Pesquisa e Ordenação de Dados

Unidade 6:

Compressão de dados

Compressão de dados

- A quantidade de dados sempre tende a crescer de maneira exponencial
- Compressão de dados
  - Codificação da informação usando menos bits do que na representação original
  - Mudança na representação de algum dado para reduzir seu tamanho
- Objetivo da compressão
  - Reduzir o espaço de armazenamento
  - Reduzir o tempo de transmissão

Tipos de compressão

- Compressão sem perdas
  - remoção (recuperável) das redundâncias
  - aplicada a textos, dados, programas
- Algoritmos clássicos:
  - Huffmann
  - Run-length
  - LZW (Lempel-Ziv-Welch)

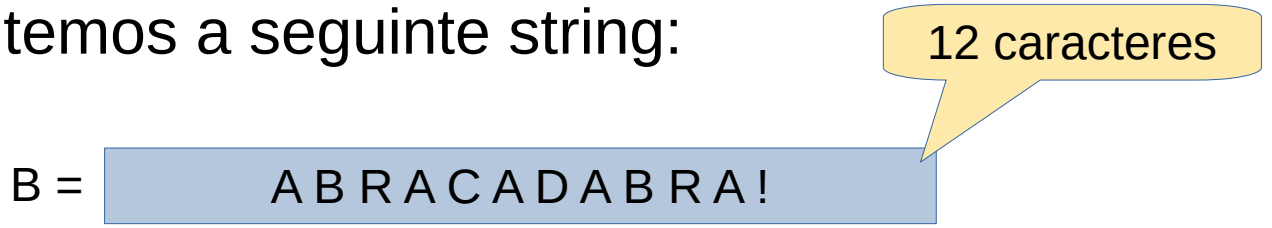
- Compressão com perdas
  - eliminação de detalhes
  - aplicada a imagens, áudios, vídeos, streams…
  - Exemplos:
    - JPEG
    - MP3
    - MP4

Compressão sem perda

- Compress:
  - gera uma representação comprimida C(B) a partir de um conjunto B de bits de entrada
- Expand:
  - reconstrói o dado binário original B
- Compress ratio (Taxa de compressão): C(B) / B

Exemplo

- Suponha que temos a seguinte string:



- Representação em ASCII:
  - 12 x 8 bits = **96 bits**

Exemplo

- Suponha que temos a seguinte string:

12 caracteres

B = A B R A C A D A B R A !

- Tabela de codificação com menos bits:
  - 12 x 3 bits = **36 bits**
  - C(B) = 010011110010100010101010011110010001
  - Códigos de comprimento fixo: a cada 3 bits sabemos que temos 1 caractere

Exemplo de codificação com 3 bits:

!	001
A	010
B	011
C	100
D	101
R	110

Exemplo

- Suponha que temos a seguinte string:

12 caracteres

B = A B R A C A D A B R A !

- Tabela de códigos com comprimento variável:
  - C(B) = 0100110011110010101001100 1111
    A B R A C A D A B R A !
  - **28 bits!**
  - Requisito: códigos livres de prefixo

!	1111
A	0
B	100
C	1110
D	101
R	110

Exemplo

- Codificação com comprimento variável:
  - Compressão:

    ABRACADABRA! → 01001100111001010100110011111

  - Expansão:

    01001100111001010100110011111 → ABRACADABRA!

Tabela de códigos

!	1111
A	0
B	100
C	1110
D	101
R	110

Tabela de símbolos

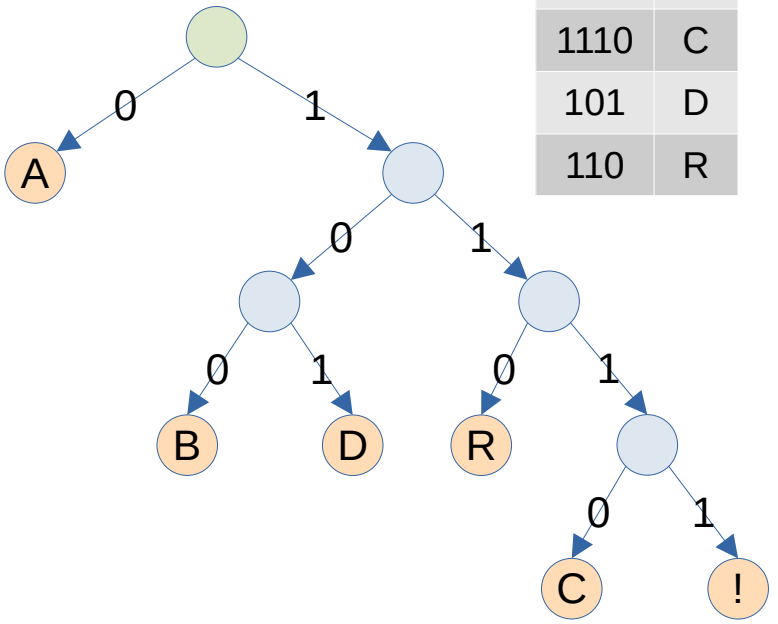
1111	!
0	A
100	B
1110	C
101	D
110	R

Codificação - Trie

- A tabela de símbolos pode ser representada de forma bastante eficiente por uma Trie binária
  - Caracteres ficam nas folhas
  - O código de um caractere é representado pelo percurso a partir da raiz até a respectiva folha
  - Arestas:
    - esquerda: 0
    - direita: 1

Tabela de símbolos

1111	!
0	A
100	B
1110	C
101	D
110	R

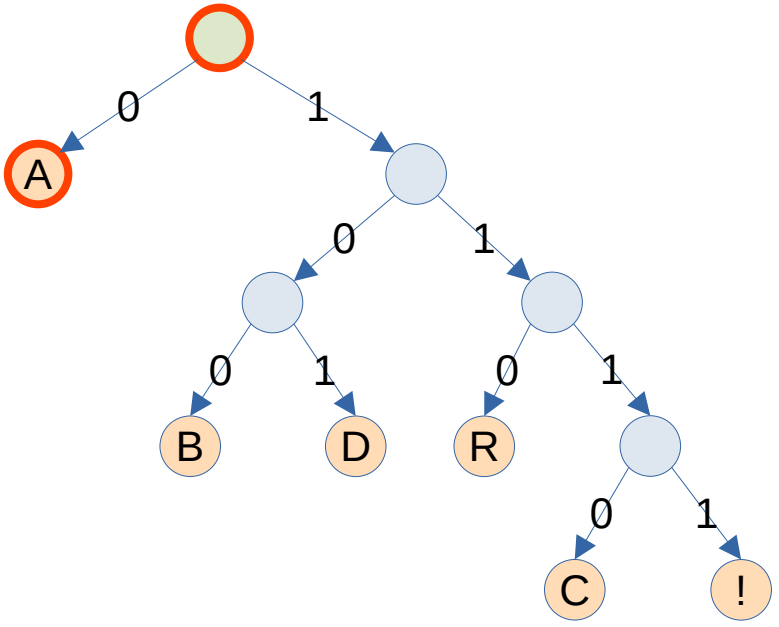


Codificação - Trie

- Expansão:

  01001100111001010100110011111

  - A

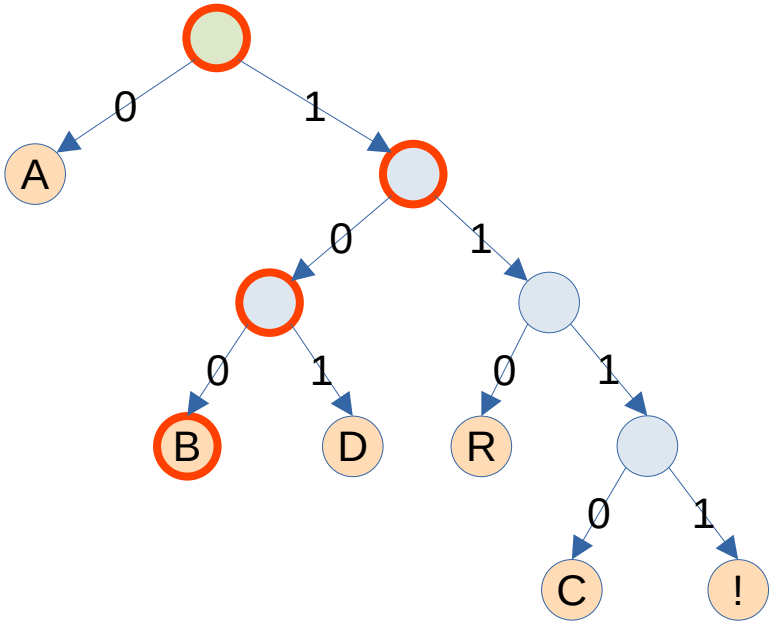


Codificação - Trie

- Expansão:

  0**100**1100111001010100110011111

  - AB

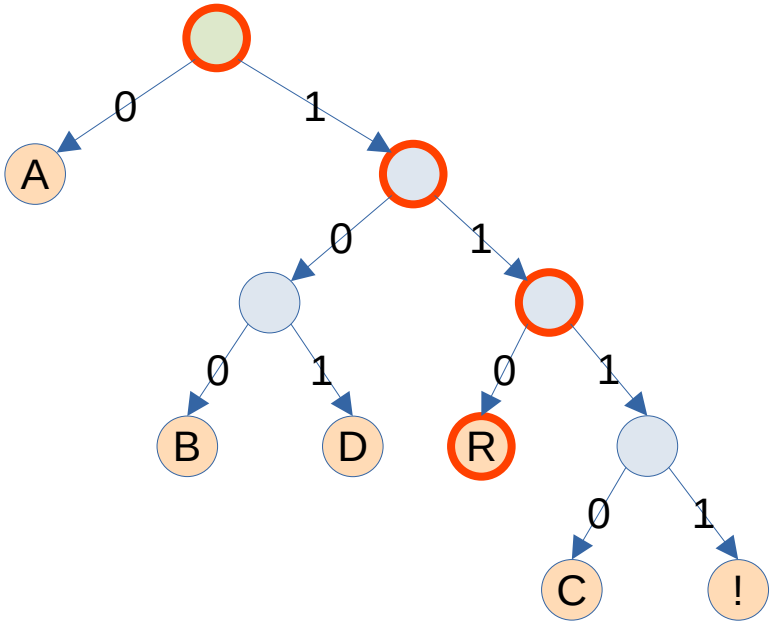


Codificação - Trie

- Expansão:

  01001100111001010100110011111

  - ABR

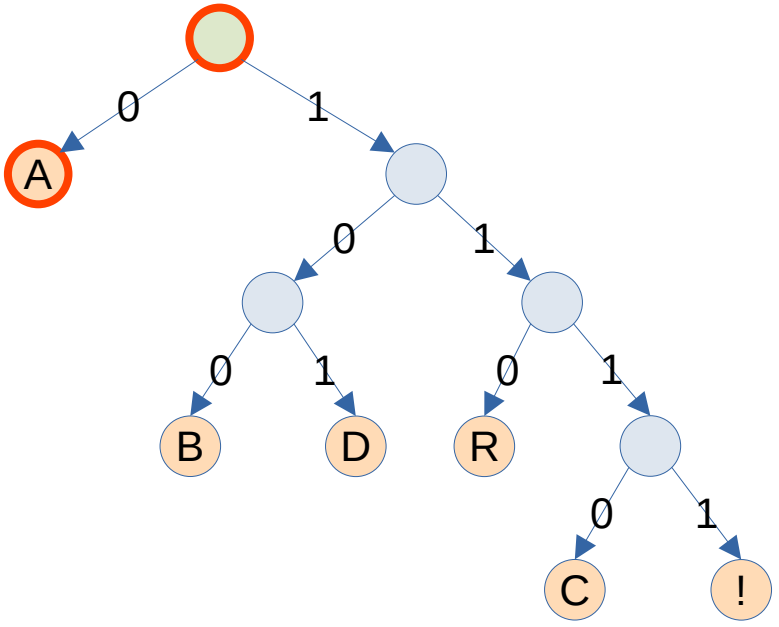


Codificação - Trie

- Expansão:

  0100110**0**1110010101001100111**1**

  - ABRA

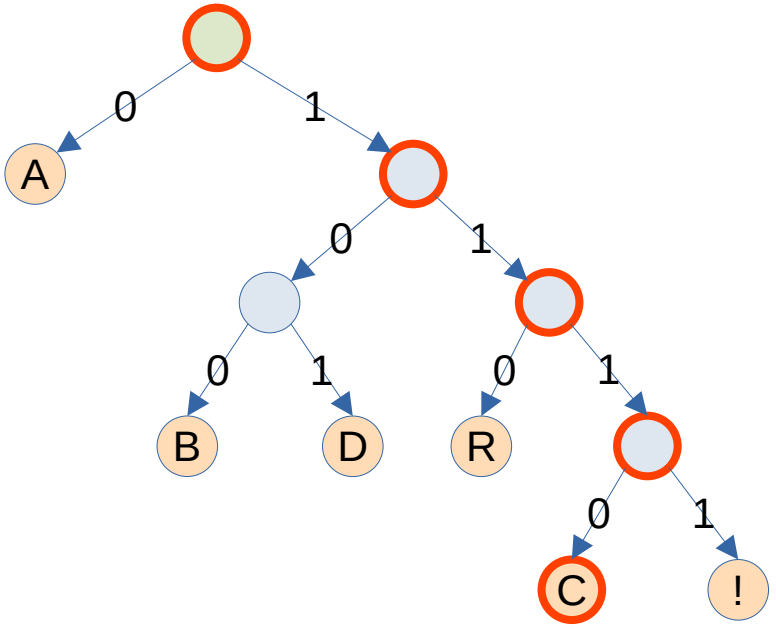


Codificação - Trie

- Expansão:

  01001100**1110**01010100110011111

  - ABRAC

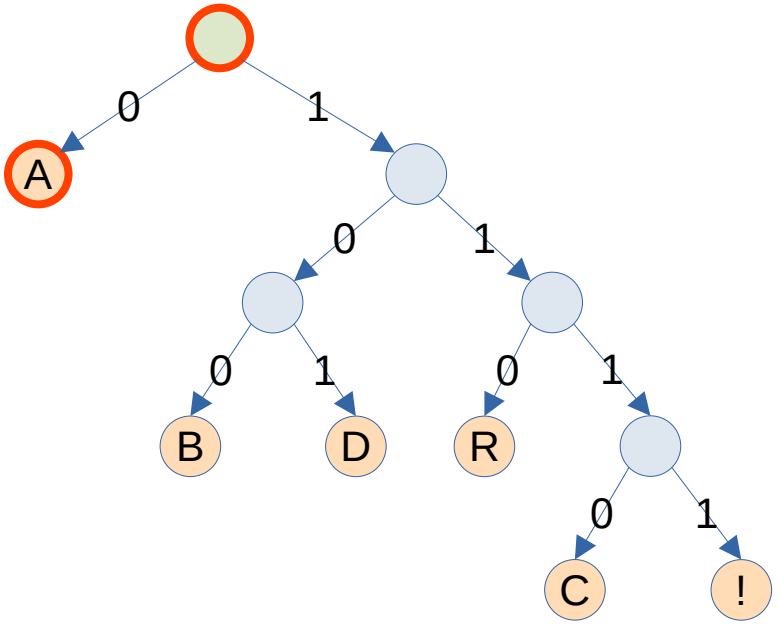


Codificação - Trie

- Expansão:

  01001100111001010100110011001111

  - ABRACA

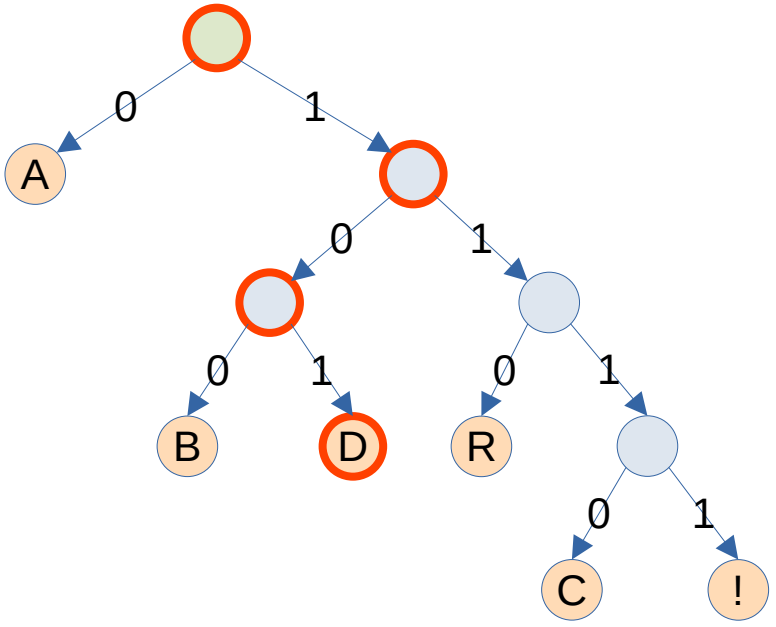


Codificação - Trie

- Expansão:

  01001100111001**01**0100110011111

  - ABRACAD

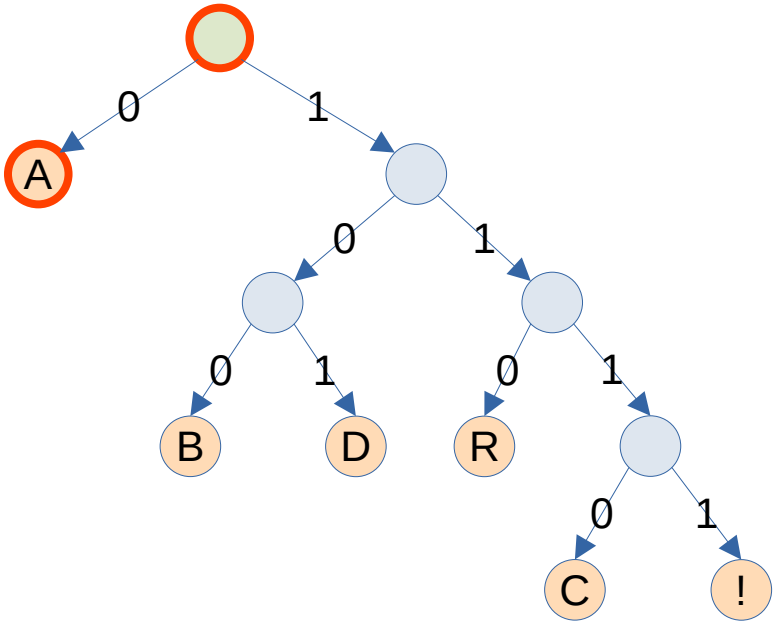


Codificação - Trie

- Expansão:

  01001100111001010100110011111

  - ABRACADA

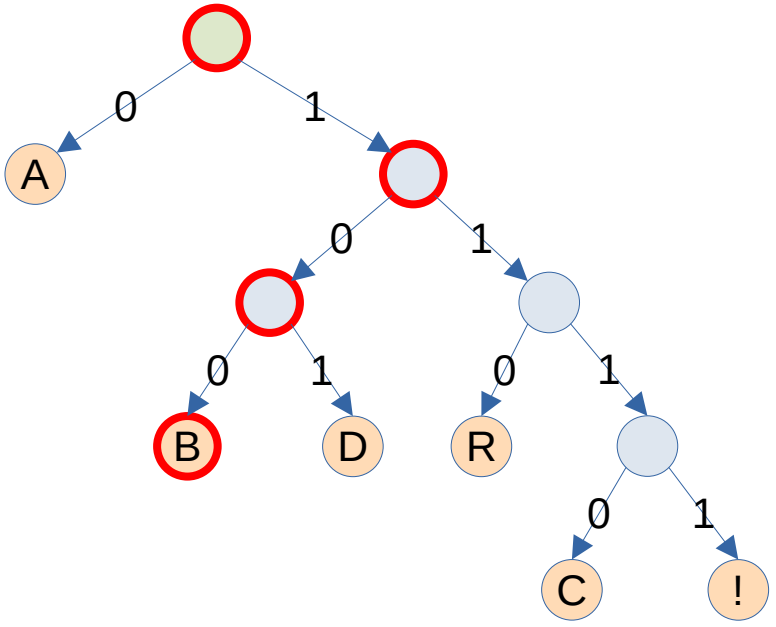


Codificação - Trie

- Expansão:

  01001100111001010**100**11001111

  - ABRACADAB

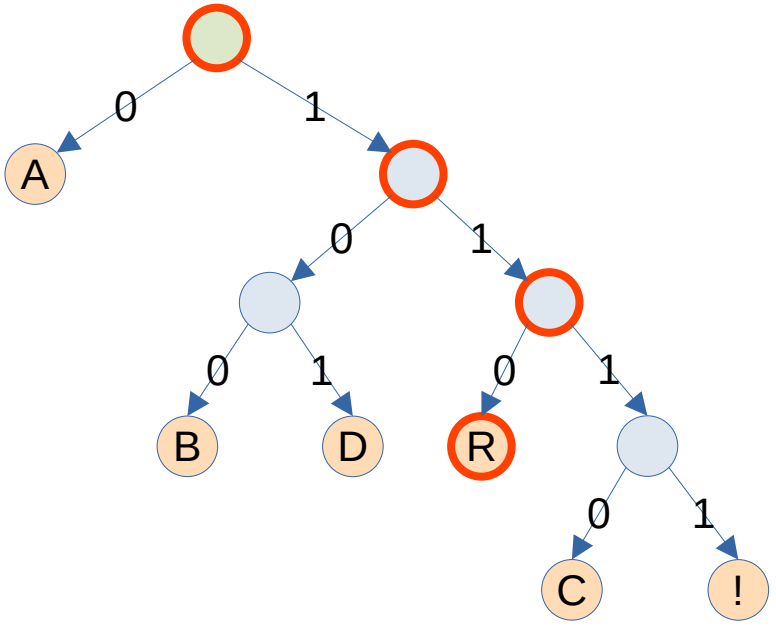


Codificação - Trie

- Expansão:

  01001100111001010100**110**01111

  - ABRACADABR

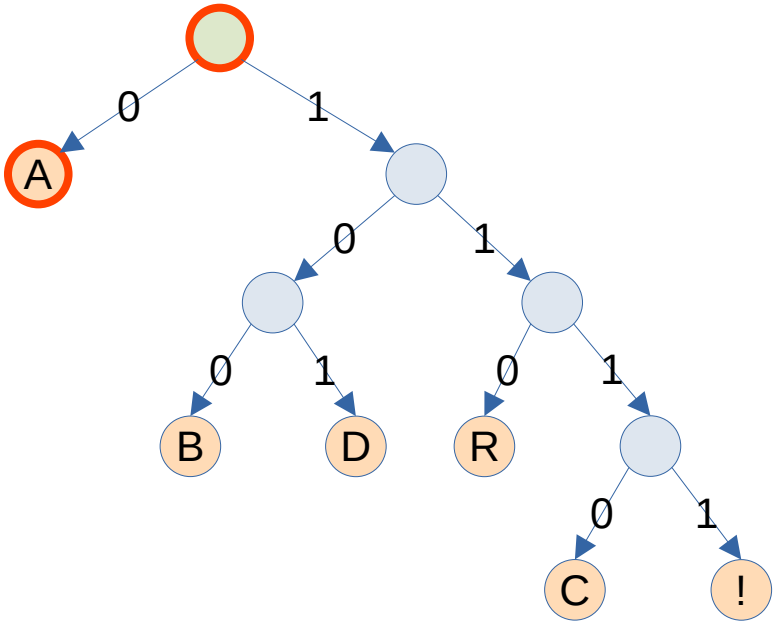


Codificação - Trie

- Expansão:

  01001100111001010100110**0**1111

  - ABRACADABRA

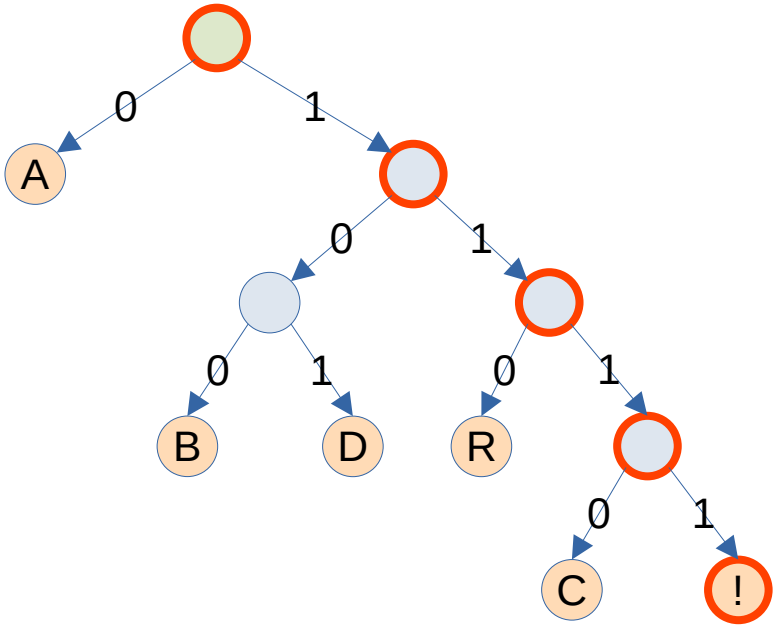


Codificação - Trie

- Expansão:

  01001100111001010100110**1111**

  - ABRACADABRA!



Codificação - Trie

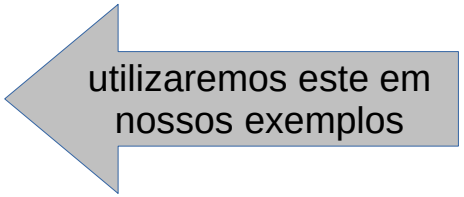
- Como encontrar o melhor comprimento para a codificação de cada símbolo, a fim de reduzir o tamanho da mensagem codificada?

Algoritmo de Huffman!

constrói a trie ótima, com códigos livres de prefixo

Codificação de Huffman

- Desenvolvida por David Huffman em 1951 durante seu doutorado no MIT.
- Algoritmo baseado em símbolos
  - entrada: string
    - símbolos: caracteres da string
  - entrada: texto:
    - símbolos: palavras do texto
  - entrada: sequência de bits
    - símbolos: conjuntos de 8 bits (ou qualquer quantidade fixa de bits)
    - torna possível comprimir qualquer conteúdo (texto, música, vídeo, etc)



Codificação de Huffman

- Ideia geral: codificar os símbolos em uma forma binária com o menor número de dígitos possível
- A codificação de cada símbolo depende de sua frequência na entrada:
  - símbolos frequentes: códigos curtos
  - símbolos menos frequentes: códigos longos
- *Prefix rule*: nenhum código é prefixo de outros códigos
  - assim, não há ambiguidade no momento da descompressão
- Complexidade de tempo: O(n log n)

Codificação de Huffman

- Passos:
  - Contar a frequência de todos os caracteres no texto de entrada;
  - Montar um floresta (várias árvores), onde cada árvore é unitária e contém, além do caractere em si, seu número de ocorrências no texto;
  - Ordenar a lista de árvores em ordem crescente de frequência;
  - Unir todas as árvores até formar um única árvore:
    - Selecionar as duas árvores que possuem as menores frequências
    - Cria uma nova árvore combinando as duas árvores obtidas anteriormente
      - A frequência desta árvore será a soma das frequências das árvores que foram unidas
    - Realocar esta árvore no conjunto de árvores, mantendo a ordenação

Exemplo Huffman

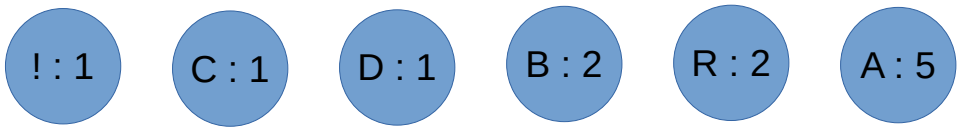
- Contar a frequência de todos os caracteres no texto de entrada

A B R A C A D A B R A !

Caractere	Frequência
!	1
A	5
B	2
C	1
D	1
R	2

Exemplo Huffman

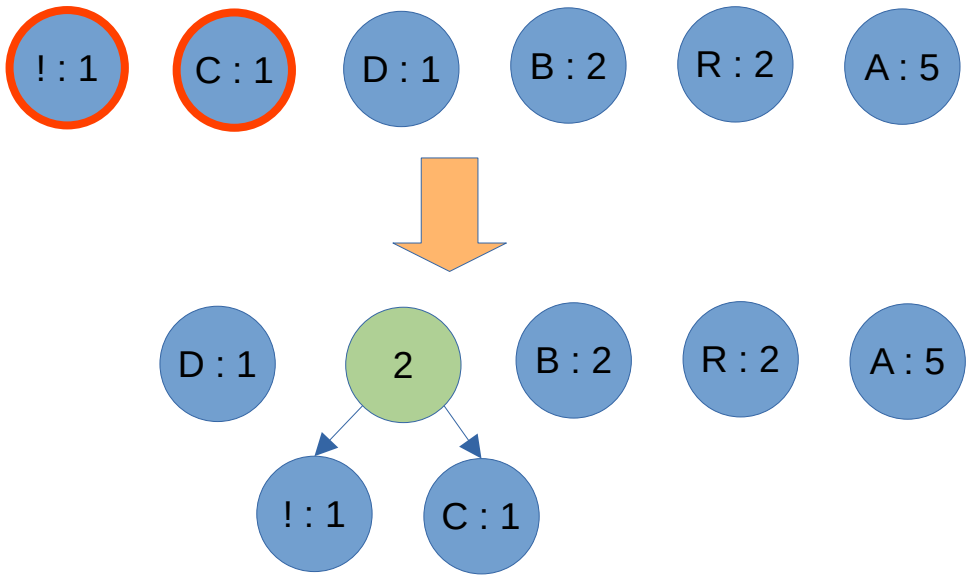
- Montar um conjunto de árvores unitárias
- Colocar em ordem crescente de frequência



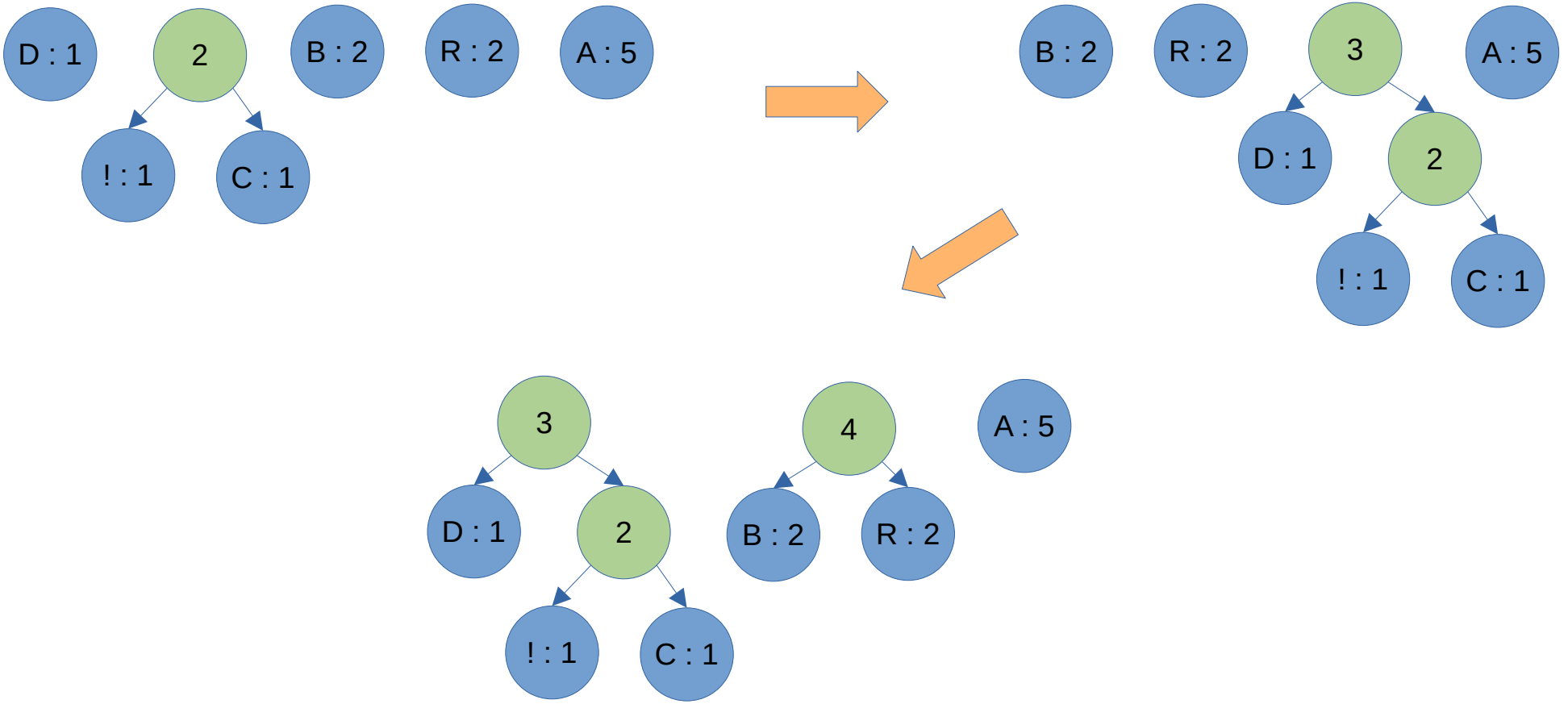
Caractere	Frequência
!	1
A	5
B	2
C	1
D	1
R	2

Exemplo Huffman

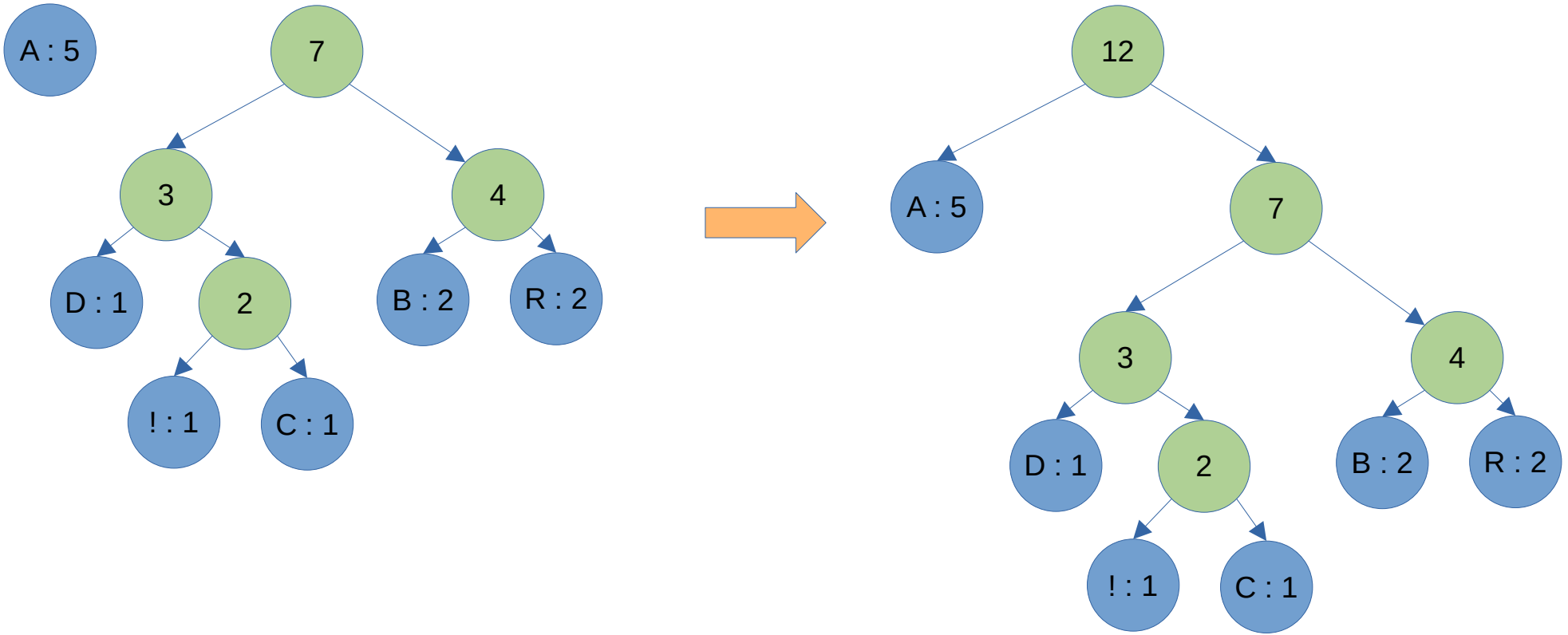
- Unir as 2 árvores com menor frequência. O nó pai armazenará a soma. Reordenar.



Exemplo Huffman

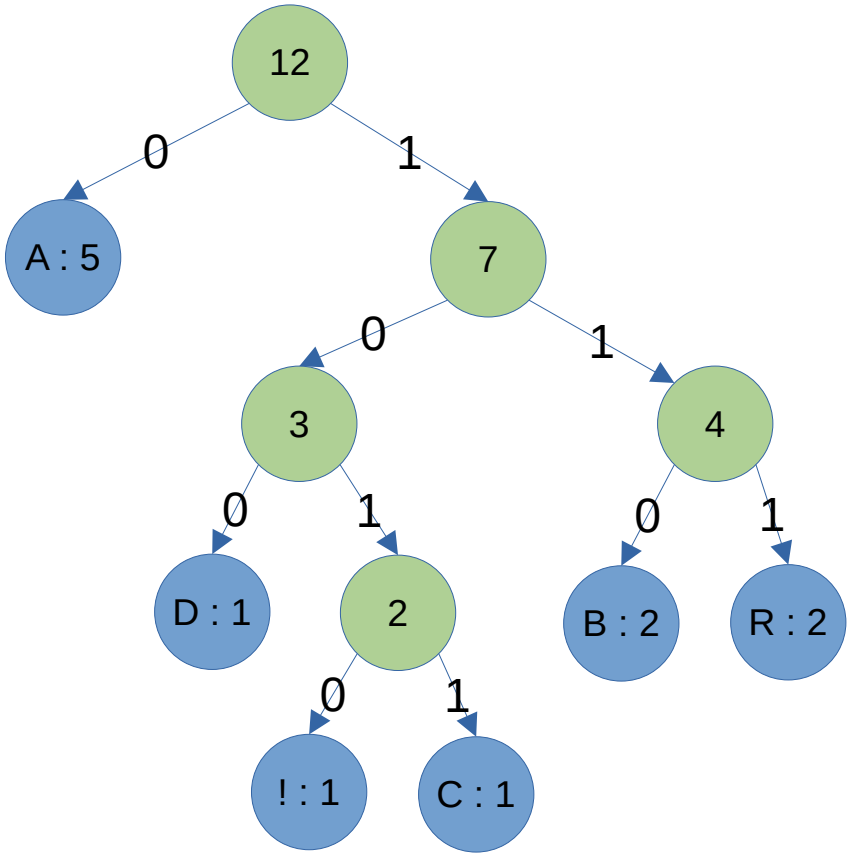


Exemplo Huffman



Exemplo Huffman

- Obtendo a codificação:
  - ! = 1010
  - A = 0
  - B = 110
  - C = 1011
  - D = 100
  - R = 111



Exemplo Huffman

- Codificando a entrada:
  - ! = 1010
  - A = 0
  - B = 110
  - C = 1011
  - D = 100
  - R = 111

A B R A C A D A B R A !

0110111010110100011011101010

Exemplo Huffman

- Expandindo:
  - lê um bit por vez e desce um nível na árvore
  - quando chega a uma folha, emite o caractere correspondente

0110111010110100110111010 10
A B R A C A D A B R A !