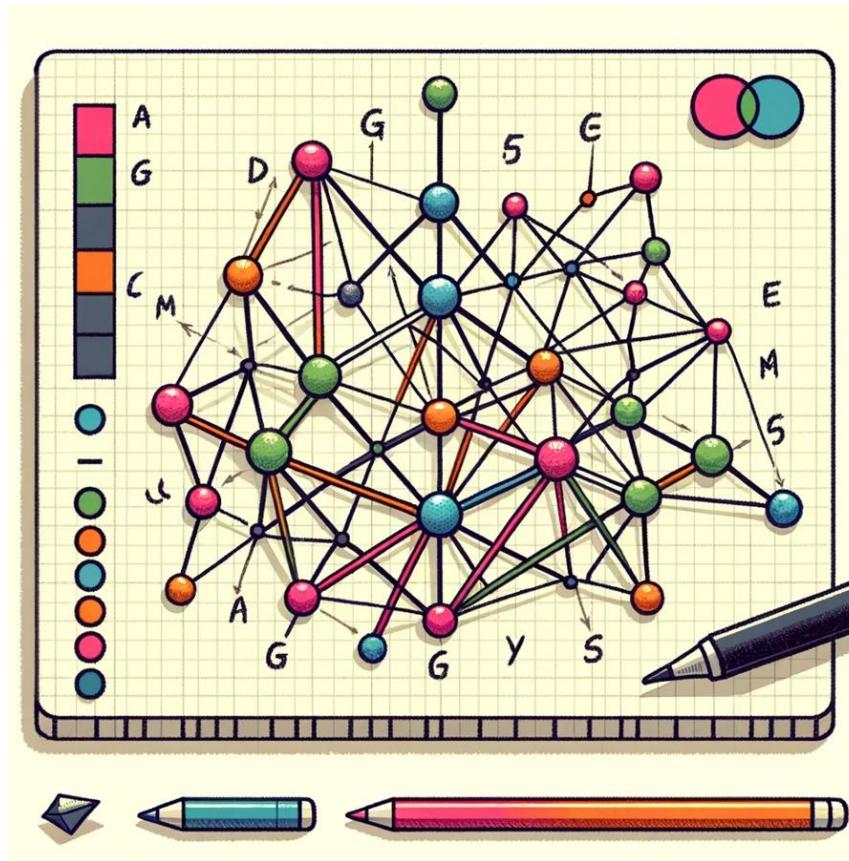


# Teoria dos Grafos

# Jairo Lucas

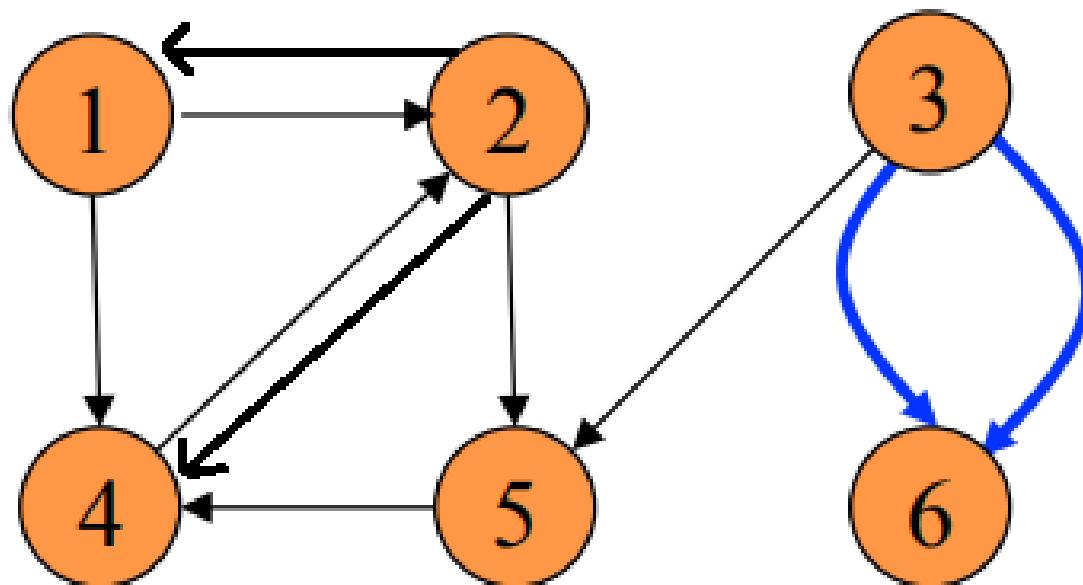


# Teoria dos Grafos – **Grafos Direcionados (Digrafos)**

## **Grafos Direcionados (Digrafos)**

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Grafos Direcionados (Digrafos)** : Grafos direcionados, também conhecidos como digrafos, são tipos de grafos onde as **arestas têm uma direção associada**, indicando uma relação **unidirecional** entre dois vértices.



# Teoria dos Grafos – Grafos Direcionados (Digrafos)

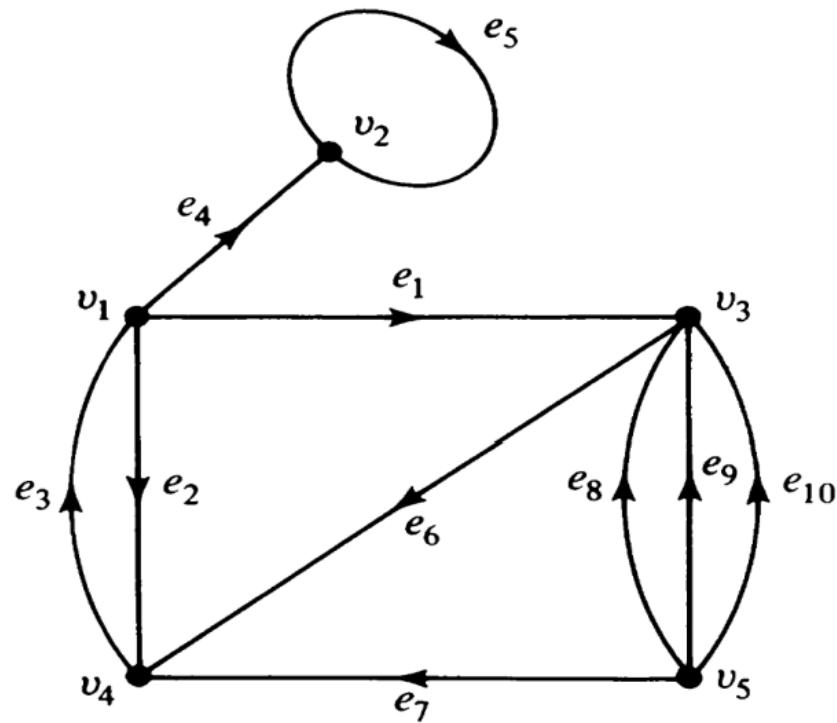
- **Grafos Direcionados (Digrafos)** : Grafos direcionados, também conhecidos como digrafos, são tipos de grafos onde as **arestas têm uma direção associada**, indicando uma relação unidirecional entre dois vértices.
- **Grau de Entrada e Saída**: Cada vértice em um grafo direcionado tem dois tipos de graus –
  - o grau de **entrada** (o número de arestas que entram no vértice)
  - e o grau de **saída** (o número de arestas que saem do vértice).
  - **Uma fonte**: quando o grau de entrada é nulo
  - **Um sumidouro**: Quando o grau de saída é nulo.
- Duas arestas são **paralelas** se elas incidem nos mesmos vértices e possuem a **mesma orientação**.
- **Vertices adjacentes**: Se existe uma aresta direcionada de **A para B** ( $A \rightarrow B$ ), então dizemos que **B** é adjacente a A, porém, B não é adjacente a A.

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Grafos Direcionados (Digrafos)** : Grafos direcionados, também conhecidos como digrafos, são tipos de grafos onde as **arestas têm uma direção associada**, indicando uma relação unidirecional entre dois vértices.
- Aplicações:
  - **Sistemas de Navegação e Mapas**: Para representar ruas de mão única.
  - **Redes Sociais**: Para indicar relações assimétricas, como seguidores e seguidos.
  - **Análise de Redes**: Em telecomunicações, redes de computadores e outros sistemas de fluxo de informação.

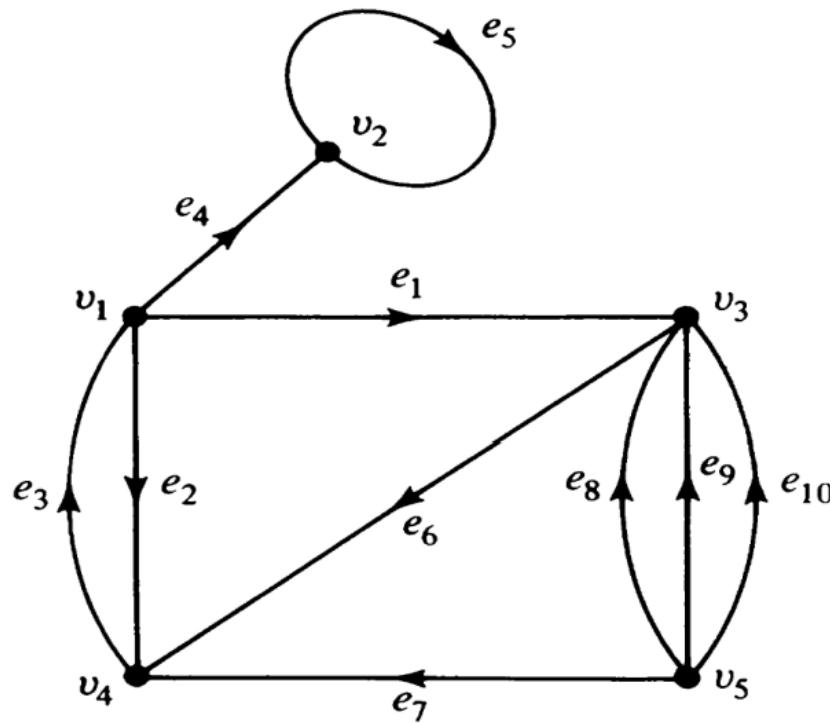
# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Grafos Direcionados (Digrafos)** : Grafos direcionados, também conhecidos como digrafos, são tipos de grafos onde as **arestas têm uma direção associada**, indicando uma relação unidirecional entre dois vértices.



# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Grafos Direcionados (Digrafos)** : Grafos direcionados, também conhecidos como digrafos, são tipos de grafos onde as **arestas têm uma direção associada**, indicando uma relação unidirecional entre dois vértices.



$$\begin{aligned}d_s(v_1) &= 3, & d_e(v_1) &= 1; \\d_s(v_2) &= 1, & d_e(v_2) &= 2; \\d_s(v_5) &= 4, & d_e(v_5) &= 0.\end{aligned}$$

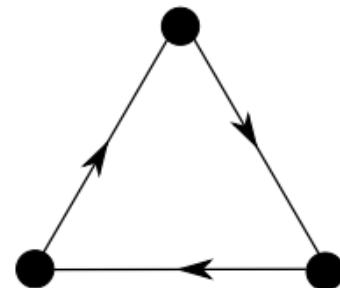
# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Tipos de Digrafos:**

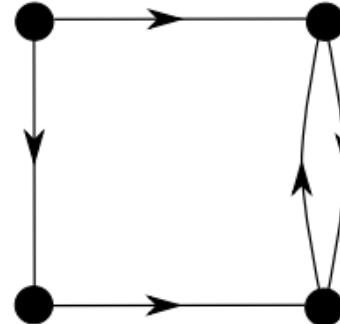
- **Simples**: se não possui loops ou **arestas paralelas**;
- **Assimétrico** se possui no máximo uma aresta orientada entre cada par de vértices;
- **Simétrico** se para cada aresta  $(a, b)$  existe também uma aresta  $(b, a)$ ;
- **Completo Simétrico**; se  $G$  é simples e existe exatamente uma aresta direcionada de todo vértice para todos os outros vértices;
- **Completo Assimétrico** se  $G$  é assimétrico existe exatamente uma aresta entre cada par de vértices;
- **Balanceado** se os graus de entrada e saídas são iguais para todos os vértices

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

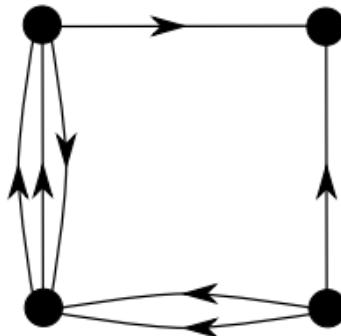
- **Tipos de Digrafos:**



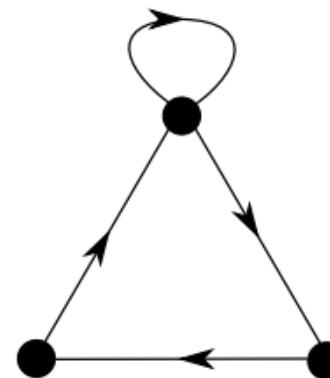
a) Digrafo Simples



b) Digrafo Simples



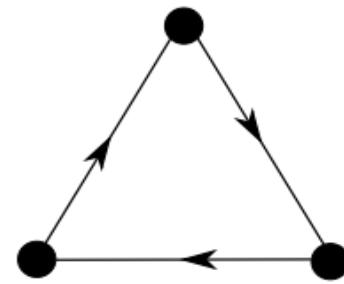
c) Digrafo NÃO simples



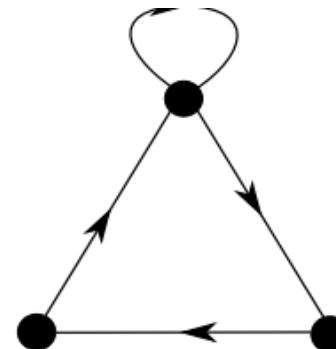
d) Digrafo NÃO Simples

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

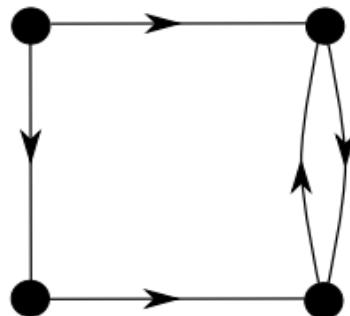
- **Tipos de Digrafos:**



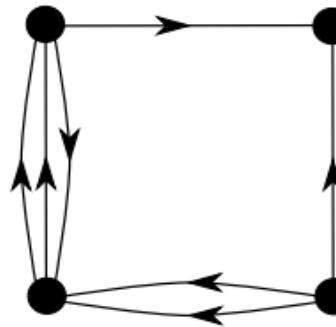
e) Digrafo Assimétrico



f) Digrafo Assimétrico



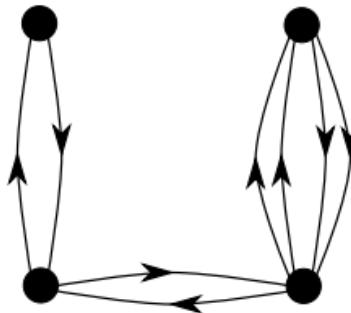
g) Digrafo NÃO assimétrico



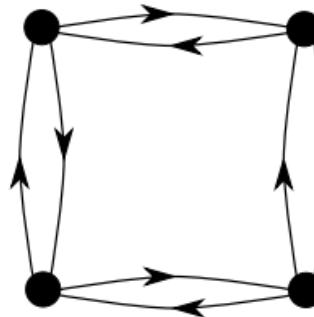
h) Digrafo NÃO assimétrico

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

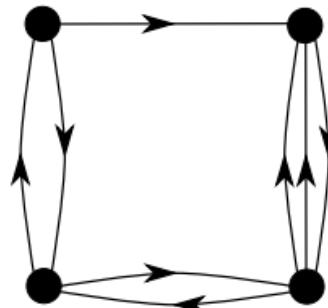
- **Tipos de Digrafos:**



i) Digrafo simétrico



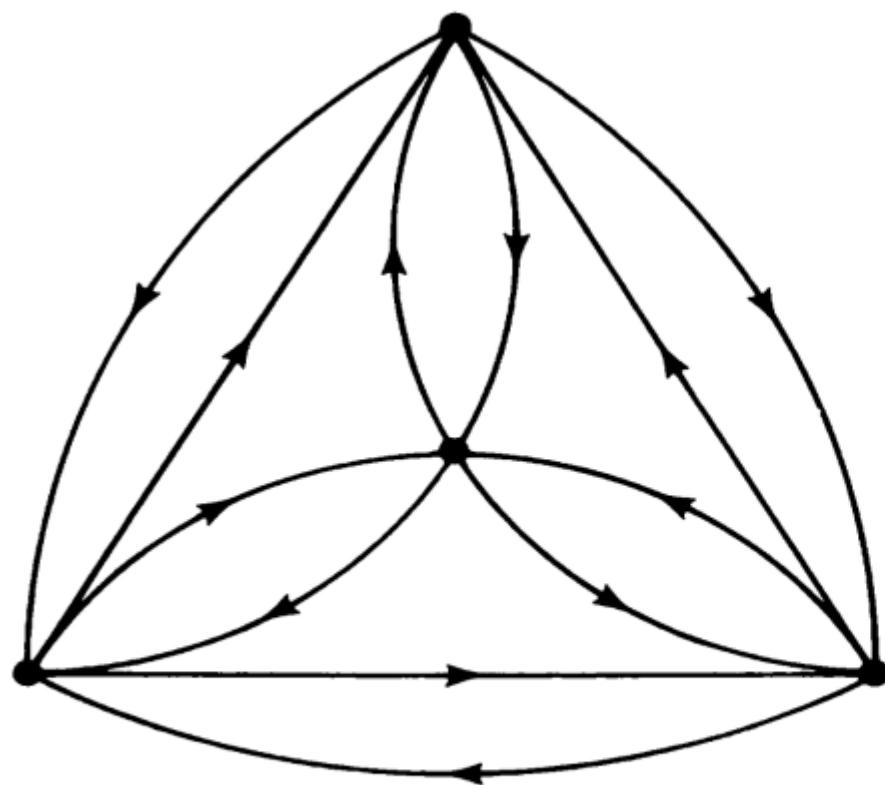
j) Digrafo Simetrico



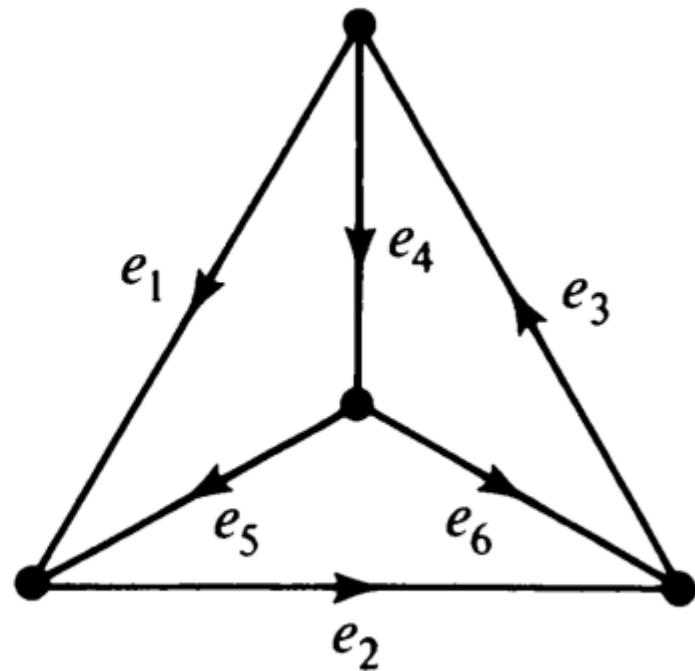
k) Digrafo NÃO simétrico

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Tipos de Digrafos:**



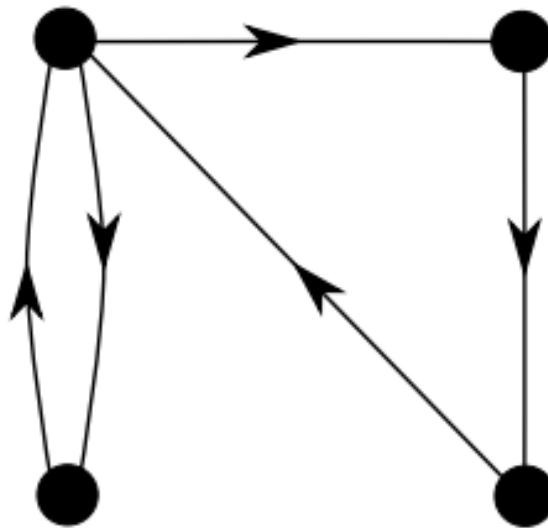
(I) Digrafo completo simétrico



(m) Digrafo completo assimétrico

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Tipos de Digrafos:**

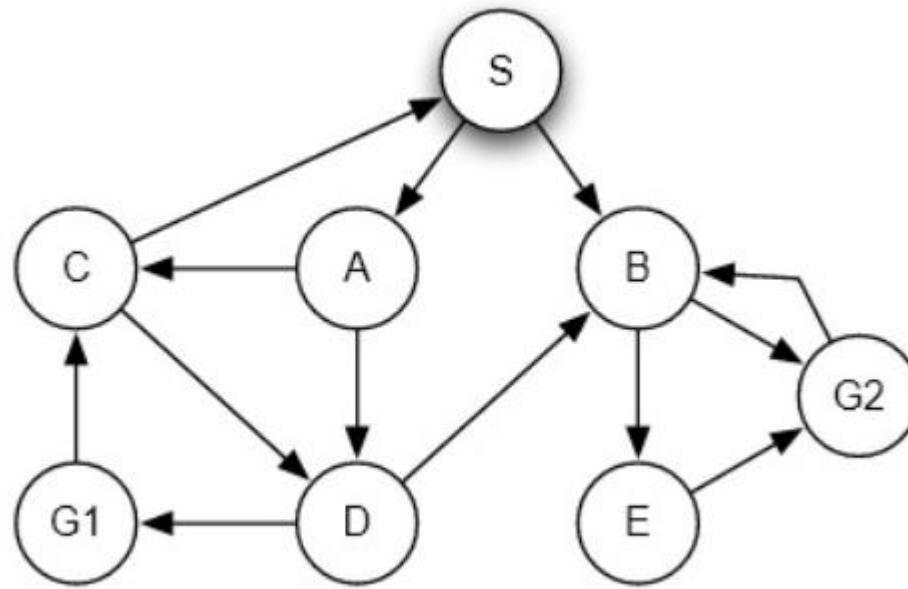


n) Digráfo Balanceado

# Teoria dos Grafos – Grafos Direcionados (Digrafos)

- **Teorema da Mão Dupla:** Para qualquer grafo direcionado, a **soma dos graus de entrada** de todos os vértices é **igual à soma dos graus de saída**, que é igual ao **número total de arestas** no grafo.
  - **ATENÇÃO:** Em grafos não direcionados simples, a soma do grau dos vértices era 2 vezes a quantidade de arestas, ou seja, cada aresta contava 2 vezes no grau do grafo, uma vez em cada vértice. Agora, cada aresta **uma vez** no **grau de entrada** e uma vez no **grau de saída**.
- As definições de **passeio**, **caminho**, e **ciclo** em grafos direcionados são as mesma as que vimos para grafos não direcionado, porém, sempre respeitando a direção da aresta

# Teoria dos Grafos – Grafos Direcionados (Digrafos)



# Teoria dos Grafos – Grafos Direcionados (Dígrafos)

- **Exercícios:**
  - a) Qual de entrada e saída de cada vértice?
  - a) G<sub>2</sub> é adjacente a E? E é adjacente a G<sub>2</sub>?
  - a) O Grafo possui arestas paralelas?
  - b) Mostre que um digrafo completo **assimétrico** com n vértices possui  $n(n - 1)/2$  arestas.
  - c) Mostre que um digrafo completo **simétrico** com n vértices possui  $n(n - 1)$  arestas.

# Teoria dos Grafos – **Grafos Direcionados (Digrafos)**

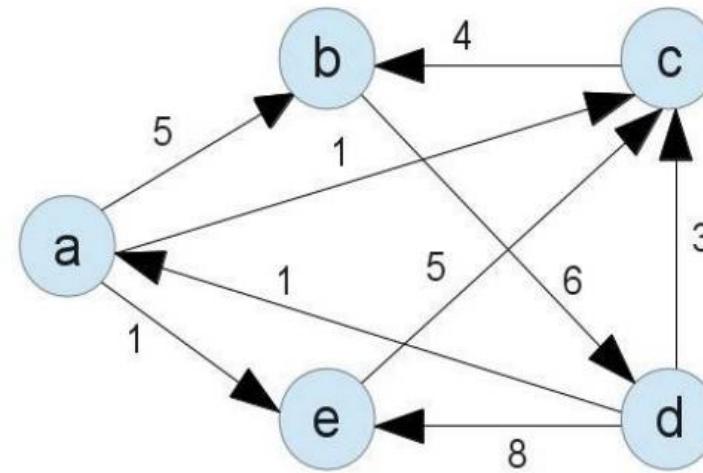
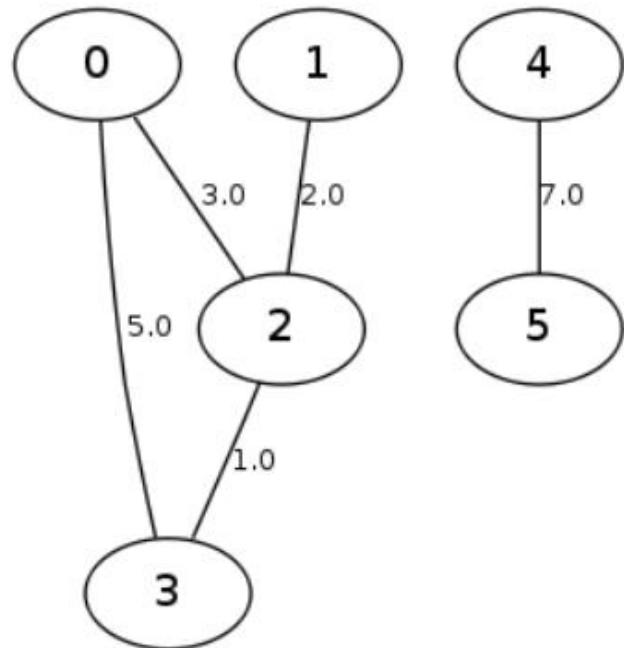
## **Grafos Valorados**

# Teoria dos Grafos – Grafos Direcionados (Dígrafos)

- **Grafo Valorado (ou Ponderado):** em um grafo ponderado cada aresta possui um valor numérico associado, conhecido como "peso".
  - Esses pesos podem representar diversos aspectos, como custo, distância, tempo, ou qualquer outra métrica que se adeque ao contexto do problema a ser modelado.
  - **Sistemas de Navegação e Mapas:** Para calcular o caminho com a menor distância entre dois pontos.
  - **Análise de Redes:** Para calcular o caminho que um pacote deve seguir (roteado) até seu destino.
  - **Engenharia Elétrica e Redes de Distribuição de Energia:** os vértices representam subestações ou geradores e as arestas, as conexões de transmissão. Os pesos podem representar a capacidade de carga ou a perda de energia, crucial para manter a estabilidade e eficiência da rede.

# Teoria dos Grafos – Grafos Direcionados (Dígrafos)

- **Grafo Valorado (ou Ponderado):** em um grafo ponderado cada aresta possui um valor numérico associado, conhecido como "peso".



- **Como representar um grafo valorado através de lista e matriz da adjacência??**

# Teoria dos Grafos – Grafos Direcionados (Dígrafos)

- **Grafo Valorado (ou Ponderado):** em um grafo ponderado cada aresta possui um valor numérico associado, conhecido como "peso".
  - Para representar um grafo ponderado usando **a matriz de adjacência**, basta usar o **peso** para indicar a ligação entre dois vértices. Ou seja, onde usámos 1 e 0(zero) para indicar (ou não) uma aresta, agora usamos o peso e o sinal de infinito  $\infty$  (para diferenciar um possível custo zero de uma aresta).

	A	B	C	D	E
A	--	5	--	--	1
B	--	--	--	6	--
C	--	4	--	--	--
D	1	--	3	--	8
E	--	--	5	--	--

# Teoria dos Grafos – Grafos Direcionados (Dígrafos)

- **Grafo Valorado (ou Ponderado):** em um grafo ponderado cada aresta possui um valor numérico associado, conhecido como "peso".
  - Para representar um grafo ponderado usando **a LISTA de adjacência**, basta representar as entradas da lista através de uma tupla (ou par), onde o primeiro elemento é o vértice adjacente e o **segundo elemento é o peso** da aresta que conecta esse vértice ao vértice original.

A-> (b,5) (e,1)

B-> (d,6)

C-> (b,4)

D-> (c,3) (a,1) (e,8)

E-> (c,5)

# Teoria dos Grafos – Grafos Direcionados (Dígrafos)

## Caminho com Menor Custo

# Teoria dos Grafos – Dijkstra

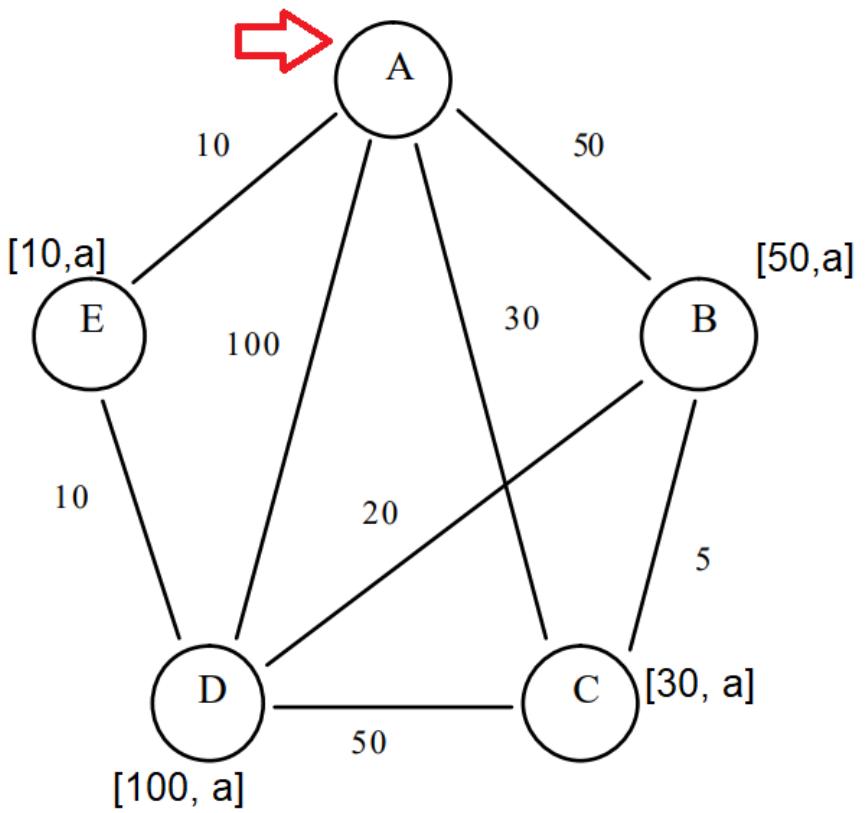
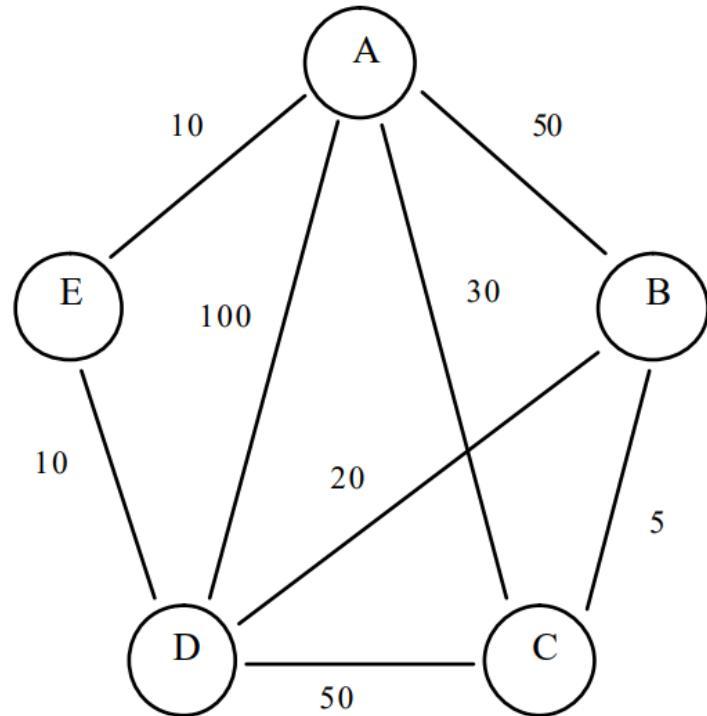
- **Algoritmo de Dijkstra:** O algoritmo de Dijkstra é um método eficiente para encontrar o **menor caminho (menor custo)** de um determinado vértice a todos os outros vértices em um grafo ponderado com **pesos não negativos**.
  - concebido pelo cientista da computação holandês Edsger Dijkstra em 1956 e publicado em 1959
  - Possui tempo computacional  **$O([m+n] \log n)$**  onde m é o número de arestas e n é o número de vértices
  - É amplamente utilizado em sistemas de roteamento, mapeamento de redes, logística, GPS, mapas etc...
- “Daisttra”, “Distra”, “Djastra”, Districa”

# Teoria dos Grafos – Dijkstra

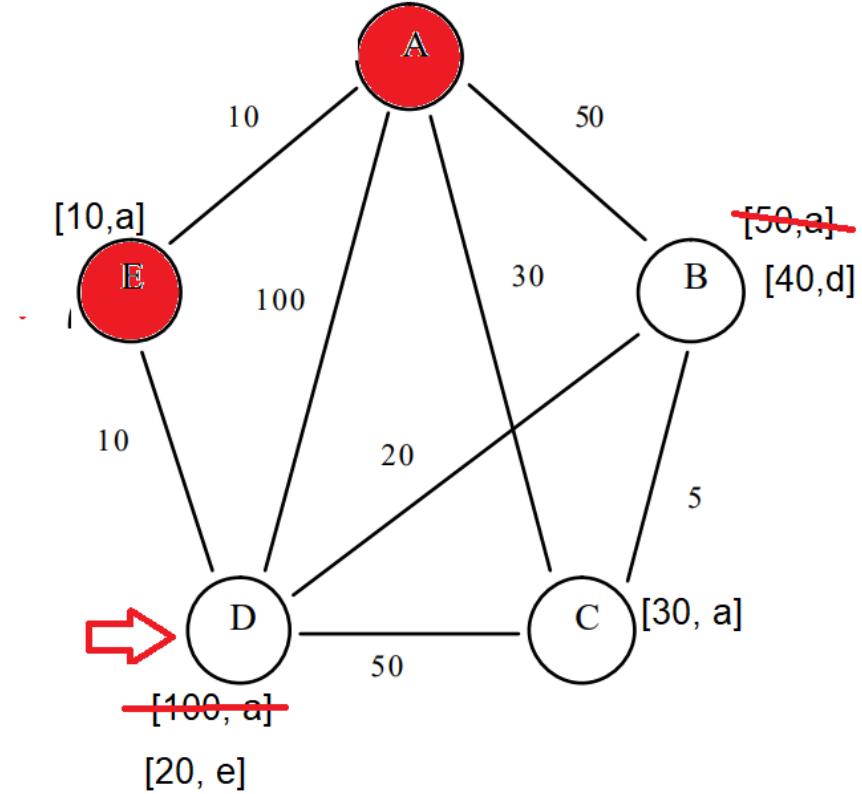
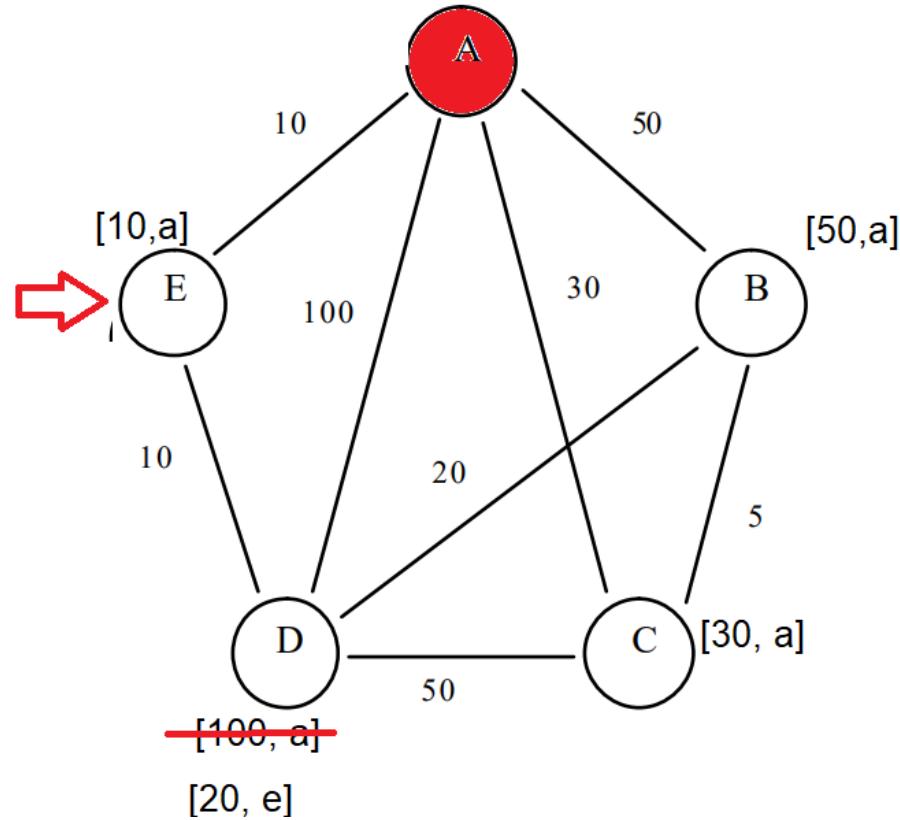
- **Algoritmo de Dijkstra:**

- i. Escolha um vértice de origem ( $v$ );
- ii. Para cada vértice adjacente a ( $v$ ) calcule a distância acumulada do vértice de origem até o mesmo.
- iii. Coloque uma etiqueta como o valor da distância e o vértice precedente em cada vértice analisado
- iv. Se a **distância calculada for menor** que a distância atual (se já houver uma) , atualize a etiqueta com os novos valores.
- v. Entre os **vértices não processado**, escolha o com a **etiqueta de menor valor**, marque o mesmo como processado e repita os passos de ii a v.

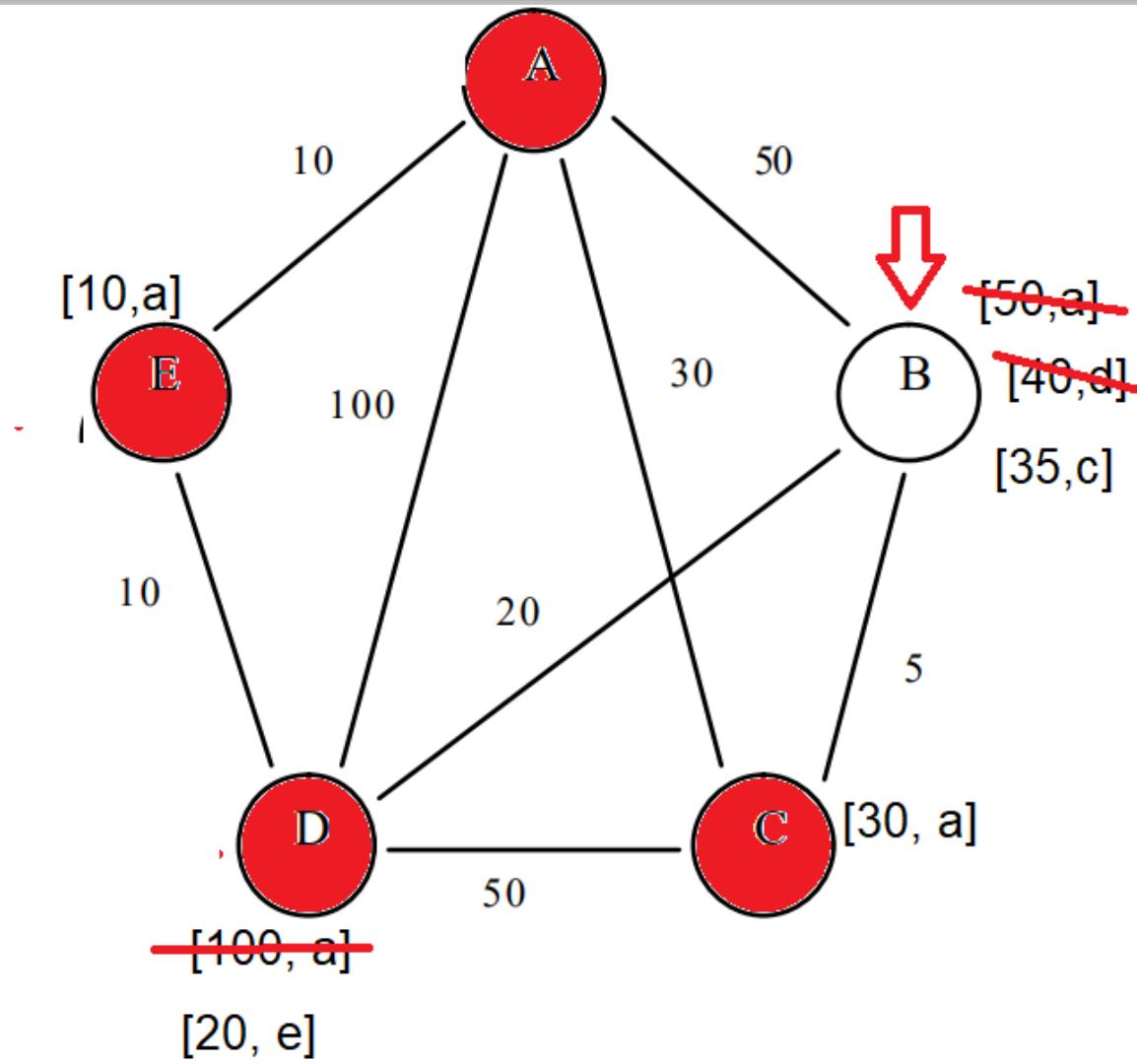
# Teoria dos Grafos – Dijkstra



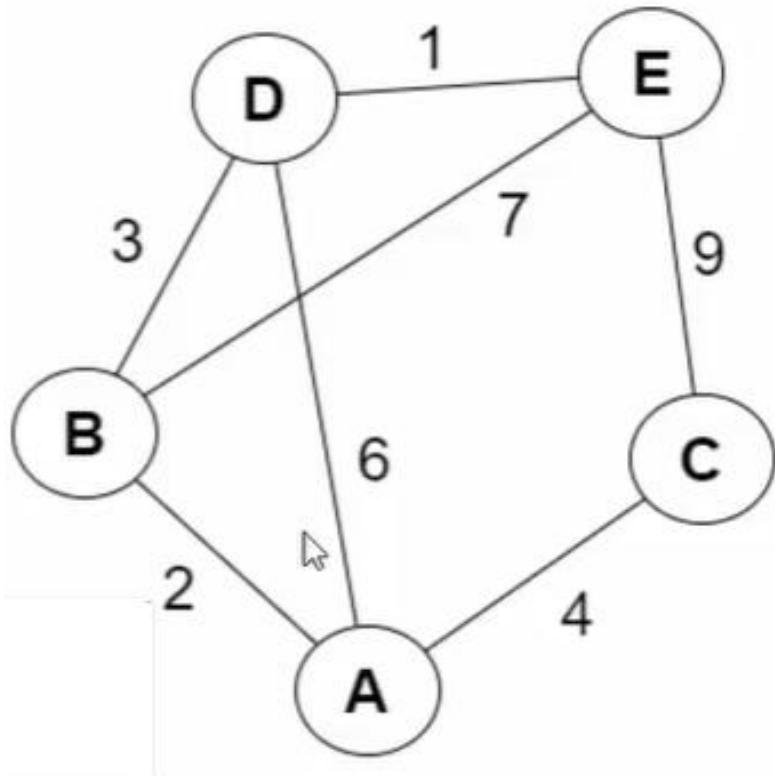
# Teoria dos Grafos – Dijkstra



# Teoria dos Grafos – Dijkstra

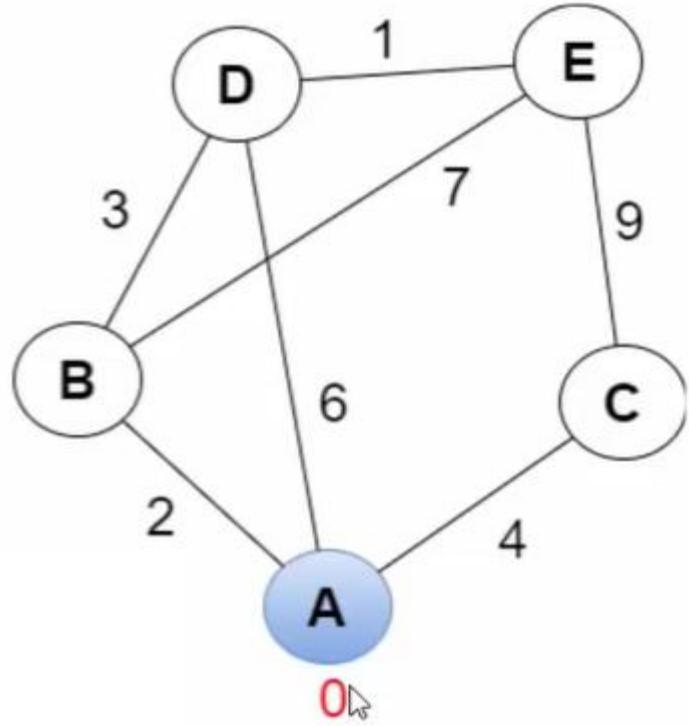


# Teoria dos Grafos – Dijkstra



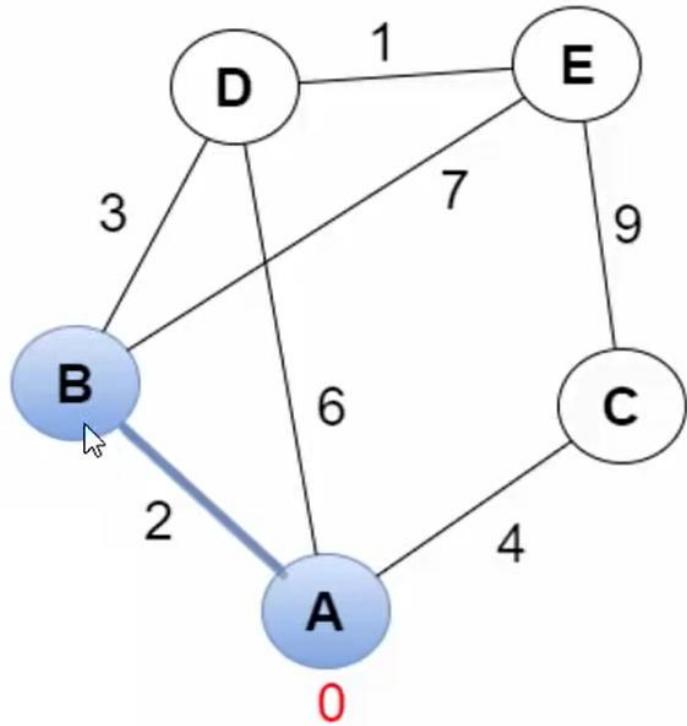
	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A					
B					
C					
D					
E					

# Teoria dos Grafos – Dijkstra



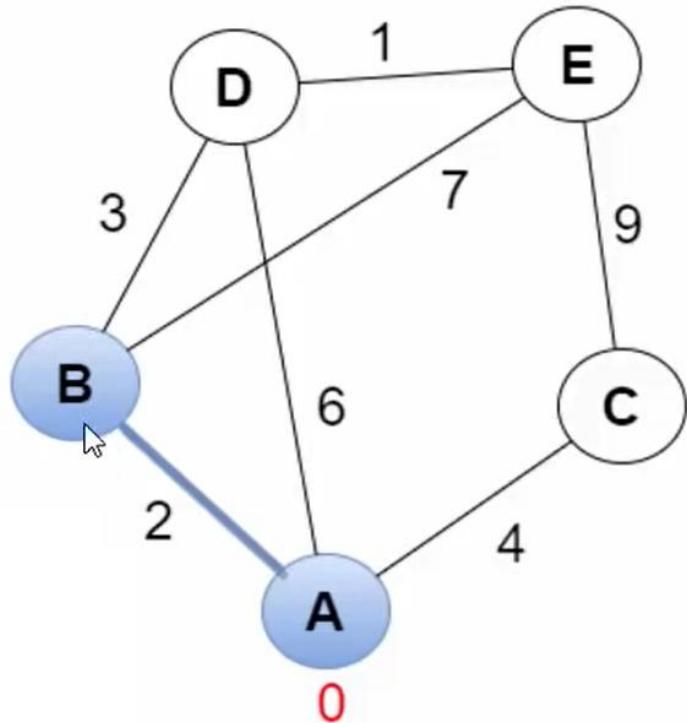
	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A	(0,A)	*	*	*	*
B	(2,A)				
C	(4,A)				
D	(6,A)				
E	-				

# Teoria dos Grafos – Dijkstra



	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A	(0,A)	*	*	*	*
B	(2,A)	(2,A)	*	*	*
C	(4,A)				
D	(6,A)				
E	-				

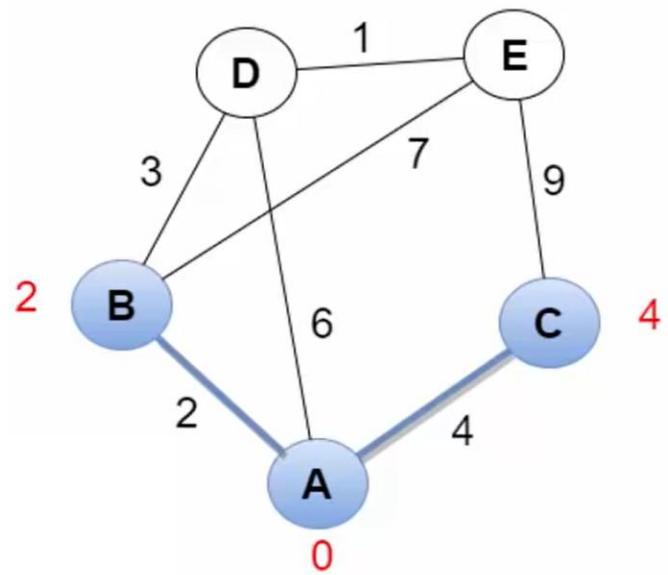
# Teoria dos Grafos – Dijkstra



	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A	(0,A)	*	*	*	*
B	(2,A)	(2,A)	*	*	*
C	(4,A)				
D	(6,A)				
E	-				

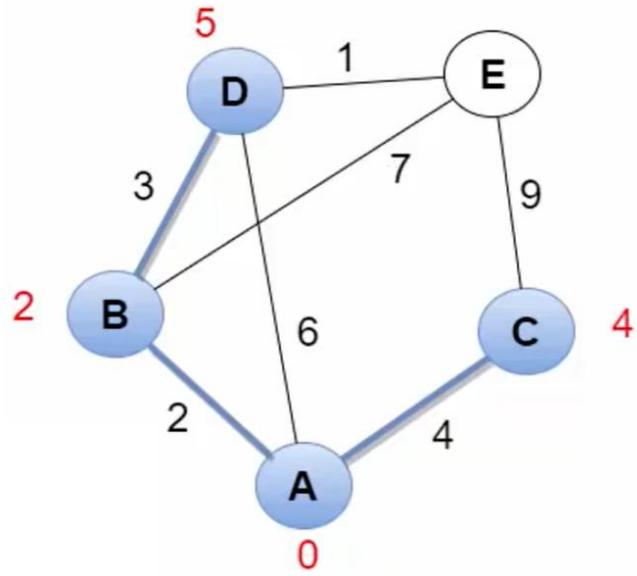
	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A	(0,A)	*	*	*	*
B	(2,A)	(2,A)	*	*	*
C	(4,A)	(4,A)			
D	(6,A)	(5,B)			
E	-	(9,B)			

# Teoria dos Grafos – Dijkstra



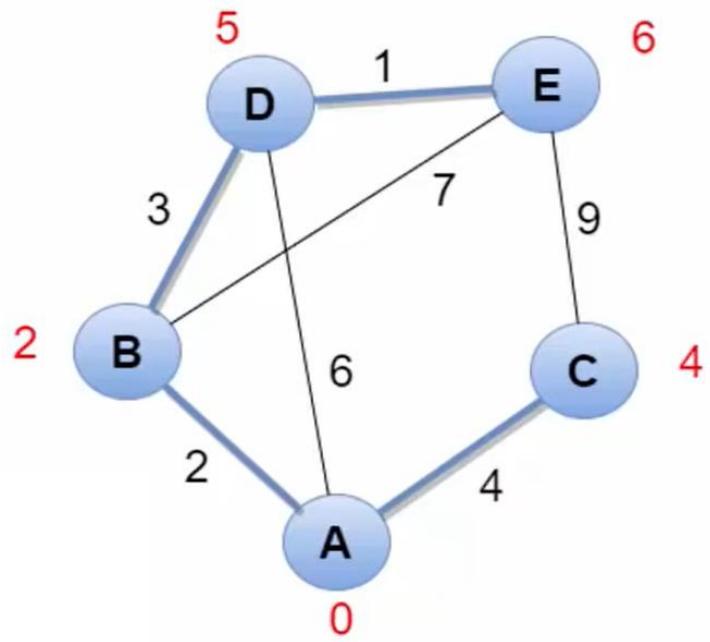
	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A	(0,A)	*	*	*	*
B	(2,A)	(2,A)	*	*	*
C	(4,A)	(4,A)	(4,A)	*	*
D	(6,A)	(5,B)	(5,B)		
E	-	(9,B)	(9,B)		

# Teoria dos Grafos – Dijkstra



	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A	(0,A)	*	*	*	*
B	(2,A)	(2,A)	*	*	*
C	(4,A)	(4,A)	(4,A)	*	*
D	(6,A)	(5,B)	(5,B)	(5,B)	*
E	-	(9,B)	(9,B)	(6,D)	

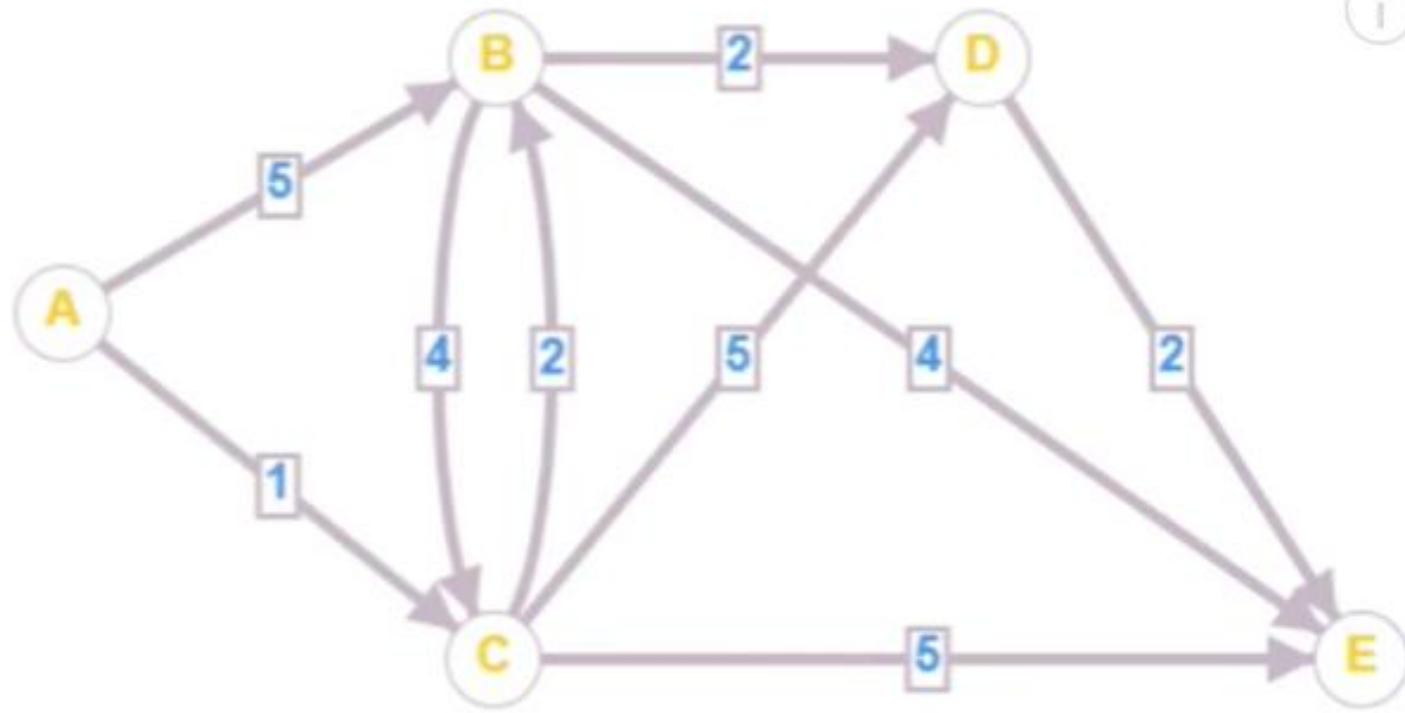
# Teoria dos Grafos – Dijkstra



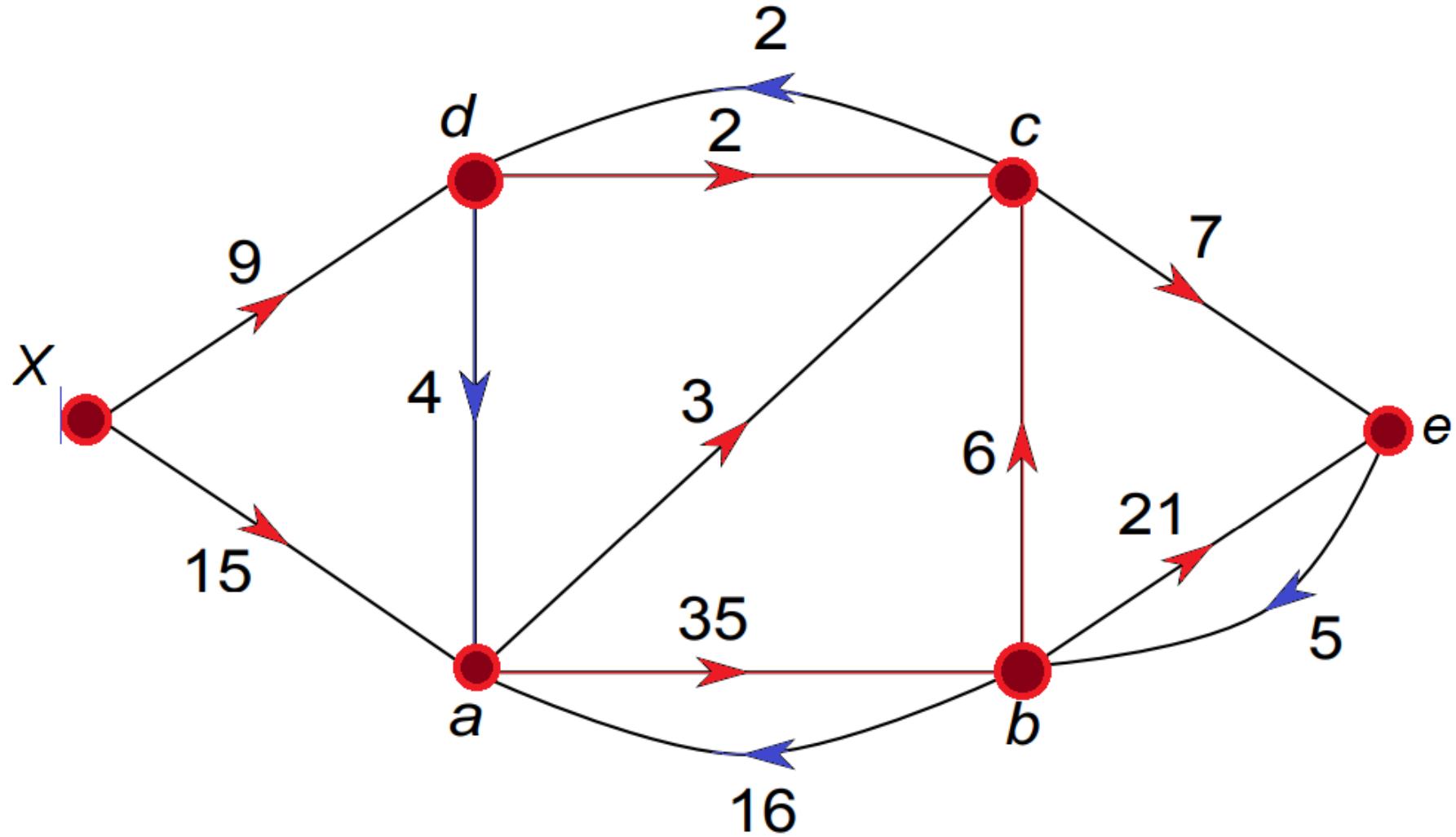
	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5
A	(0,A)	*	*	*	*
B	(2,A)	(2,A)	*	*	*
C	(4,A)	(4,A)	(4,A)	*	*
D	(6,A)	(5,B)	(5,B)	(5,B)	*
E	-	(9,B)	(9,B)	(6,D)	(6,D)

# Teoria dos Grafos – Dijkstra

i



# Teoria dos Grafos – Dijkstra



# Código de Huffman

**Código de Huffman**

# Código de Huffman

- Principais conceitos:
  - **Código de Huffman:** É uma técnica de compressão (um compactador) que utiliza códigos de tamanho variável para representar os símbolos do texto.
    - A ideia básica do algoritmo é atribuir códigos de bits menores para os símbolos mais frequentes no texto, e códigos mais longos para os mais raros.
    - É uma técnica de compressão **sem perda**

# Código de Huffman

- Principais conceitos:
  - Código de Huffman:
    - **Compressão sem perda (Lossless):** Reduz o tamanho do arquivo **sem perder qualquer informação**. Isso significa que, ao descomprimir o arquivo, você recupera uma cópia exata dos dados originais.
      - É um **processo reversível**
      - ZIP ou PNG (para imagens)
    - **Compressão com Perda (Lossy):** Reduz o tamanho do arquivo **eliminando** partes dos dados que são consideradas menos importantes.
      - Implica perda de dados
      - É um **processo irreversível**. Uma vez que os dados foram removidos durante a compressão, eles não podem ser recuperados na descompressão. JPEG, Mp3,

# Código de Huffman



# Código de Huffman

Dec	Chr	Dec	Chr
64	@	96	`
65	A	97	a
66	B	98	b
67	C	99	c
68	D	100	d
69	E	101	e
70	F	102	f
71	G	103	g
72	H	104	h
73	I	105	i
74	J	106	j
75	K	107	k
76	L	108	l
77	M	109	m
78	N	110	n
79	O	111	o
80	P	112	p
81	Q	113	q
82	R	114	r
83	S	115	s
84	T	116	t
85	U	117	u
86	V	118	v
87	W	119	w
88	X	120	x
89	Y	121	y
90	Z	122	z
91	[	123	{
92	\	124	
93	]	125	}
94	^	126	~
95		127	DEL

- Principais conceitos:

- Código de Huffman:

- bom esse bombom

Char	ASCII	Binário
b	98	0110 0010
o	111	0110 1111
m	109	0110 1101
e	101	0110 0101
s	115	0111 0011
Espaço	32	0010 0000

98 111 109 32 101 115 115 101 32 98 111 109 98 111 109

# Código de Huffman

- Principais conceitos

- Código de Huffman:

- bom esse bombom

Char	ASCII	Binário
b	98	0110 0010
o	111	0110 1111
m	109	0110 1101
e	101	0110 0101
s	115	0111 0011
Espaço	32	0010 0000

b

o

m

--

e

0110 0010 0110 1111 0110 1101 0010 0000 0110 0101

s

s

e

--

b

0111 0011 0111 0011 0110 0101 0010 0000 0110 0010

o

m

b

o

m

0110 1111 0110 1101 0110 0010 0110 1111 0110 1101.

# Código de Huffman

- Principais conceitos:

- Código de Huffman:

- bom esse bombom

Char		Binário
b	0	000
o	1	001
m	2	010
e	3	011
s	4	100
Espaço	5	101

**45 bits para  
armazenar o texto!**

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

000 001 010 101 011 100 100 011

1 101 000 001 010 000 001 010.

# Código de Huffman

- Principais conceitos:
  - **Limitações deste método:** Com 3 bits, eu consigo representar somente 8 símbolos / letras ( $2^3$ ), logo, a eficácia depende da quantidade de símbolos do meu domínio.
  - O algoritmo de Huffman contorna este problema usando um número variável de bits para representar cada símbolo, onde:
    - **Menos bits** são utilizados para caracteres que aparecem mais vezes no texto
    - **Mais bits** são utilizados para caracteres que aparecem menos vezes no texto.

# Código de Huffman

## ■ Passo a passo:

- **Contar a Frequência:** Inicialmente, contamos a frequência de cada caractere (ou outro tipo de dado) no conjunto de dados a ser comprimido.
- **Criar Folhas da Árvore:** Cada caractere é tratado como uma folha de árvore e é criada uma lista onde essas folhas são inseridas de acordo com as suas frequências. Os caracteres menos frequentes estão no início da lista.
- **Construir a Árvore de Huffman:** Enquanto houver mais de um nó na fila de prioridade:
  - Remova os dois nós do inicio da lista (menor frequência).
  - Crie um novo nó com esses dois nós como filhos. A frequência deste novo nó será a soma das frequências de seus filhos.
  - Insira o novo nó de volta na fila de prioridade.

# Código de Huffman

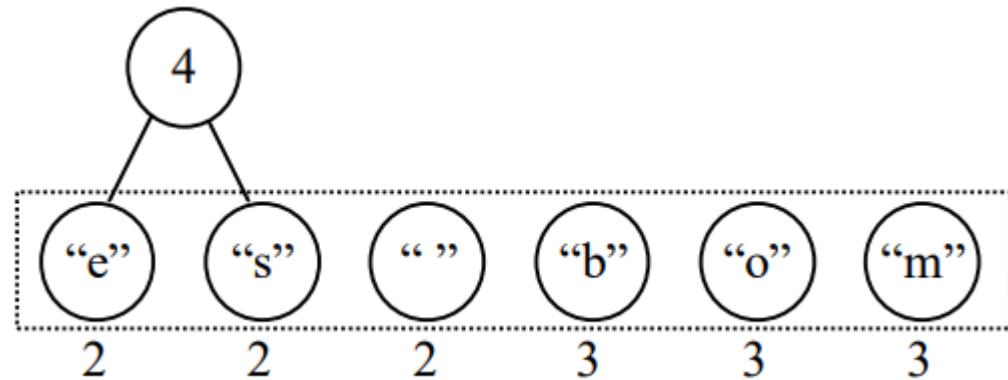
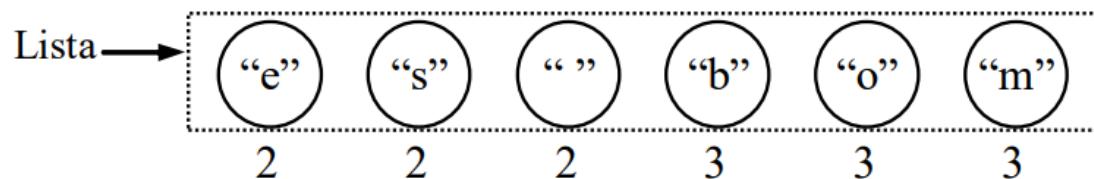
- **Passo a passo:**
  - **Gerar Códigos:** Uma vez que a árvore está construída, atribui-se um código a cada caractere. Isso é feito percorrendo a árvore da raiz até cada folha:
    - Atribua '0' para o caminho que vai para a esquerda e '1' para o caminho que vai para a direita.
  - **Codificar os Dados:** Use os códigos gerados para substituir cada caractere pelos seus respectivos códigos binário.

# Código de Huffman

- Passo a passo:

Char	
b	3
o	3
m	3
e	2
s	2
Espaço	2

Lista →

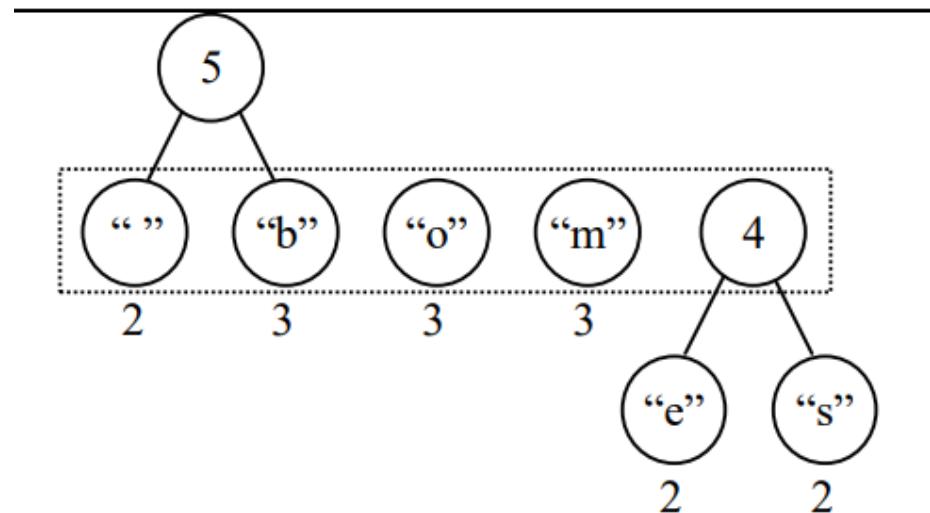
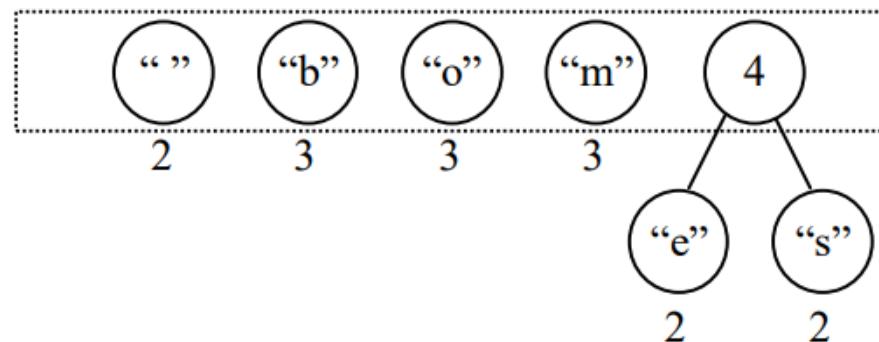


# Código de Huffman

- Passo a passo:

Char	
b	3
o	3
m	3
e	2
s	2
Espaço	2

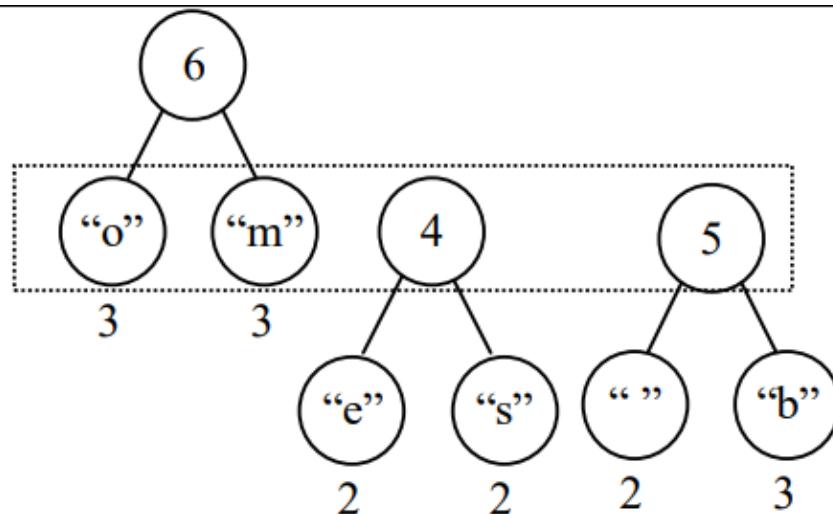
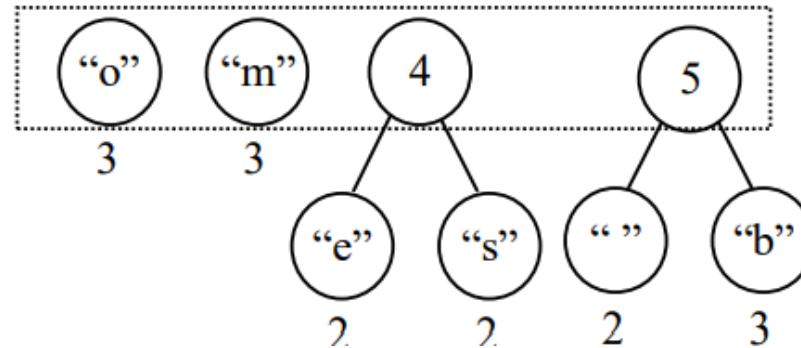
- Insiro o novo nó na lista , mantendo a mesma ordenada e repito a operação



# Código de Huffman

- Passo a passo:

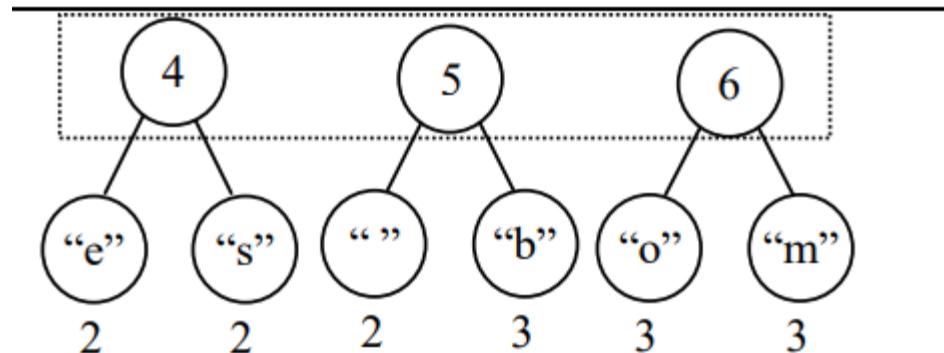
Char	
b	3
o	3
m	3
e	2
s	2
Espaço	2



# Código de Huffman

- Passo a passo:

Char	
b	3
o	3
m	3
e	2
s	2
Espaço	2

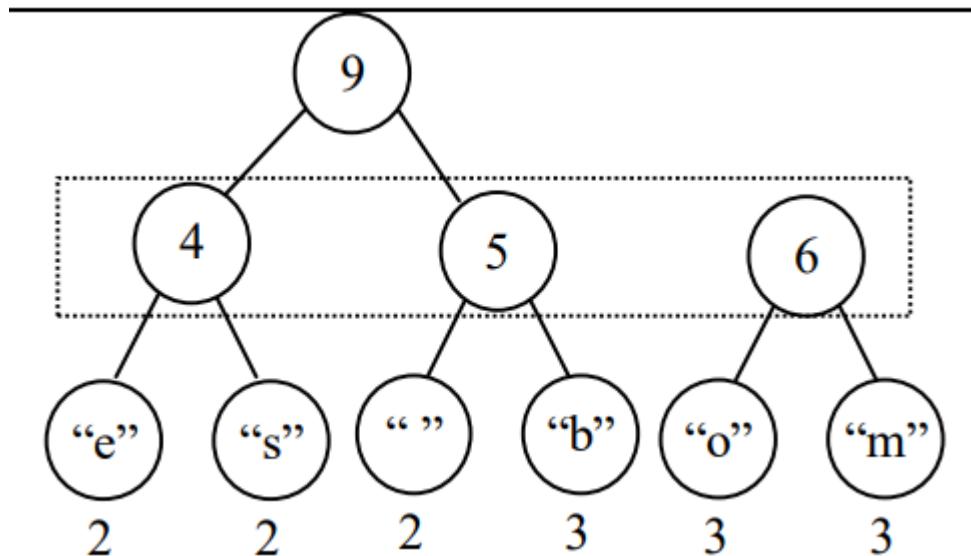


# Código de Huffman

- Passo a passo:

Char	
b	3
o	3
m	3
e	2
s	2
Espaço	2

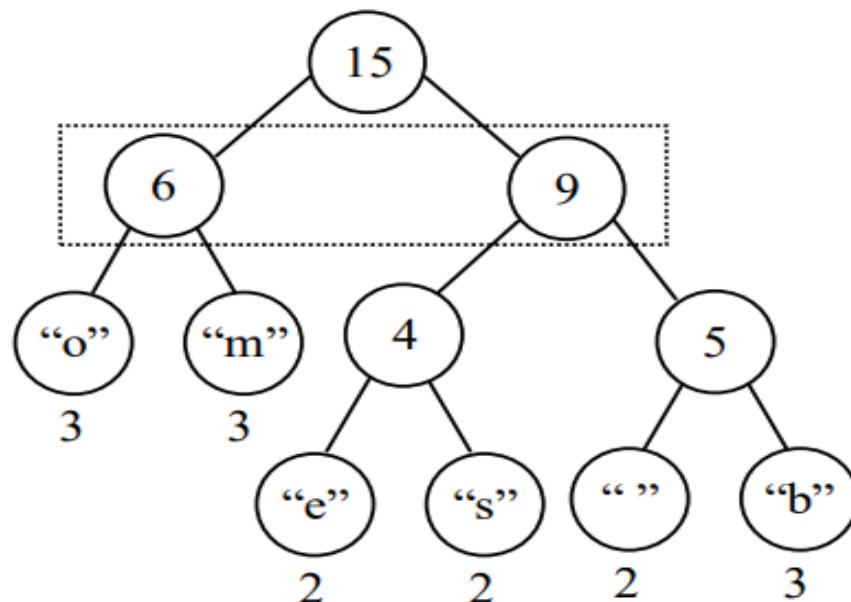
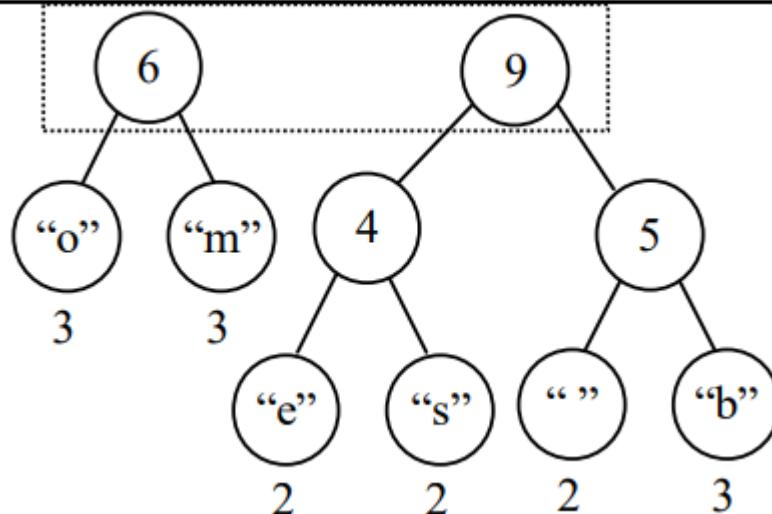
- Continuo repetindo a operação até ter somente uma arvore.
- Note que agora estou “juntando” arvore com mais de um nó.



# Código de Huffman

- Passo a passo:

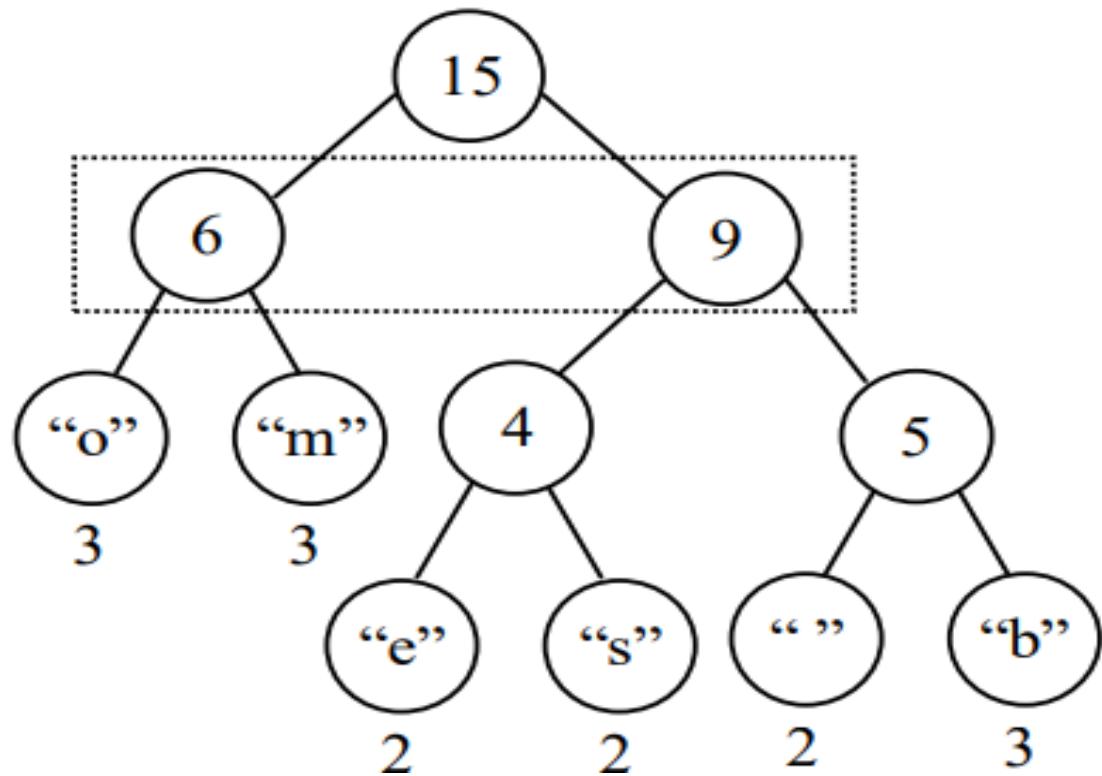
Char	
b	3
o	3
m	3
e	2
s	2
Espaço	2



# Código de Huffman

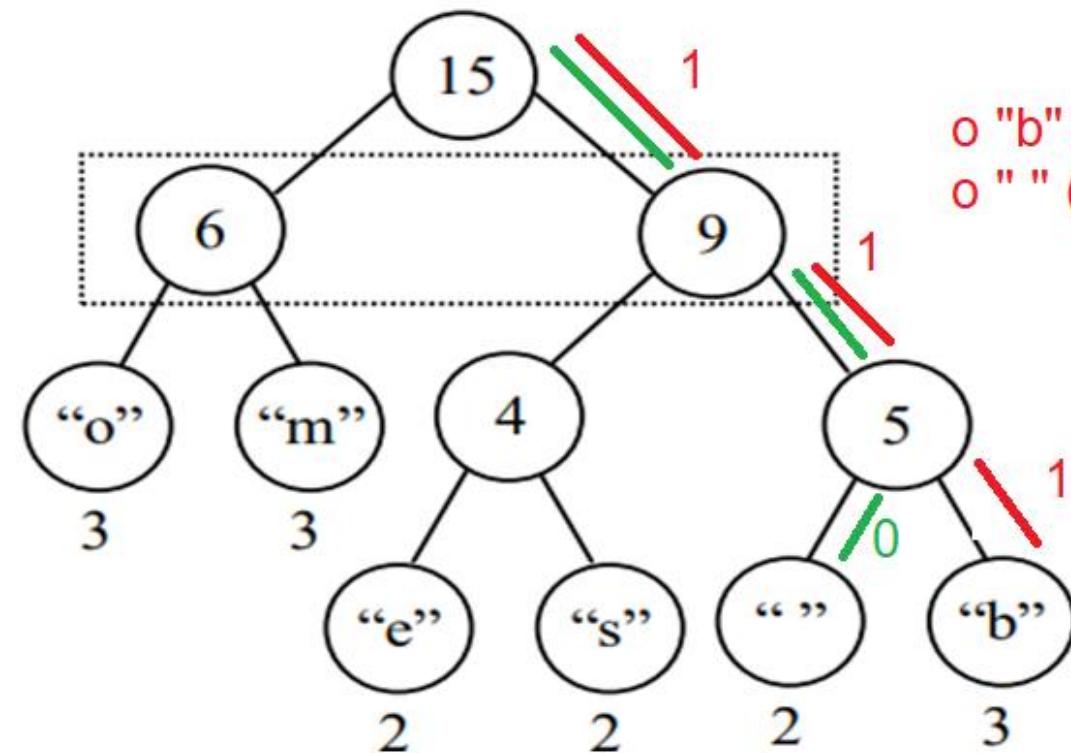
- Passo a passo:

Char	
b	3
o	3
m	3
e	2
s	2
Espaço	2



# Código de Huffman

- Passo a passo:
  - Com a arvore de codificação completa, gero os códigos fazendo o caminho até cada caracter.
    - Para cada nó:
    - Considero 0 (zero) para arcos da esquerda
    - 1 (um) para arcos da direita.



o "b" será codificado como 111  
o " " (espaco) sera codificado como 110

Char	Binário
b	111
o	00
m	01
e	100
s	101
Espaço	110

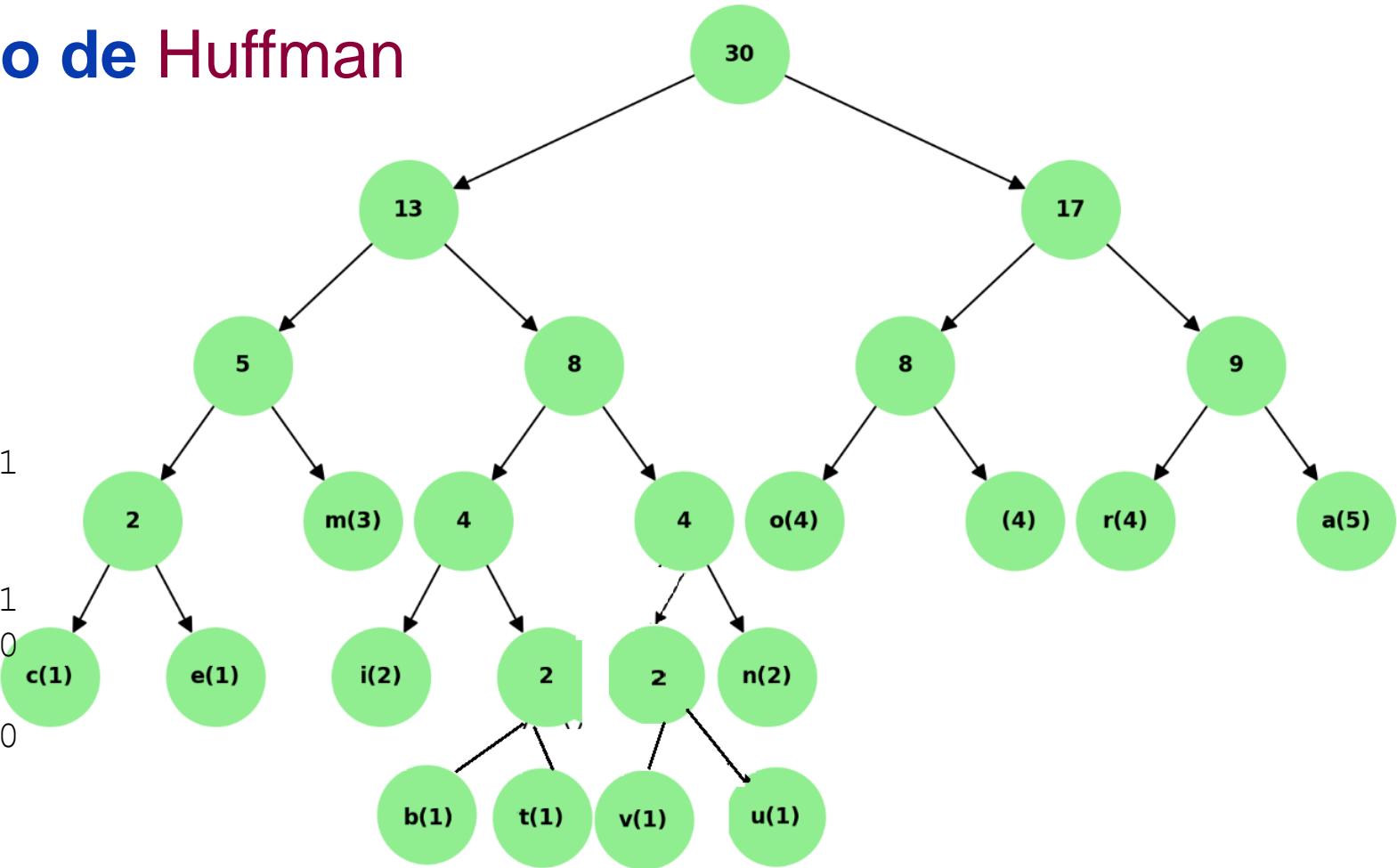
111 00 01 110 100 101 101 100 110 111 00 01 111 00 01

**Apenas 39 bits para armazenar o texto!!!**

- Crie a arvore de Huffman, a tabela de códigos e a codificação final da frase:
  - ***como montar uma arvore binaria***

# Código de Huffman

c	(1)	:	0000
o	(4)	:	100
m	(3)	:	001
'	'(4)	:	101
n	(2)	:	0111
t	(1)	:	01011
a	(5)	:	111
r	(4)	:	110
u	(1)	:	01101
v	(1)	:	01100
e	(1)	:	0001
b	(1)	:	01010
i	(2)	:	0100



000010000110010100110001110101111111

010101101001111101111110011001001100

14:28:09

0011010101001000111111100100111

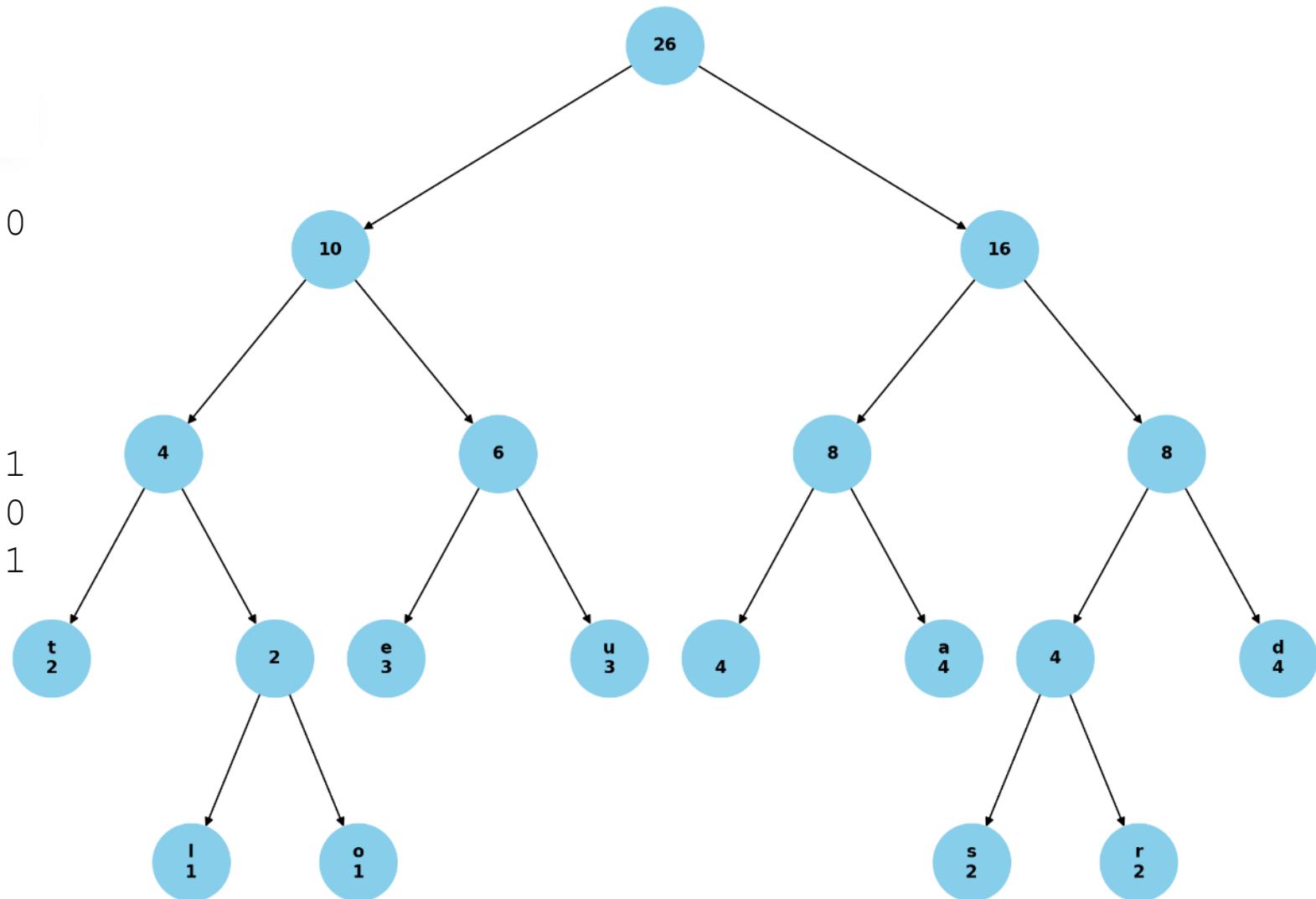
# Código de Huffman

- **Descompactação:**

# Código de Huffman

t:	1101	1111100110110100111001100100000100
s:	1100	10001110011111010010111001101101000
a:	101	
u:	1110	101111001111110011011001111101101111
e:	1111	010011101000101111000101010101000110
d:	010	
r:	10011	0
l:	01101	
o:	1000	
é:	10010	
ã:	01100	
n:	01110	
m:	01111	
“ ”	00	

**t:** 000  
**s:** 1100  
**a:** 101  
**u:** 011  
**e:** 010  
**d:** 111  
**r:** 1101  
**l:** 0010  
**o:** 0011  
**" "** 100



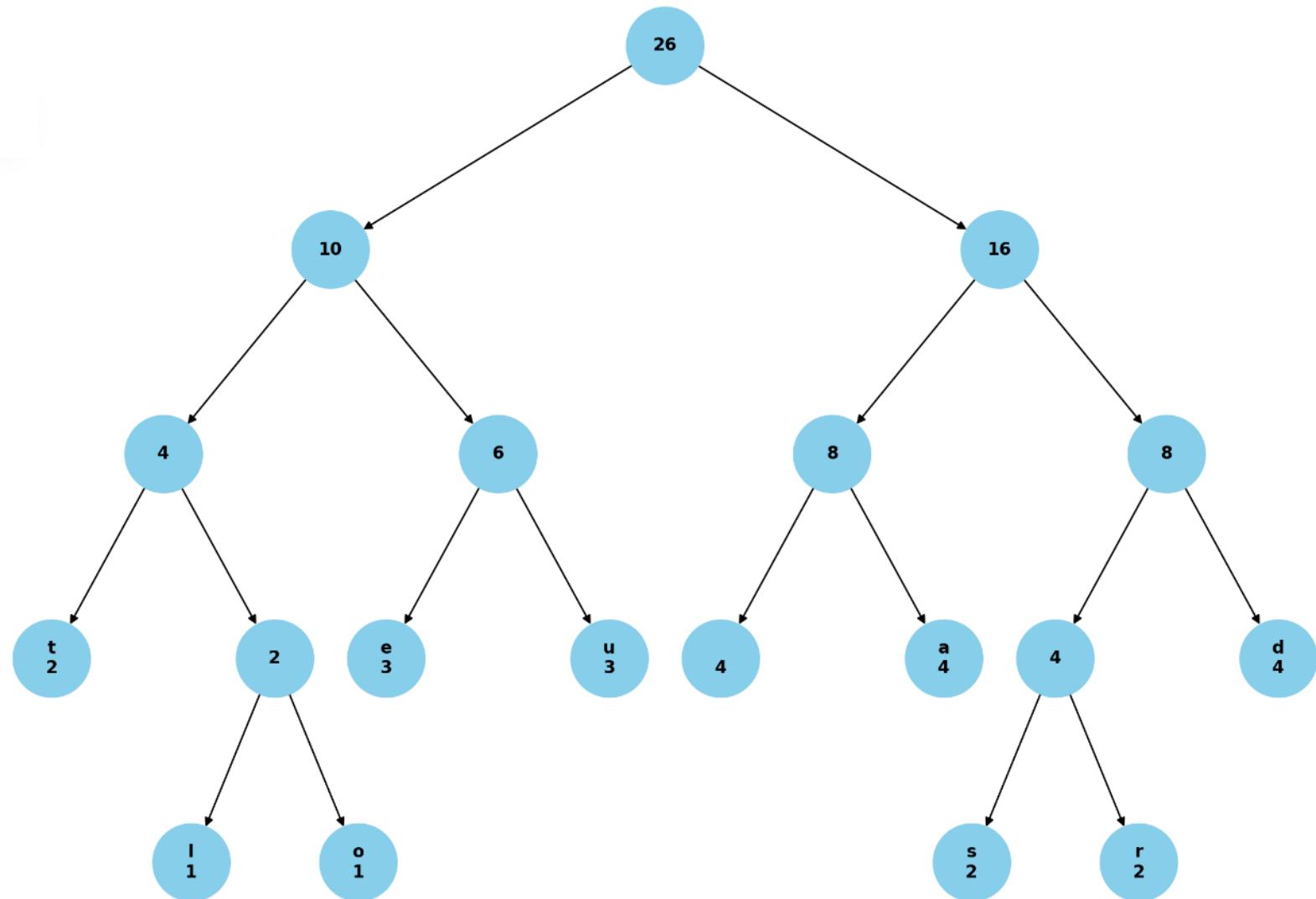
101011001010110011101010001011000001

101011000011110110110011101010011110

14:28:09

111100111100

t: ??  
s: ??  
a: ??  
u: ??  
e: ??  
d: ??  
r: ??  
l: ??  
o: ??  
" " ??



101011001010110011101010001011000001

101011000011110110110011101010011110

14:28:09

111100111100