Instituto Superior de Economia e Gestão
UNIVERSIDADE TÉCNICA DE LISBOA

Mestrado Decisão Económica e Empresarial
COMPUTAÇÃO

Sumário:

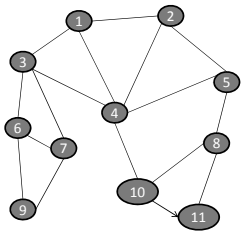
Árvores

Vectorizar uma matriz

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

Grafo G=(N,E)

N conjunto de vértices
E conjunto de arestas

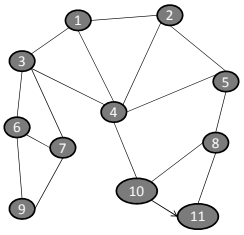


Vectorizar uma matriz

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	1	1							
2	1	0		1	1						
3	1		0	1		1	1				
4	1	1	1	0	1					1	
5		1		1	0			1			
6			1			0	1		1		
7			1			1	0		1		
8					1			0		1	1
9						1	1		0		
10				1				1		0	1
11								1		1	

Grafo G=(N,E)

N conjunto de vértices
E conjunto de arestas



Na matriz A (triângular superior i<j) quadrada de ordem n

a_{ij} linha i coluna j matriz

No vector índice k

Se i>j troca i com j

$$k = \sum_{s=1}^{i-1} (n-s) + j - i$$

	1	2	3	4	5	6
1		1	2	3	4	5
2			6	7	8	9
3				10	11	12
4					13	14
5						15
6						

Árvores: Motivação

Diversas aplicações necessitam de estruturas mais complexas do que as listas, pilhas e filas?

Exitem algoritmos eficientes para tratamento de árvores.

Árvore

Estabelece um estrutura hierárquica numa colecção de objectos. Exemplos:

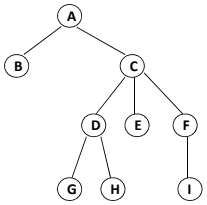
- Organização dos directorios num computador;
- Índice de um livro;
- Árvore geneológica;
- Estrutura hierárquica de uma organização;
- ...

Definição

Uma árvore T é um conjunto finito de elementos denominados nós ou vértices tais que:

- T = 0 é a árvore vazia ou
- Existe um nó r, chamado raiz de T; os nós restantes constituem um único conjunto vazio ou são divididos em m≥1 conjuntos distintos não vazios que são as sub-árvores de r, cada sub-árvore é também uma árvore. (recursivo)

Hierárquica



Alinhamento dos nós

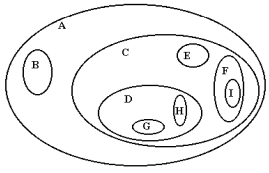
A
B
C
D
G
H
E
F
I

Representação

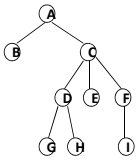
Parênteses aninhados

(A (B) (C (D (G) (H)) (E) (F (I))))

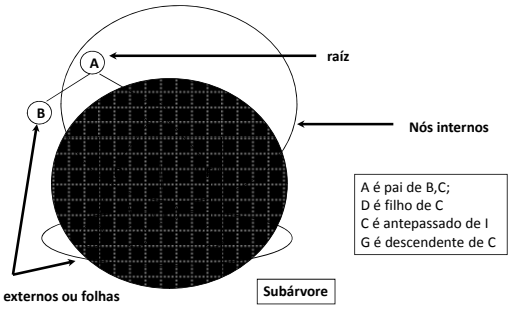
Diagramas de inclusão



Hierárquica



Terminologia



Terminologia

Qualquer nó da árvore (raíz incluída)

- ou é interno – nó que tem um ou mais filhos
- ou é externo (ou folha) – nó sem filhos.

Diz-se que o nó u é antepassado de v sse u=v ou u antepassado do pai de v;
Diz-se que nó o v é descendente de u sse v=u ou v descendente do filho de u;
Subárvore de T com raíz u – nó u e todos os seus descendentes mantendo as relações entre si;

Aresta da árvore – (u,v) em que u é pai de v;
Caminho na árvore – sequência de nós em que 2 nós consecutivos são uma aresta da árvore
(na terminologia de computação)

Altura e profundidade

Altura

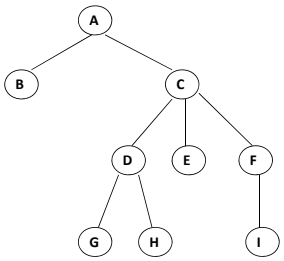
- Do nó – número de nós do caminho mais longo até uma folha;
- Da árvore – altura da raíz

convenções: altura da folha 1, altura da árvore vazia 0

Profundidade

- Do nó – número de nós do único caminho da raíz ao nó;
- Da árvore – profundidade máxima (das suas folhas);

convenções: profundidade da raíz 1, **profundidade** da árvore vazia 0



Alturada árvore 4
Profundidade da árvore 4

Nó	altura	prof
A	4	1
B	1	2
C	3	2
D	2	3
E	1	3
F	2	3
G	1	4
H	1	4
I	1	4

Operações na ADT árvore

- Determinar o número de nós da árvore
- Responder se a árvore está ou não vazia
- Verificar se um nó
  - É interno
  - É folha
  - É a raíz
- Devolver
  - O pai de um nó
  - Os filhos de um nó
  - A informação contida num nó
- Operações para
  - Criar uma árvore
  - Colocar informação num nó
  - Adicionar uma subárvore a uma árvore

Percursos

Ordem pela qual são visitados os nós de uma árvore

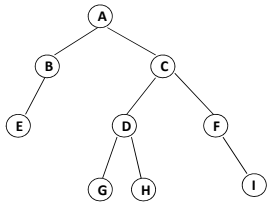
- **Prefixo** um nó é visitado antes dos seus descendentes (pre-order)
- **Sufixo** um nó é vistado depois dos seus descendentes (post-order)
- **Por nível** os nós de um nível são visitados antes dos nós do nível seguinte

Árvores binárias

Uma árvore binária é uma árvore em que cada nó tem no máximo 2 filhos.

Os filhos são habitualmente designados por
filho esquerdo e
filho direito.

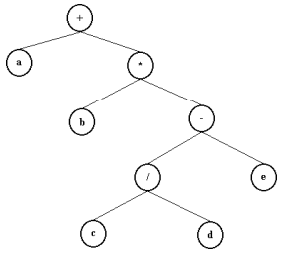
Exemplo de árvore binária



Exemplos de Aplicações

- Árvores de decisão
  - Questões SIM / NÃO
- Pesquisa de elementos
- Árvores com expressões aritméticas
  - Operandos nos nós internos
  - Operadores nas folhas

(a + (b *((c / d) - e)))



a+(b*(c/d-e))

b*(c/d-e)

c/d-e

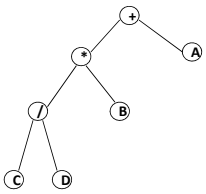
c/d

Percursos em árvores binárias

E – movimentar para a esquerda
D – movimentar para a direita
V – visitar o nodo

Como E precede sempre D temos:
- VED prefixo (pre-order)
- EVD infixo (in-order)
- EDV sufixo (post-order)

- Por nível



Percurso	expressão
Prefixo	+*/CDBA
Infixo	C/D*B+A
Sufixo	CD/B*A+
Por nível	+*A/BCD

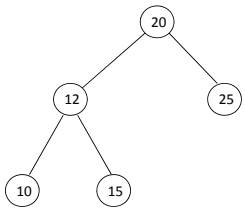
Árvore binária de pesquisa

Numa árvore binária ordenada, para o todo o nó *u* verifica-se:

- Qualquer nó pertencente à subárvore esquerda com raíz em *u* tem valor menor do que o valor de *u*
- Qualquer nó pertencente à subárvore direita com raíz em *u* tem valor maior do que o valor de *u*

Uma árvore binária de pesquisa está ordenada:
- Se não há repetições de elementos
- Se as subárvores esquerda e direita também são árvores binárias de pesquisa

Árvore binária de pesquisa exemplo



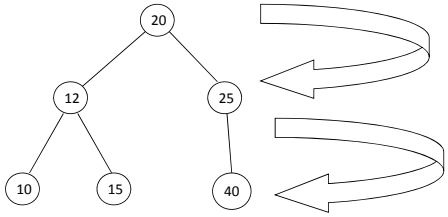
Pesquisa - Complexidade

A pesquisa de um elemento na árvore binária de pesquisa devolve ou o nó em que se encontra o elemento ou a informação de que tal elemento não se encontra na árvore.

No pior caso o número de comparações é da ordem da altura da árvore.

INSERÇÃO Árvore binária de pesquisa

Inserção de 40

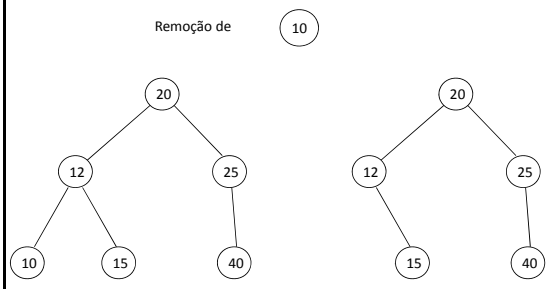


REMOÇÃO Árvore binária de pesquisa

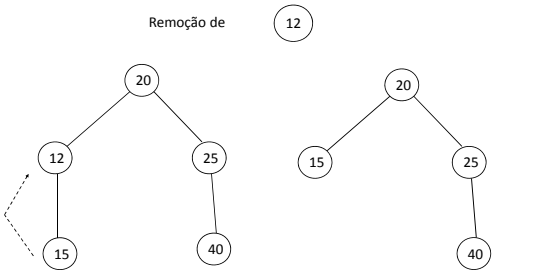
Remoção de um elemento que

- é uma folha – retira-se simplesmente esse nó da árvore;
- é nó com apenas um filho – filho colocado no lugar do nó a retitrar;
- É nó com 2 filhos – coloca-se no lugar do nó a remover o menor elemento da subárvore direita.

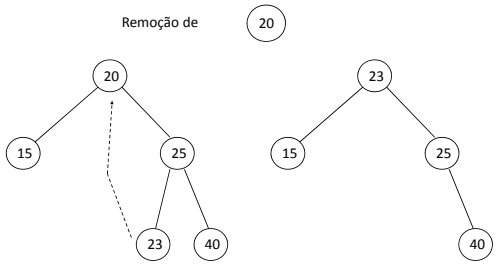
REMOÇÃO de uma folha



REMOÇÃO elemento com um filho



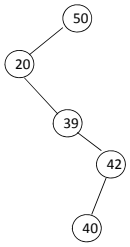
REMOÇÃO elemento com 2 filhos



Árvore – lista

Árvore binária de pesquisa pode degenerar numa lista e assim o tempo de pesquisa passa a ser como numa lista

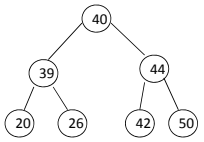
Sequência das inserções na árvore: 50,20,39,42,40

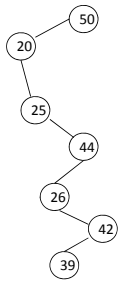


Neste caso, não existe vantagem em ter uma árvore

Árvore Completas

Árvores completas minimizam o número de comparações efectuadas no pior caso para pesquisar um elemento.





Uma árvore diz-se completa se o seu número de nós for igual a
Sendo h altura da árvore $2^h - 1$

Permite a representação
V[0] raíz; filhos de v[i] são v[2*i+1] e v[2*i+2]

Árvores Balanceadas AVL

Uma árvore binária diz-se balanceada AVL sse

para todo o nó da árvore a diferença entre as alturas das suas subárvores não excede 1.

Uma árvore que contenha um nó que não satisfaça esta condição é uma árvore desregulada.

A diferença entre as alturas das subárvores de um nó é designada por factor de balanceamento do nó.

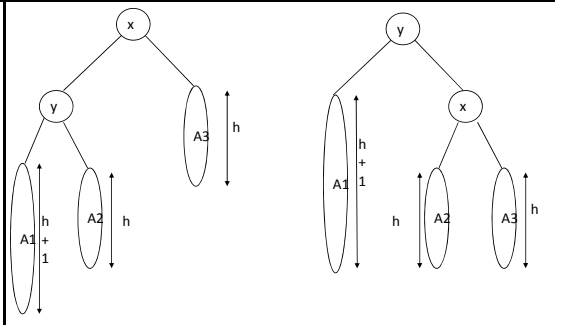
Adel'son-Vel'skii e Landis (1962)

AVL

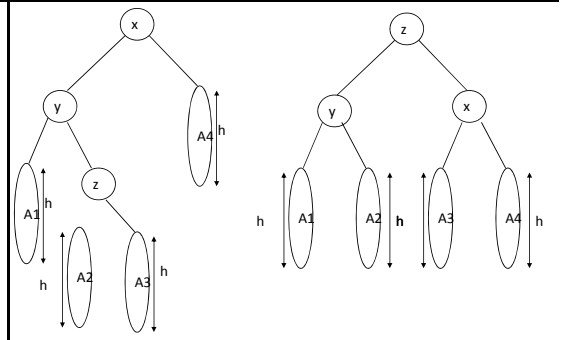
A manutenção do balanceamento de uma árvore é feita por diversas operações de rotação
- Simples ou dupla
- à direita e à esquerda

Implicam consumo de tempo, por isso há que ponderar o seu uso.
Uma possibilidade para conseguir menos consumo de tempo é manter um registo em cada nó com seu factor de balanceamento, mas consome mais memória e é mais complexo.

Rotação simples à direita



Rotação dupla à direita



Sites interessantes com animação sobre árvores

http://www.csi.uottawa.ca/~stan/csi2514/applets/avl/BT.html

Algoritmo de kruskal

```
' Inicialização:
      AGM = { }

' Iteração
      Enquanto (|AGM| < n -1) AND (E <> { })
            e=menor aresta de E
            E=E – {e}
            Se (E U {e} não contém ciclo)
                  AGM=AGM U {e}
      Se (|AGM|< n -1) grafo não conexo
```

G=(V,E) grafo
n número de vértices
m número de arestas

Pontos cruciais da implementação

1. Determinação da menor aresta de E ainda não examinada
2. Registo das arestas da árvore de modo a testar eficientemente se a união de uma nova aresta forma ciclo

Estruturas de dados

Entrada de dados:
3 vectores

	1	2	3	4	5	...		m
ArestaV1								
ArestaV2								
PesoAresta								

Saída de Resultados:
2 vectores e um escalar

	1	2	...		n-1
AGMArestaV1					
AGMArestaV2					

PesoAGM

Determinação da menor aresta de E

Utilizar um ***heap***

=> **basta em cada iteração identificar a menor aresta ainda não examinada**. Habitualmente examina-se apenas uma parte das arestas de E e portanto ordenar completamente o conjunto das arestas resulta em trabalho desnecessário

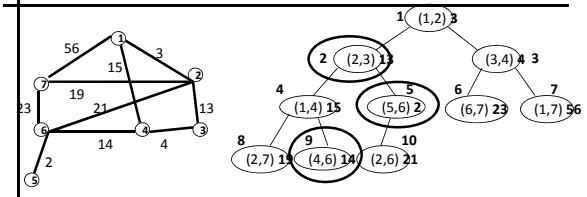
Construção do heap inicial O(m)
Obtenção da aresta seguinte O(log m)

se o número de arestas exeminadas for p a complexidade é O(m+p*log m)

Manipular o *heap*
Construir *heap*

Remover aresta do *heap*
Restaurar *heap*

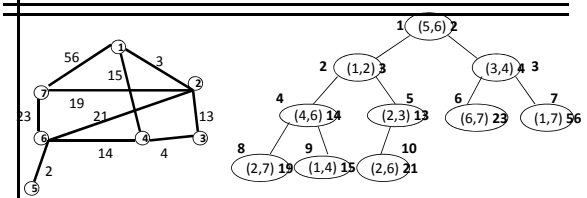
EXEMPLO - dados



	1	2	3	4	5	6	7	8	9	10
	1	2	3	1	5	6	1	2	4	2
	2	3	4	4	6	7	7	7	6	6
	3	13	4	15	2	23	56	19	14	21


```
Sub Heap(Primeiro, DimHeap)
Dim menor As Integer  Dim esq As Integer  Dim dir As Integer
Dim htemp1 As Integer  Dim htemp2 As Integer  Dim htemp3 As Double
esq = 2 * Primeiro
dir = 2 * Primeiro + 1
If ((esq <= DimHeap)
     If (ArestaPeso(esq) < ArestaPeso(Primeiro))) Then
      menor = esq
Else
     menor = Primeiro
End If
If ((dir <= DimHeap)
          if (ArestaPeso(dir) < ArestaPeso(menor))) Then
     menor = dir
End If
If menor <> Primeiro Then ' trocar
   htemp1 = ArestaV1(Primeiro)   htemp2 = ArestaV2(Primeiro) htemp3 = ArestaPeso(Primeiro)
   ArestaV1(Primeiro) = ArestaV1(menor)
   ArestaV2(Primeiro) = ArestaV2(menor)
   ArestaPeso(Primeiro) = ArestaPeso(menor)
   ArestaV1(menor) = htemp1 ArestaV2(menor) = htemp2  ArestaPeso(menor) = htemp3
   If menor <= DimHeap / 2 Then Call Heap(menor, DimHeap)
End If
```


EXEMPLO



	1	2	3	4	5	6	7	8	9	10
	5	1	3	1	2	6	1	2	1	2
	6	2	4	4	3	7	7	7	4	6
	2	3	4	15	13	23	56	19	15	21

Ordenadas no heap

Depois de
For i = M / 2 To 1 Step -1
** Call Heap(i, M)**
Next i

Manipular o heap

Construir *heap*	For i = M / 2 To 1 Step -1 Call Heap(i, M) Next i
Remover aresta do *heap*	ArestaV1(1) = ArestaV1(Ultimo) ArestaV2(1) = ArestaV2(Ultimo) ArestaPeso(1) = ArestaPeso(Ultimo) Ultimo = Ultimo – 1
Restaurar *heap*	Call Heap(1, Ultimo)

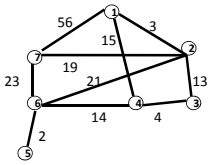
Verificar se (u,v) forma ciclo

Armazenar as arestas já incluídas na AGM em **árvores com raíz**

=> a raíz é um elemento do subconjunto que serve para o identificar

Funções

Uniao
Pesquisa



Pai

-2	1	-2	3	-2	5	-1

Pesquisa(2)=1
Pesquisa(3)=3

Juntar a aresta (2,3)

-4	1	1	3	-2	5	-1


```
Function Pesquisa(i)
Dim Ptr As Integer
Ptr = i
While Pai(Ptr) > 0
   Ptr = Pai(Ptr)
Wend
Pesquisa = Ptr
End Function
```



```
Sub Uniao(i, j)
Dim x As Integer
x = Pai(i) + Pai(j)
If Pai(i) > Pai(j) Then
   Pai(i) = j
   Pai(j) = x
Else
   Pai(i) = x
   Pai(j) = i
End If
End Sub
```



```
Inicialização:
    Para todo o v    V Pai(v) = -1
    Construir o heap inicial com m arestas O(m)
    AEconta = 0 ' numero de arestas já examinadas
    AGMconta =0 ' numero de arestas na AGM
    AGM = { }

Iteração
    Enquanto (AGMconta < n -1){ max m vezes
        e=(u,v) do topo do heap
        AEconta = AEconta +1
        remover e do topo do heap
        restaurar o heap O(log m)
        r1=Pesquisa(u)
        r2=Pesquisa(v)
        Se (r1<>r2){
            AGM = AGM U {e}
            AGMconta =AGMconta+1
            Uniao(r1,r2)
        }
    }
    Se (AGMconta < n -1) escrever grafo não conexo
```

G=(V,E) grafo
n número de vértices
m número de arestas

Complexidade O(m log n)

$m < n^2 \Rightarrow \log(m) = O(\log n)$