INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
CAMPUS CEDRO

ALESSON SOUSA VIANA

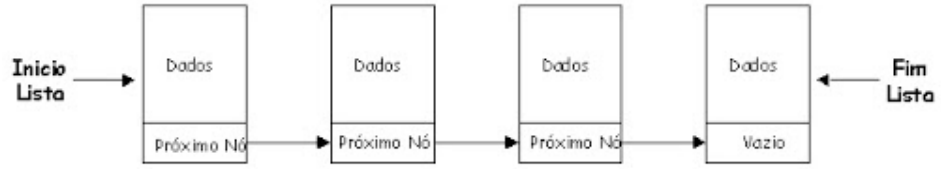
LISTA, FILA, PILHA E ÁRVORE

Resumo

Cedro – Ce

2020.2

Lista



uma **lista** ou **sequência** é uma estrutura de dados abstrata que implementa uma coleção ordenada de valores, onde o mesmo valor pode ocorrer mais de uma vez. Uma instância de uma lista é uma representação computacional do conceito matemático de uma sequência finita, que é, uma tupla. Cada instância de um valor na lista normalmente é chamada de um **item**, **entrada** ou **elemento** da lista. Se o mesmo valor ocorrer várias vezes, cada ocorrência é considerada um item distinto.

Uma estrutura de lista encadeada isoladamente, implementando uma lista com 3 elementos inteiros.

O nome **lista** também é usado para várias estruturas de dados concretas que podem ser usadas para implementar listas abstratas, especialmente listas encadeadas.

As chamadas estruturas de lista *estática'* permitem apenas a verificação e enumeração dos valores. Uma lista **mutável** ou **dinâmica** pode permitir que itens sejam inseridos, substituídos ou excluídos durante a existência da lista.

Muitas linguagens de programação fornecem suporte para **tipos de dados lista** e possuem sintaxe e semântica especial para listas e operações com listas. Uma lista pode frequentemente ser construída escrevendo-se itens em sequência, separados por vírgulas, ponto e vírgulas ou espaços, dentro de um par de delimitadores como parênteses '()', colchetes '[]', chaves '{}' ou chevrons '<>'. Algumas linguagens podem permitir que tipos lista sejam indexados ou cortados como os tipos vetor. Em linguagens de programação orientada a objetos, listas normalmente são fornecidas

como instâncias ou subclasses de uma classe "lista" genérica. Tipos de dado lista são frequentemente implementados usando arrays ou listas encadeadas de algum tipo, mas outras estruturas de dados podem ser mais apropriadas para algumas aplicações. Em alguns contextos, como em programação Lisp, o termo lista pode se referir especificamente à lista encadeada em vez de um array.

É forma de organização através da enumeração de dados para melhor visualização da informação. Em informática, o conceito expande-se para uma estrutura de dados dinâmica, em oposição aos vetores, que são estruturas de dados estáticas. Assim, uma lista terá virtualmente infinitos elementos.

Numa lista encadeada existem dois campos. Um campo reservado para colocar o dado a ser armazenado e outro campo para apontar para o próximo elemento da lista. Normalmente a implementação é feita com ponteiros.

Pilha

São estruturas de dados do tipo LIFO (last-in first-out), onde o último elemento a ser inserido, será o primeiro a ser retirado. Assim, uma pilha permite acesso a apenas um item de dados - o último inserido. Para processar o penúltimo item inserido, deve-se remover o último.

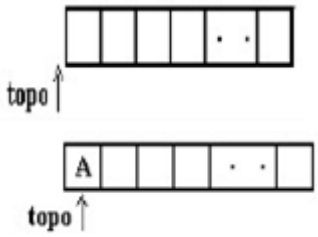
São exemplos de uso de pilha em um sistema:

- Funções recursivas em compiladores;
- Mecanismo de desfazer/refazer dos editores de texto;
- Navegação entre páginas Web;
- etc.

A implementação de pilhas pode ser realizada através de vetor (alocação do espaço de memória para os elementos é contígua) ou através de listas encadeadas. Numa pilha, a manipulação dos elementos é realizada em apenas uma das extremidades, chamada de topo, em oposição a outra extremidade, chamada de base.

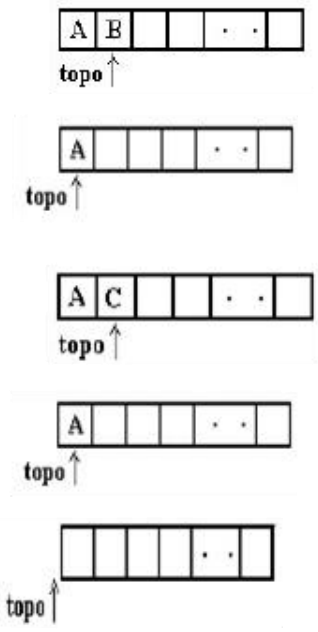
Conceito de Pilhas

Pilhas são listas onde a inserção de um novo item ou a remoção de um item já existente se dá em uma única extremidade, no topo.



Pilha vazia

Insere(A)



Representação da execução de uma pilha com as operações *push* (empilhar) e *pop* (desempliilhar).

Retira(B)

Insere(C)

Retira(C)

Retira(A)

Definição:

Dada uma pilha P=(a(1), a(2), ..., a(n)), dizemos que a(1) é o elemento da base da pilha; a(n) é o elemento topo da pilha; e a(i+1) está acima de a(i).

Pilhas são também conhecidas como listas **LIFO** (last in first out).

Operações Associadas:

1. **criar (P)** - criar uma pilha P vazia
2. **inserir (x, P)** - insere x no topo de P (empilha): push(x,P)
3. **vazia (P)** - testa se P está vazia
4. **topo (P)** - acessa o elemento do topo da pilha (sem eliminar)
5. **elimina (P)** - elimina o elemento do topo de P (desempilha): pop(P)

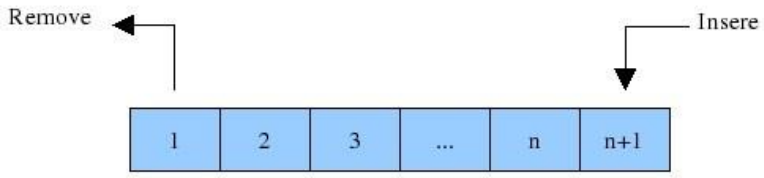
Implementação de Pilhas

Como lista Sequencial ou Encadeada?

No caso geral de listas ordenadas, a maior vantagem da alocação encadeada sobre a sequencial - se a memória não for problema - é a eliminação de deslocamentos na inserção ou eliminação dos elementos. No caso das pilhas, essas operações de deslocamento não ocorrem.

Portanto, podemos dizer que a alocação sequencial é mais vantajosa na maioria das vezes.

Fila



São estruturas de dados do tipo FIFO (first-in first-out), onde o primeiro elemento a ser inserido, será o primeiro a ser retirado, ou seja, adiciona-se itens no fim e remove-se do início.

São exemplos de uso de fila em um sistema:

- Controle de documentos para impressão;
- Troca de mensagens entre computadores numa rede; etc.

A implementação de filas pode ser realizada através de vetor (alocação do espaço de memória para os elementos é contígua) ou através de listas encadeadas

Conceito de Filas

É uma lista linear em que a inserção é feita numa extremidade e a eliminação na outra.
(FIFO: first in, first out).

a_1	a_2	....	...	a_n

Eliminações no início Inserções no final
Exemplo:

Escalonamento de "Jobs": fila de processos aguardando os recursos do sistema operacional.

Definição:

Uma fila é uma estrutura de dados dinâmica que admite remoção de elementos e inserção de novos objetos. Mais especificamente, uma *fila* (= *queue*) é uma estrutura sujeita à seguinte regra de operação: sempre que houver uma remoção, o elemento removido é o que está na estrutura há mais tempo.

Operações associadas:

1. **Criar (F)** - criar uma fila F vazia
2. **Inserir (x, F)** - insere x no fim de F
3. **Vazia (F)** - testa se F está vazia
4. **Primeiro (F)** - acessa o elemento do início da fila
5. **Elimina (F)** - elimina o elemento do início da fila

Implementação de filas como lista Sequencial ou Encadeada? Dinâmica ou Estática?

Só tem sentido falarmos em fila sequencial ou encadeada dinâmica, uma vez que não existe movimentação de elementos. As filas encadeadas são usadas quando não há previsão do tamanho máximo da fila.

Implementação Sequencial de Fila

Definição da Estrutura de Dados


```
Type índice = 0..maxfila;
     fila   = array[1..maxfila] of TipoElem;
Var
  F: fila;
  Começo,        {posição anterior ao primeiro elemento}
Fim: índice;   {posição do último elemento}
```

Operações com Filas:

1. **Criar (F)** - criar uma fila F vazia

```
Procedure CriaFila (Var Começo, Fim: indice);
Begin
  Começo := 0;
  Fim := 0;
End;
```

2. **Inserir (x, F)** - insere x no fim de F

```
Procedure Inserir (x: TipoElem; Var F: fila; Var Fim: indice);
Begin
  If Fim < maxfila
    Then Begin
      Fim := Fim + 1;
      F[Fim] := x;
    End
    Else
     { OVERFLOW }
End;
```

3. **Vazia (F)** - testa se F está vazia

```
Function Vazia (Começo, Fim: indice): Boolean;
Begin
  If Começo = Fim
   Then
    {FILA VAZIA}
End;
```

4. **Primeiro (F)** - acessa o elemento do início da fila

```
Function Primeiro (F: fila; Começo: indice): TipoElem;
Begin
    x := F[Começo + 1];
End;
```

5. **Elimina (F)** - elimina o elemento do início da fila

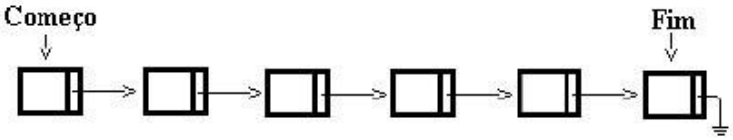
```
Procedure Eliminar (Var Começo: indice, Fim: indice);
Begin
```

```
    If (Começo = Fim)
      Then
        {FILA VAZIA}
      Else
        Começo := Começo + 1;
End;
```

Implementação Encadeada de Fila

Definição da Estrutura de Dados

```
 Type tpont    = ^reg_fila;
fila    = tpont;       reg_fila
= record                  info:
TipoElem;                  lig:
tpont;                End;
 Var Começo, Fim: fila;
```



Operações com Filas:

1. **Criar (F)** - criar uma fila F vazia

```
 Procedure CriaFila (Var Começo, Fim: fila);
   Begin
     Começo := nil;
     Fim := nil;
   End;
```

2. **Inserir (x, F)** - insere x no fim de F, {só se lista for não vazia}

```
 Procedure Inserir (x: TipoElem;  Var Fim:
fila);    Begin      new(Fim^.lig);      Fim :=
Fim^.lig;
```

```
    Fim^.info := x;
    Fim^.lig := nil;
  End;
```

3. Vazia (F) - testa se F está vazia

```
 Function Vazia (Começo, Fim: fila): boolean;
  Begin
    Vazia := ( Começo = nil) and (Fim = nil);
  End;
```

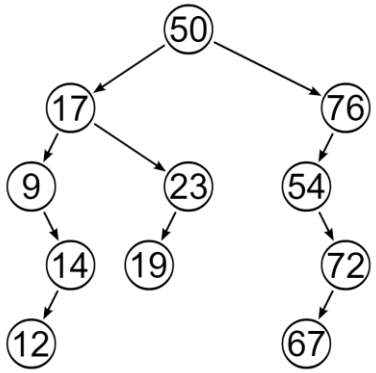
4. Primeiro (F) - acessa o elemento do início da fila

```
 Function Primeiro (Var Começo: fila): TipoElem;
  Begin
    Primeiro := Começo^.info;
  End;
```

5. Elimina (F) - elimina o elemento do início da fila

```
 Procedure Elimina (Var Começo, Fim: fila;);
 Var      p:      fila;
Begin
  if (começo <> nil)
then begin            p
:= Começo;
          Começo := Começo^.lig;      {válido se lista não vazia}
          If ( Começo = Fim)
             Then Begin
                  Começo := nil;
                  Fim      :=   nil;
End;              dispose(p);
       end
 End;
```

Arvore

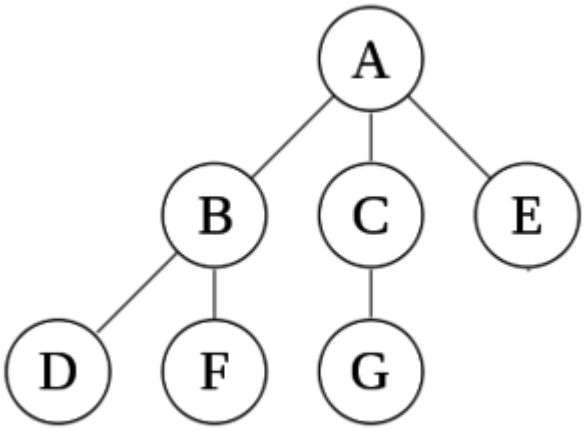


é uma das mais importantes estruturas de dados não lineares. Herda as características das topologia em árvore. Conceitualmente diferente das listas, em que os dados se encontram numa sequência, nas árvores os dados estão dispostos de forma hierárquica, seus elementos se encontram "acima" ou "abaixo" de outros elementos da árvore.

São estruturas eficientes e simples em relação ao tratamento computacional, diferentemente dos grafos]. Há inúmeros problemas no mundo real que podem ser modelados e resolvidos através das árvores. Estruturas de pastas de um sistema operacional, interfaces gráficas, bancos de dados e sites da internet são exemplos de aplicações de árvores.

Uma árvore é formada por um conjunto de elementos que armazenam informações chamados **nodos** (ou nós). Toda a árvore possui o elemento chamado **raiz**, que possui ligações para outros elementos denominados ramos ou filhos. Estes ramos podem estar ligados a outros elementos que também podem possuir outros ramos. O elemento que não possui ramos é conhecido como **nó folha**, nó terminal ou nó externo.

Uma terminologia muito utilizada nas estruturas de árvores tem origem das árvores genealógicas. O relacionamento entre nodos é descrito com os termos "pai" (ou "mãe") para os antecessores diretos de um nodo, "filhos" (ou "filhas") para os descendentes diretos e "irmãos" (ou "irmãs") para todos os nodos com mesmo pai



Representação simples de uma árvore. O nodo A, a raiz, tem como filhos diretos: B, C, E e filhos indiretos: D, F e G. B, C e E são irmãos, assim como D e F.

Como visto, listas podem ser convenientemente definidas da seguinte forma: Uma lista do tipo T é

Uma lista (estrutura) vazia ou

Uma concatenação (cadeia) de um elemento do tipo T com uma lista cujo tipo básico também seja T

Nota-se que a recursão é utilizada como ferramenta de definição

Uma árvore é uma estrutura sofisticada cuja definição por meio de recursão é elegante e eficaz

Uma árvore, com tipo T, pode ser definida recursivamente da seguinte forma:

Uma árvore (estrutura) vazia ou

Um nó do tipo T associado a um número finito de estruturas disjuntas de árvore do mesmo tipo T, denominadas subárvores

Observando a similaridade das definições é evidente que uma lista possa ser considerada como uma árvore na qual cada nó tem, no máximo, uma única subárvore

Por este motivo, uma lista é também denominada árvore degenerada

Uma árvore é um conjunto finito de um ou mais nós (ou vértices) tais que

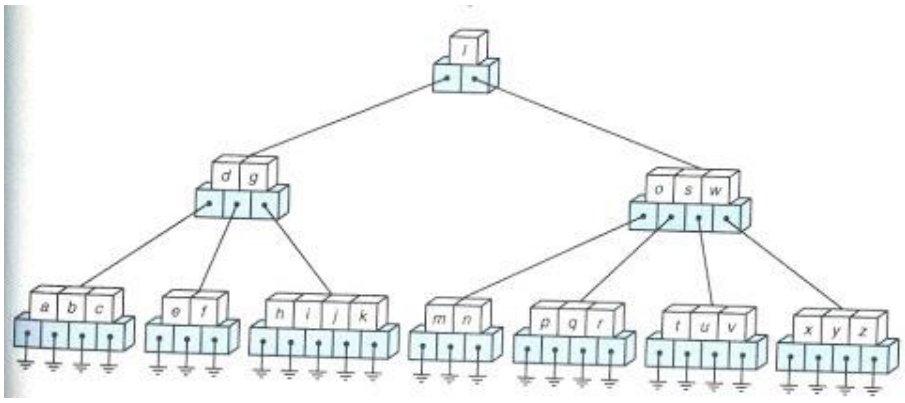
Existe um nó especial, denominado raiz

Os demais nós encontram-se desdobrados em n ≥ 0 conjuntos disjuntos T1, ..., Tn sendo que cada conjunto se constitui numa árvore

T1, ..., Tn são denominadas subárvores da raiz

Utilizaremos grafos para representar árvores

Todavia, existem outras representações equivalentes para árvores: conjuntos aninhados (diagrama de inclusão), parênteses aninhados, paragrafação (indentation)



REFEREÊNCIAS

Árvores, árvores binárias e percursos. uff. Disponível em: < http://www.ic.uff.br/~boeres/slides_ed/ed_ArvoresPercursos.pdf >. Acesso em: 15/09/2020.

Algoritmos e Estrutura de Dados. Iftm. Disponível em: < http://www.waltenomartins.com.br/algdados_apostila_pilhafila.pdf >. Acesso em: 15/09/2020.

Pilhas - Departamento de Computação e Matemática - USP. Usp. Disponível em: < http://dcm.ffclrp.usp.br/~augusto/teaching/aedi/AED-I-Pilhas.pdf >. Acesso em: 15/09/2020.