Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA

# Mestrado Decisão Económica e Empresarial
# COMPUTAÇÃO

**Sumário:**

Tipos de dados abstractos.
Representação de Grafos. Determinação de componentes conexas.

## Tipo de dados abstractos ADT

**ADT abstract data type**

Pode ser pensado como um **modelo matemático com uma colecção de operações definidas para o modelo.**

Exemplo: conjunto de inteiros conjuntamente com as operações União, Intersecção e Diferença de conjuntos.

Podem estar envolvidas mais do que um tipo de ADT numa operação, inclusivé o resultado pode ser uma instância de outra ADT mas pelo menos um dos operandos é do tipo da ADT em causa.



## Generalização e Encapsulamento

São **generalizações** de tipos de dados primitivos (por exemplo inteiro); os procedimentos são igualmente generalizações de operações primitivas (por exemplo +,-,...)

**Encapsulamento** no sentido em que a definição do tipo bem como as operações envolvidas estão localizadas num secção do programa.

- Vantagens
Fora da sua definição, a ADT pode ser tratada como um tipo de dado primitivo, legibilidade e facilidade de manutenção.
- Desvantagens
Quando as operações envolvem mais do que uma ADT essas operações aparecem em ambas as ADTs.



## Tipo, estrutura de dados e ADT

Tipo (ou tipo de dado) é o conjunto de valores que uma variável pode assimir. Exemplos Bool, inteiro...

ADT é um modelo matemático, juntamente com a várias operações definidas no modelo

Estruturas de dados são colecções de variáveis, possivelmente incluindo dados de diferentes tipos, relacionados de várias formas. usadas para representar os modelos subjacentes a uma ADT

## Agregação de Células

Célula (cell) é a peça básica da estruturas de dados

representa-se 

As Estruturas de dados são criadas dando designações a agregados de células

exemplos

mais simples array
[(VBA) Dim nome(n) As tipo de dado ]

representa-se 

registo (record ou struct ou ...)

Ficheiro é outro modo de agrupar células

## Acesso

Arrays e records são de Acesso Aleatório

fácil acesso a uma célula
exigem especificação da memória necessária
cada célula tem tipo de dados especificado

Ficheiros são de Acesso Sequêncial
difícil acesso a uma célula
não exigem especificação da memória necessária
tipo de dados especificado

## Ponteiros e cursores

Outra forma de agrupar células é especificar o seu relacionamento através de ponteiros (ou apontadores)

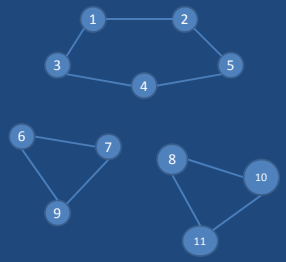
Ponteiros é uma célula cujo conteúdo indica outra célula,
é uma célula cujo conteúdo é o endereço de outra célula

Linguagens como o Pascal e o C suportam ponteiros, quando assim não é usam-se cursores (uma célula com um inteiro que representa o índice dum vector)

## Grafos

Grafo G=(N,E)

N conjunto de vértices
E conjunto de arestas

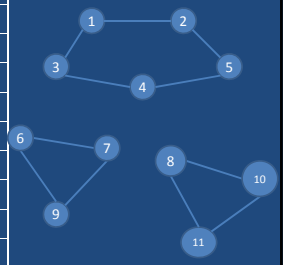



## Matriz de adjacência

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 1 | 1 | | | | | | | | |
| 2 | 1 | | | | 1 | | | | | | |
| 3 | 1 | | | 1 | | | | | | | |
| 4 | | | 1 | | 1 | | | | | | |
| 5 | | 1 | | 1 | | | | | | | |
| 6 | | | | | | | 1 | | 1 | | |
| 7 | | | | | | 1 | | | 1 | | |
| 8 | | | | | | | | | | 1 | 1 |
| 9 | | | | | | 1 | 1 | | | | |
| 10 | | | | | | | | 1 | | | 1 |
| 11 | | | | | | | | 1 | | 1 | |

Grafo G=(N,E)

N conjunto de vértices
E conjunto de arestas

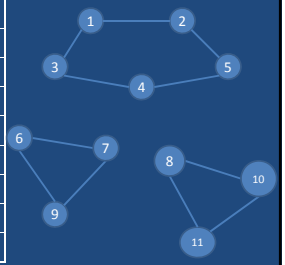



## Matriz de incidência

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | | | | | | | | | |
| 2 | 1 | | 1 | | | | | | | | |
| 3 | | 1 | | 1 | | | | | | | |
| 4 | | | | 1 | 1 | | | | | | |
| 5 | | | 1 | | 1 | | | | | | |
| 6 | | | | | | 1 | | 1 | | | |
| 7 | | | | | | 1 | 1 | | | | |
| 8 | | | | | | | | | 1 | 1 | |
| 9 | | | | | | | 1 | 1 | | | |
| 10 | | | | | | | | | 1 | | 1 |
| 11 | | | | | | | | | | 1 | 1 |

Grafo G=(N,E)

N conjunto de vértices
E conjunto de arestas




## Lista de adjacência

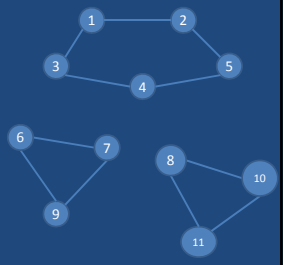
AdjList[1] → 2 → 3

AdjList[2] → 1 → 5

AdjList[11] → 8 → 10

Grafo G=(N,E)

N conjunto de vértices
E conjunto de arestas




## Pilhas e Filas

Pilha - implementa a política LIFO (last-in, first-out)
inserção : início da lista
remoção: início da lista

Fila - implementa a política FIFO (first-in,first-out)
inserção : final da lista
remoção : início da lista

## Opções de implementação

**Arrays** – aqui a afectação sequêncial é favorável porque inserções e remoções não implicam movimentação.

elementos a serem inseridos ou removidos estão em posições especiais.

**Pilha -** uma variável que mantenha o topo da lista (top).

**Fila -** duas variáveis que mantenham o início (head) e o final da lista (tail).



## Identificação das componentes conexas

- Pilha LIFO

Input

- n número de vértices;
- m número de arestas
- Ar1(m),Ar2(m) arestas

Variáveis auxiliares
•Grau(n)
•Adj(n)(n)
•Pilha(2*m)

Output

- nComp número de componentes conexas;
- Componente(n) componentes;



## Identificação das componentes conexas

```
Construir o vector de graus dos vértices e a matriz de adjacência, inicializar a zero  o vector comp;
ncomp=0
    For i=1 to n
        if comp(i)=0
            ncomp=ncomp+1
              DimPilha=1
              Pilha(DimPilha)=i
              Do while (dimPilha>0)
                    v=Pilha(DimPilha)
                    DimPilha=DimPilha-1
                    Comp(v)=ncomp
                    for j=1 to grau(v)
                          if Comp(adj(v,j))=0 then
                                DimPilha=DimPilha+1
                                Pilha(DimPilha)=adj(v,j)
                          endif
                    next j
            Loop
        endif
    Next i
```



## Exercícios

Implementar um algoritmo de determinação das componentes conexas de um grafo com utilização de uma pilha