ELEMENTO NA STACK (PUSH)

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

72 utad
Pedro Melo-Pinto 2022

data newelement

```
STATUS intStackPush(int *S, int size, int *p_TOP, int data)
{
    if (*p_TOP == size) {
        // overflow error
        return ERROR;
    }
    *p_TOP++;
    S[*p_TOP] = data;
    return OK;
}
```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

73 utad Pedro Melo-Pinto 2022

```

    STATUS intStackPush(int *S, int size, int *p_TOP, int data)
    {
        if (*p_TOP == size) {
            // overflow error
            return ERROR;
        }
        *p_TOP++;
        S[*p_TOP] = data;
        return OK;
    }

```

Complexidade temporal: O(1)

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

74 utad Pedro Melo-Pinto 2022

stack S com 9 elementos

SAÍDA DE ELEMENTO DA STACK (POP)

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

75

utad
Pedro Melo-Pinto 2022

27 popped element

```
STATUS intStackPop(int *S, int *p_TOP, int *p_Data)
{
    if (emptyStack(S, *p_TOP)) {
        // underflow error
        return ERROR;
    }
    *p_Data = S[*p_TOP];
    *p_TOP--;
    return OK;
}
```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

76

utad
Pedro Melo-Pinto 2022

27 popped element

```
STATUS intStackPop(int *S, int *p_TOP, int *p_Data)
{
    if (emptyStack(S, *p_TOP)) {
        // underflow error
        return ERROR;
    }
    *p_Data = S[*p_TOP];
    *p_TOP--;
    return OK;
}
```

```
BOOLEAN emptyStack(int *S, int TOP)
{
    if (TOP == -1)
        return TRUE;
    return FALSE;
}
```

Time complexity: O(1)

Complexidade temporal: O(1)

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

77 utad Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

LISTA STACKS METODOLOGIAS DE IMPLEMENTAÇÃO

2 listas encadeadas

VANTAGENS

Entrada/saída de dados rápida.
Uso eficaz dos recursos de memória.

DESVANTAGENS

Menor facilidade de utilização.

LISTA
METODOLOGIAS DE IMPLEMENTAÇÃO

1 arrays

VANTAGENS

Acesso directo a qualquer elemento.
Facilidade em percorrer a lista.

DESVANTAGENS

O operação de inserção (ou remoção) lenta.
Uso inefficaz dos recursos de memória.
necessita muita alocação de memória.

2 listas encadeadas

VANTAGENS

Uso eficaz dos recursos de memória.
Operações de inserção e remoção rápidas.
(dependendo do tipo de lista)

DESVANTAGENS

Acesso sequencial aos elementos.
Pode haver dificuldade em percorrer a lista.

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | STACKS

78 utad Pedro Melo-Pinto 2022

stack S com 7 elementos

ENTRADA DE ELEMENTO NA STACK (PUSH)

79

Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADO: LISTAS | STACKS

utad

Pedro Melo-Pinto 2022

A operação de **push** é a operação de entrada no início de uma lista encadeada
Complexidade temporal: O(1)

80

Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADO: LISTAS | STACKS

utad

Pedro Melo-Pinto 2022

stack S com 7 elementos

SAÍDA DE ELEMENTO DA STACK (POP)

Algoritmia (e Estruturas de Dados)

ESTRUTURAS DE DADOS: LISTAS | STACKS

81

utad

Pedro Melo-Pinto 2022

A operation de **pop** é a operação de saída no início de uma lista encadeada
Complexidade temporal: O(1)

Algoritmia (e Estruturas de Dados)

ED01C

LISTAS: FILAS DE ESPERA

82

utad

Pedro Melo-Pinto 2022

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

83

TIPO ABSTRACTO DE DADOS

LISTA FILAS DE ESPERA

Subconjunto das listas Restrições na entrada/saída de elementos

Operações de entrada e saída de elementos
são efectuadas por extremos opostos da lista.

ENTRADA → 3 → -2 → 5 → 0 → 11 → 5 → SAÍDA

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

84

TIPO ABSTRACTO DE DADOS

LISTA FILAS DE ESPERA

$a_0 \rightarrow a_1 \rightarrow \dots \rightarrow a_i \rightarrow \dots \rightarrow a_{n-2} \rightarrow a_{n-1}$

definições:

listas que apresentam restrições nas operações de entrada|saída de elementos
estas operações são efetuadas por extremidades opostas da lista
a operação de entrada de dados é designada por **enqueue**
a operação de saída de dados é designada por **dequeue**

uma fila de espera diz-se de tipo FIFO (*first in first out*)
o primeiro elemento a entrar é o primeiro a sair

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

85

TIPO ABSTRACTO DE DADOS

LISTA FILAS DE ESPERA

propriedades:

- pode ter zero ou mais elementos
- o elemento de uma das extremidades designa-se por FRONT e o da outra por REAR
- só os elementos FRONT e REAR devem ser acedidos
- um novo elemento é adicionado atrás do elemento REAR
- em cada instante, só pode ser eliminado o elemento FRONT

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

86

TIPO ABSTRACTO DE DADOS

LISTA FILAS DE ESPERA

operações habituais:

- criação de uma fila de espera (vazia)
- inserção de elemento(s)
- eliminação de elemento(s)
- determinar se a fila de espera está vazia
- determinar o número de elementos da fila de espera

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

87

TIPO ABSTRACTO DE DADOS

FILAS DE ESPERA

METODOLOGIAS DE IMPLEMENTAÇÃO

1 arrays

VANTAGENS	DESVANTAGENS
Entrada/saída de dados rápida. Facilidade de utilização.	Uso ineficaz dos recursos de memória. mesmo com alocação dinâmica há que ter cuidado com o "falso" overflow

LISTA
METODOLOGIAS DE IMPLEMENTAÇÃO

1 arrays

VANTAGENS	DESVANTAGENS
Acesso directo a qualquer elemento. Facilidade em percorrer a lista.	O operação de inserção (ou remoção) lenta. Uso ineficaz dos recursos de memória. <small>memória com alocação dinâmica</small>

2 listas encadeadas

VANTAGENS	DESVANTAGENS
Uso eficaz dos recursos de memória. Operações de inserção e remoção rápidas. <small>dependendo do tipo de lista</small>	Acesso sequencial aos elementos. Pode haver dificuldade em percorrer a lista.

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

88

fila de espera Q com 7 elementos

fila de espera Q com 7 elementos

ENTRADA DE ELEMENTO NA FILA DE ESPERA

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

89

utad
Pedro Melo-Pinto 2022

data newelement

```

STATUS intQueueEnqueue(int *Q, int size, int *p_FRONT,
                      int *p_REAR, int data)
{
    if (*p_REAR == size) {
        // overflow error
        return ERROR;
    }
    *p_REAR++;
    if (emptyQueue(*p_FRONT, *p_REAR)) *p_FRONT++;
    S[*p_REAR] = data;
    return OK;
}

BOOLEAN emptyQueue(int FRONT, int REAR)
{
    if (FRONT == -1 && REAR == -1)
        return TRUE;
    return FALSE;
}

```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

90

utad
Pedro Melo-Pinto 2022

data newelement

```

STATUS intQueueEnqueue(int *Q, int size, int *p_FRONT,
                      int *p_REAR, int data)
{
    if (*p_REAR == size) {
        // overflow error
        return ERROR;
    }
    *p_REAR++;
    if (emptyQueue(*p_FRONT, *p_REAR)) *p_FRONT++;
    S[*p_REAR] = data;
    return OK;
}

BOOLEAN emptyQueue(int FRONT, int REAR)
{
    if (FRONT == -1 && REAR == -1)
        return TRUE;
    return FALSE;
}

```

Time complexity: O(1)

Complexidade temporal: O(1)

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

91

utad
Pedro Melo-Pinto 2022

fila de espera Q com 7 elementos

SAÍDA DE ELEMENTO DA FILA DE ESPERA

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

92

utad
Pedro Melo-Pinto 2022

3 output element

```
STATUS intQueueDequeue(int *Q, int *p_FRONT, int REAR, int *p_Data)
{
    if (emptyQueue(*p_FRONT, REAR)) {
        // underflow error
        return ERROR;
    }
    *p_Data = S[*p_FRONT];
    if (*p_FRONT == *p_REAR) {
        *p_FRONT = -1;
        *p_REAR = -1;
    } else
        *p_FRONT++;
    return OK;
}
```

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

93

utad
Pedro Melo-Pinto 2022

3 output element

```
STATUS intQueueDequeue(int *Q, int *p_FRONT, int REAR, int *p_Data)
{
    if (emptyQueue(*p_FRONT,REAR)) {
        // underflow error
        return ERROR;
    }
    *p_Data = S[*p_FRONT];
    if(*p_FRONT == *p_REAR) {
        *p_FRONT = -1;
        *p_REAR = -1;
    } else
        *p_FRONT++;
    return OK;
}
```

emptyQueue() time complexity: O(1)
Complexidade temporal: O(1)

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

94

TIPO ABSTRACTO DE DADOS
FILA DE ESPERA IMPLEMENTADA COM ARRAYS

ENQUEUE 8

DEQUEUE

DEQUEUE

ENQUEUE 4

DEQUEUE

Após um conjunto de entradas e saídas, a Fila de Espera “move-se” para a direita.
Este aspecto pode gerar “falsos” problemas de *overflow*.

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

95 utad Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

FILA DE ESPERA

IMPLEMENTADA COM ARRAYS

	0	1	2	3	4	5	6	7	8	9	10	-
Q →			3	14	17	11	2	11	27	8		
ENQUEUE 8			3	14	17	11	2	11	27	8		
DEQUEUE				14	17	11	2	11	27	8		
DEQUEUE					17	11	2	11	27	8		
ENQUEUE 4						17	11	2	11	27	8	4
DEQUEUE							11	2	11	27	8	4

Pode ser utilizada em situações em que a Fila de Espera fica vazia várias vezes ao longo da sua utilização.

PORQUÉ?

Após um conjunto de entradas e saídas, a Fila de Espera "move-se" para a direita.

Este aspecto pode gerar "falsos" problemas de overflow.

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADO: LISTAS | FILAS DE ESPERA

96 utad Pedro Melo-Pinto 2022

TIPO ABSTRACTO DE DADOS

FILA DE ESPERA CIRCULAR

IMPLEMENTADA COM ARRAYS

	0	1	2	3	4	5	6	7	8	9	10	-
Q →			3	14	17	11	2	11	27			

O último elemento do array fica ligado (de modo conceptual) ao primeiro.

A posição seguinte a $n-1$ (último elemento) é 0.

Uma Fila de Espera vazia é aquela em que **FRONT = REAR = -1**.

Uma Fila de Espera completa acontece quando **FRONT = REAR + 1**.







Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS | LISTAS | FILAS DE ESPERA

103

utad
Pedro Melo-Pinto 2022

queue

Conseguimos melhorar a complexidade temporal?

A operação de `dequeue` pode ser a operação de saída no fim de uma lista encadeada
Complexidade temporal: $O(n)$

Algoritmia (e Estruturas de Dados)
ESTRUTURAS DE DADOS | LISTAS

104

utad
Pedro Melo-Pinto 2022

Leitura Adicional:

- ① Cormen T.H., Leiserson C.E., Rivest R.L. and Stein C., 2009.
Introduction to Algorithms 3rd Edition. **CAPÍTULO 10**
- ② Sedgewick R. and Wayne, K., 2011
Algorithms 4th Edition. **SUBCAPÍTULO 1.3**
- ③ Adrego da Rocha A., 2014
Estruturas de Dados e Algoritmos em C 3^a Edição. **CAPÍTULO 3**
- ④ Melo Pinto P., Couto P., 2021
Estruturas de Dados: Listas. 2 vols.