Compactação de Dados

Fonte de consulta: Szwarcfiter, J.; Markezon, L. Estruturas de Dados e seus Algoritmos, 3a. ed. LTC. Seção 12.5 em diante.

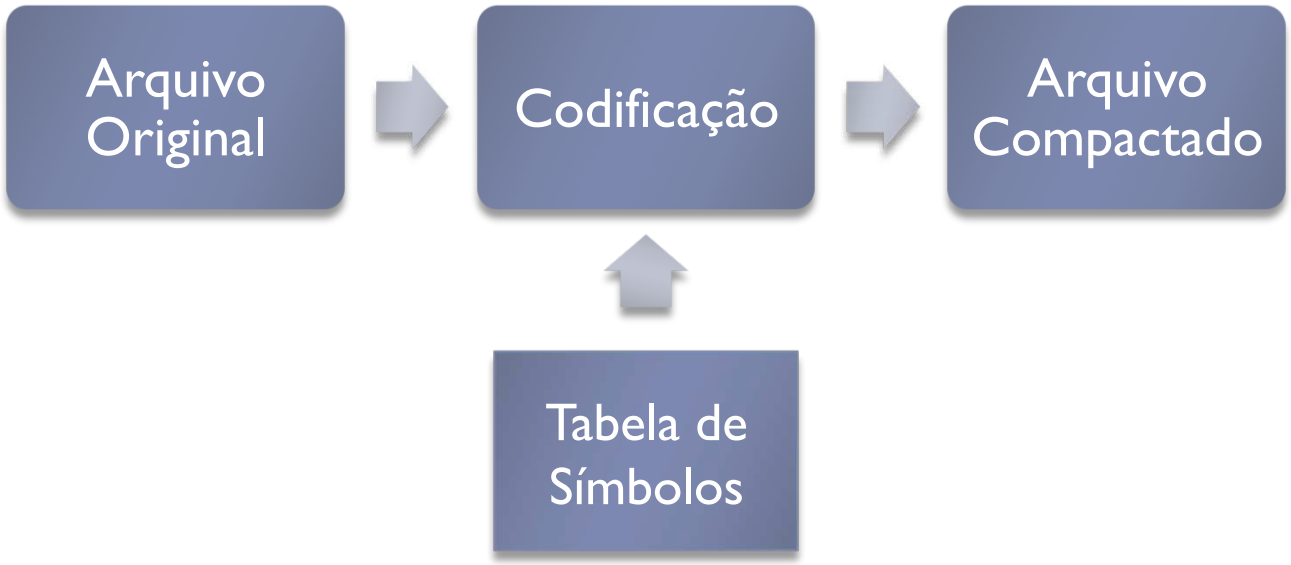
Compactação de Dados

- Armazenar arquivos grandes (backup)
  - Compactação permite armazenamento ocupando mesmo espaço
- Transmitir arquivos grandes por uma rede
  - Compactação permite transmissão mais eficiente

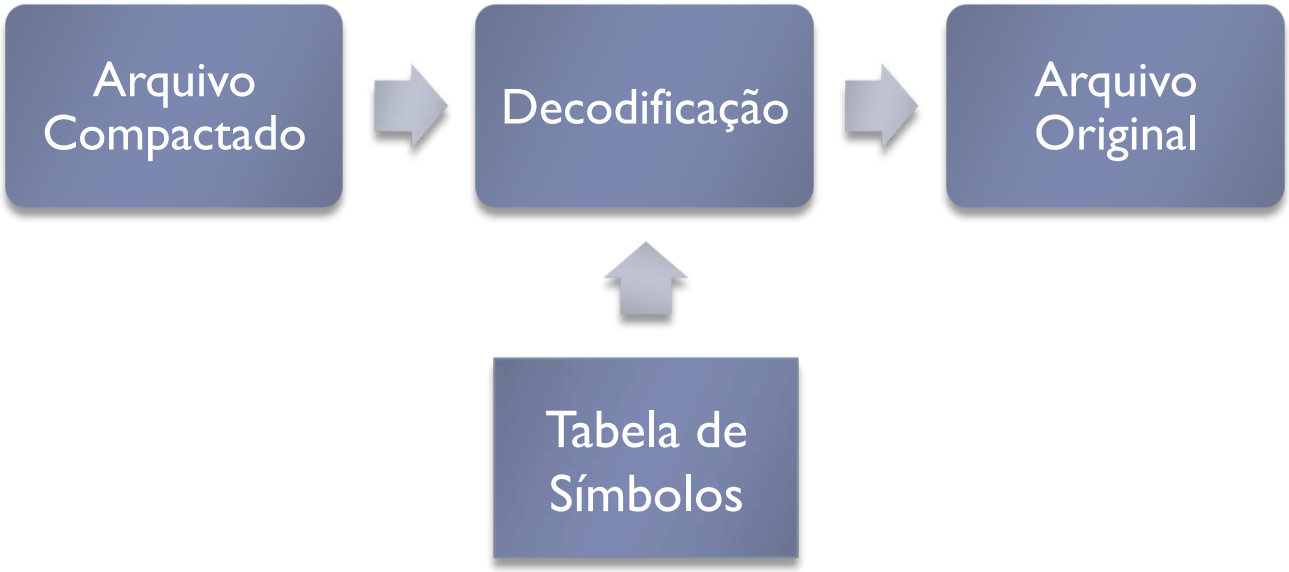
Compactação

- Dados do arquivo são substituídos por símbolos

Compactação = Codificação



Descompactação = Decodificação



Alternativas

- Algoritmo de Frequência de Caracteres
- Algoritmo de Huffman

Alternativas

- **Algoritmo de Frequência de Caracteres**
- Algoritmo de Huffman

Algoritmo de Frequência de Caracteres

- Pode ser usado em arquivos texto

- Exemplo: sequência de DNA
  - GGCTAACAAACAAGAAACATAAACAGAATAGGCGCCCT

Compactação

- Determinar a quantidade de símbolos idênticos consecutivos existentes no texto
- Cada uma das sequências máximas de símbolos idênticos é substituída por um número indicando da frequência do símbolo em questão, seguido do símbolo

Exemplo

- Cadeira Original:
  - GGCTAACAAACAAGAAACATAAACAGAATAGGCGCCCT
- Cadeia Compactada:
  - 2GICIT2AIC3AIC2AIG3AICIAIT3AICIAIG2AITIA2GICIG3CIT

Exemplo

- Cadeira Original:
  - GGCTAACAAACAAGAAACATAAACAGAATAGGCGCCCT
- Cadeia Compactada:
  - 2G1C1T2A1C3A1C2A1G3A1C1A1T3A1C1A1G2A1T1A2G1C1G3C1T

Ineficiente! Texto compactado ficou maior que o original

Otimização

- Assumir que ausência de frequência significa ocorrência = 1

- Exemplo:
- Cadeira Original:
  - GGCTAACAAACAAGAAACATAAACAGAATAGGCGCCCT
- Cadeia Compactada:
  - 2GCT2AC3AC2AG3ACAT3ACAG2ATA2GCG3CT

Arquivos que contêm números

- Textos que contêm números como símbolos podem causar problema, uma vez que o número pode ser confundido com frequência
- Solução: usar um caractere de escape, como @

- Exemplo:
- Cadeia Original:
  - AAA33333BA6666888DDDDDDD99999999999AABBB
- Cadeia Compactada:
  - 3A5@3BA4@63@87D11@92A3B

Alternativas

- Algoritmo de Frequência de Caracteres
- **Algoritmo de Huffman**

Algoritmo de Huffman

- Objetivo: produzir codificação ótima dentro de certos critérios

Texto e Alfabeto

- Texto é composto por um conjunto de símbolos $S = \{s_1, ..., s_n\}$, $n > 1$
- Para cada símbolo, sabe-se a frequência em que ele aparece no texto
  - Frequência total, incluindo ocorrências não consecutivas

- Exemplo:
  - Texto: AAA33333BA6666888DDDDDDD99999999999AABBB
  - S = {A, 3, B, 6, 8, D, 9}
  - Frequências: f(A) = 6; f(3) = 5; f(B) = 4; f(6) = 4; f(8) = 3; f(D) = 7; f(9) = 11

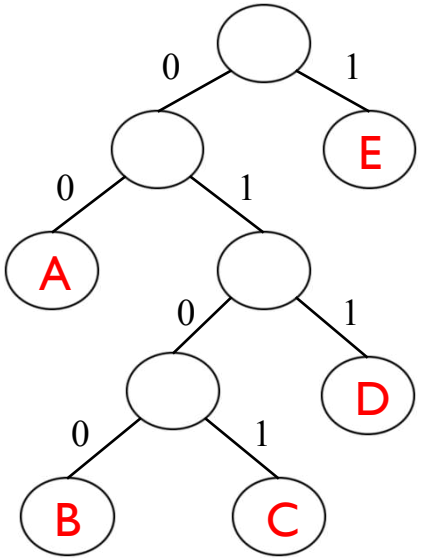
Códigos

- Nenhum código pode ser prefixo de outro
- Uso de árvore binária de prefixos
- Cada símbolo de S está em uma folha da árvore
- Códigos atribuídos a cada símbolo de S são sequências binárias

Exemplo

- S = {A, B, C, D, E}

Símbolo	Código
A	00
B	0100
C	0101
D	011
E	1



Codificação

- Uso da tabela de símbolos

Símbolo	Código
A	00
B	0100
C	0101
D	011
E	1

- Texto:
- DCCACADEACCCCCB CEBBBD

Codificação

- Uso da tabela de símbolos

Símbolo	Código
A	00
B	0100
C	0101
D	011
E	1

- Texto:
- DCCACADEACCCCCB CEBBBD

- Texto Codificado:
- 011010101010001010001 1100010101010101010 10101000101101000100 00011

Codificação

- Uso da tabela de símbolos

Símbolo	Código
A	00
B	0100
C	0101
D	011
E	1

- Texto:
- DCCACADEACCCCCB CEBBBD (21 bytes)

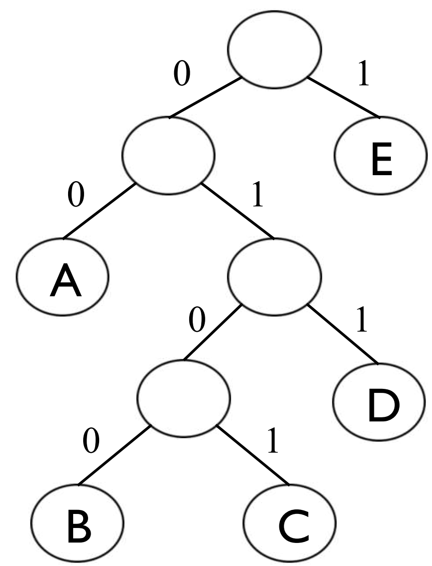
- Texto Codificado:
- 011010101010001010001 110001010101010101010 101010001011010001000 100011 (8 bytes + 5 bits)

Decodificação

- Uso da Árvore Binária de Prefixos
  - Percorrer o texto da esquerda para a direita, ao mesmo tempo em que a árvore é percorrida
  - Ao atingir uma folha, substituir a sequência pelo símbolo identificado, e voltar para a raiz da árvore

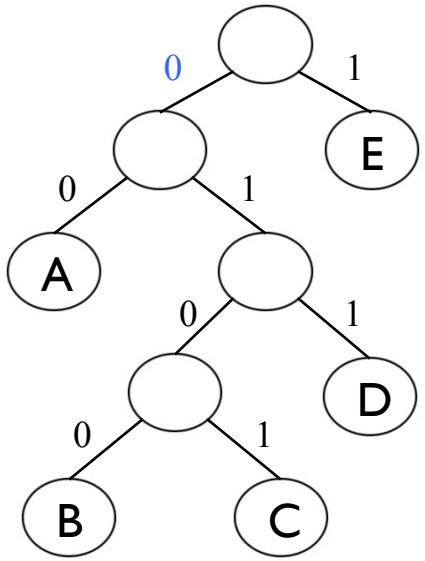
Exemplo de Decodificação

- 011010101010001010001
  110001010101010101010
  101010001011010001000
  100011



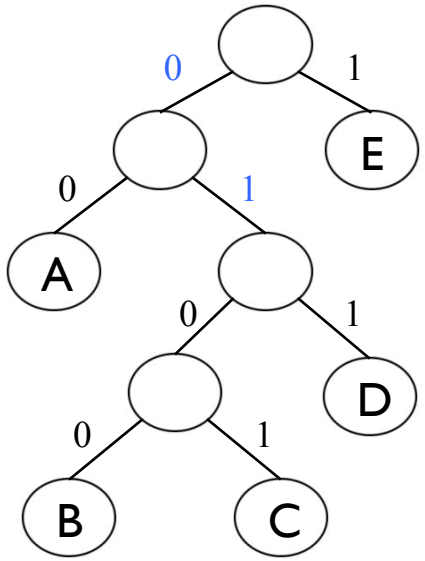
Exemplo de Decodificação

- **0**110101010001010001
  11000101010101010101010
  10101000101101000100
  10011



Exemplo de Decodificação

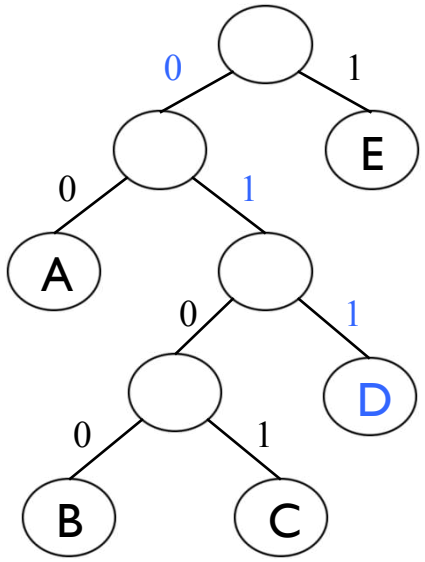
- **01**101010100010100011000101010101010101010100010110100010001100011



Exemplo de Decodificação

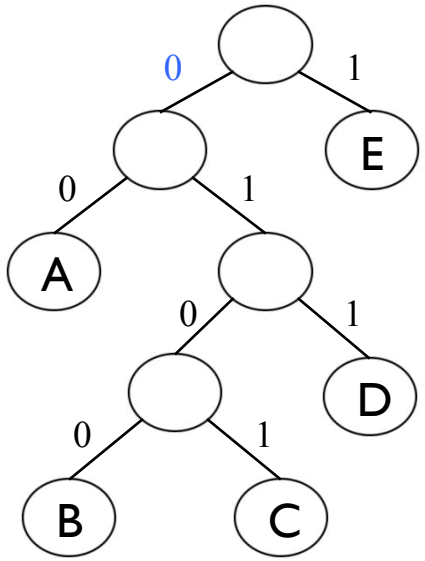
- **0 1 1** 0 1 0 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1
  1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
  1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0
  1 0 0 0 1 1

- Texto Decodificado:
- D



Exemplo de Decodificação

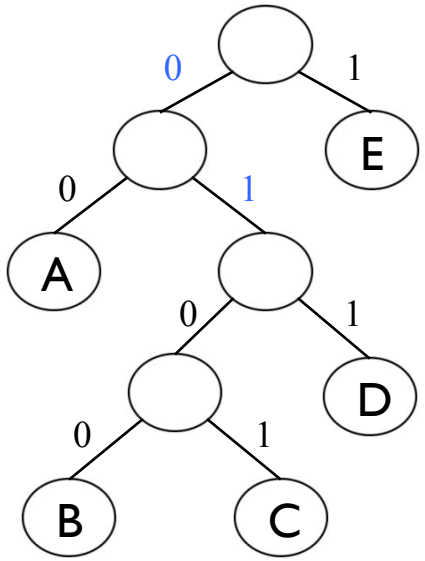
- 0110101010100010100011100010101010101010101010101000101101000100000111
- Texto Decodificado:
- D



Exemplo de Decodificação

- 0110101010100010100011100010101010101010101010100010110100010000100011

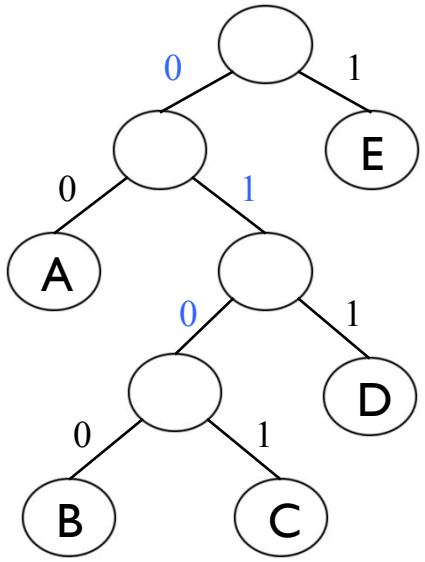
- Texto Decodificado:
- D



Exemplo de Decodificação

- 0 1 1 **0 1 0** 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1
  1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
  1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0
  1 0 0 0 1 1

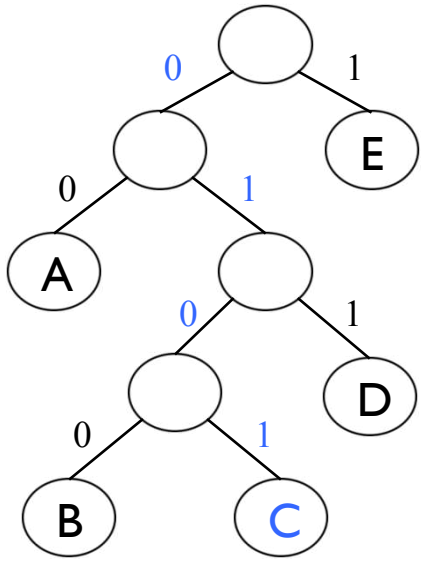
- Texto Decodificado:
- D



Exemplo de Decodificação

- 0 1 1 **0 1 0 1** 0 1 0 1 0 0 0 1 0 1 0 0 0 1
  1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
  1 0 1 0 1 0 0 0 1 0 1 1 0 1 0 0 0 1 0 0 0
  1 0 0 0 1 1

- Texto Decodificado:
- DC



Objetivo do Algoritmo de Huffman

- Minimizar o comprimento do texto codificado
- Ideia: atribuir códigos menores a símbolos mais frequentes

Checando o tamanho do texto codificado – 3 bytes menor que o exemplo anterior

- Uso da tabela de símbolos

Símbolo	Frequência	Código
A	3	101
B	4	111
C	9	0
D	3	110
E	2	100

- Texto:
- DCCACADEACCCCCB CEBBBD (21 bytes)

- Texto Codificado:
- 100001010101110100101 00000111010011111111 110 (5 bytes + 5 bits)

Geração dos códigos:
Árvore de Huffman

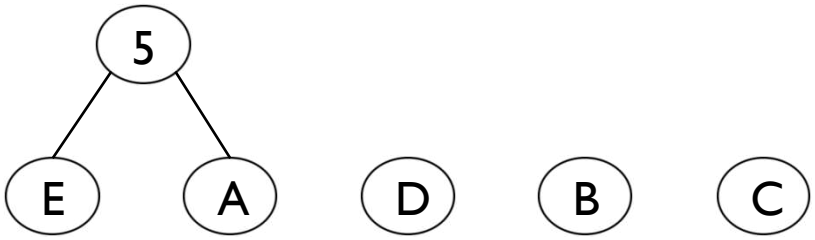
- Criar um nó da árvore para cada símbolo de S. Cada nó será considerado uma subárvore.
- Unir as duas subárvores com menor frequência, formando uma nova subárvore
  - A raiz da nova subárvore conterá a soma das frequências das duas subárvores unidas
- Continuar unindo subárvores até que reste apenas uma
- Rotular arestas com ZEROS e UNS (zeros à esquerda e uns à direita)

Exemplo

Símbolo	Frequência
A	3
B	4
C	9
D	3
E	2

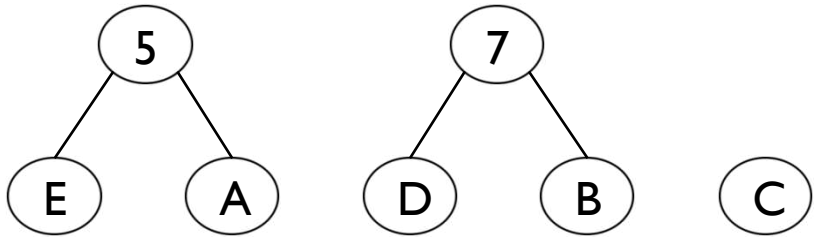
Exemplo

Símbolo	Frequência
A	3
B	4
C	9
D	3
E	2



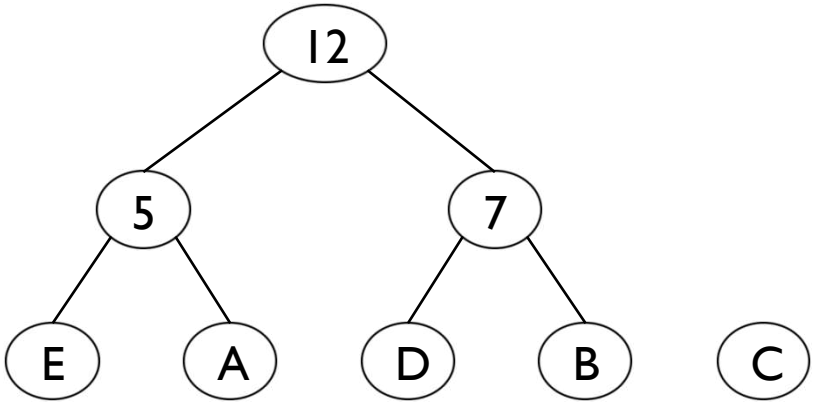
Exemplo

Símbolo	Frequência
A	3
B	4
C	9
D	3
E	2



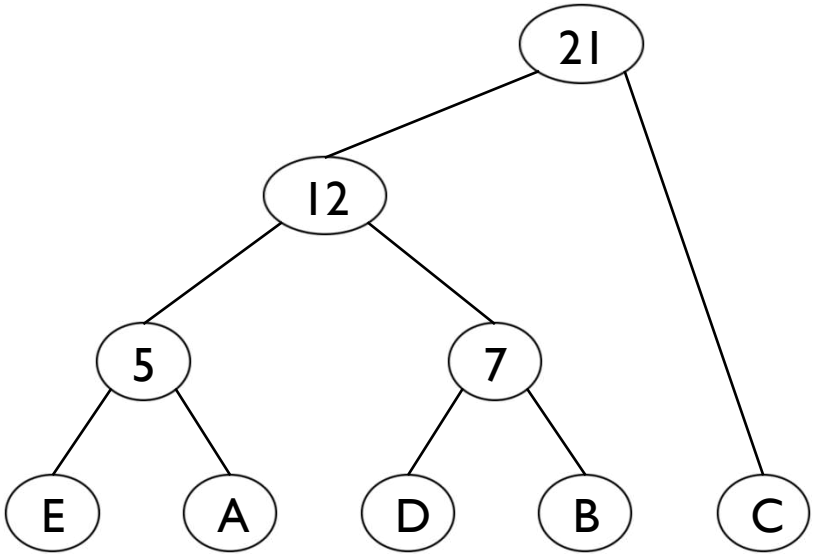
Exemplo

Símbolo	Frequência
A	3
B	4
C	9
D	3
E	2



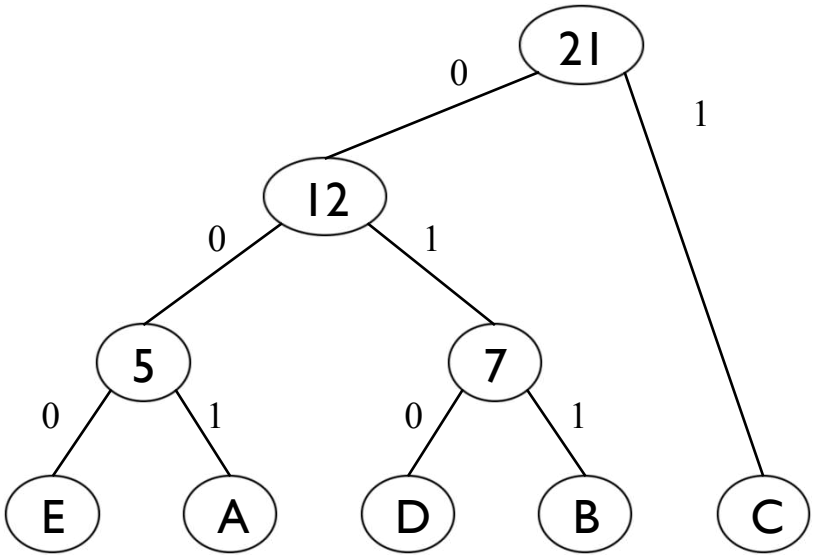
Exemplo

Símbolo	Frequência
A	3
B	4
C	9
D	3
E	2



Exemplo

Símbolo	Frequência	Código
A	3	001
B	4	011
C	9	1
D	3	010
E	2	000



Notem que essa árvore é um espelho da árvore do slide 32 – ambas são mínimas

Algoritmo de Huffman

- Uso de listas de prioridade

- Construir uma lista de prioridade com as frequências de cada subárvore
  - ATENÇÃO: prioridades são invertidas
  - Nó com menor frequência é considerado como tendo maior prioridade

Construção da Árvore de Huffman

```
/* T' e T" são subárvores
     ⊕ significa fusão de árvores
     minimo(T, F) retira o nó T de menor
     prioridade da lista de prioridades F
     inserir(T, f, F) insere o nó T na lista de
     prioridades F, com priorodade f
     n é o número de símbolos no alfabeto S
*/
procedimento huffman(F, n)
   para i:= 1 até n-1
      minimo(T', F)
      minimo(T", F)
      T := T' ⊕ T"
      f(T) := f(T') + f(T")
      inserir(T, f, F)
```


Fonte: Algoritmo 12.4, página 289

Exercício 1

- Escrever o texto compactado correspondente ao texto abaixo usando:

(a) Algoritmo de Frequência de Caracteres (omitindo o número 1 para ocorrências de um único caractere)

(b) Algoritmo de Huffman

BBBCCCCDEFFFBAAAAAAAAAACDDFFFFFFFCF

Exercício 2

- Desenhar a árvore de Huffman para o seguinte conjunto de símbolos e frequências

Símbolo	Frequência
A	1
B	6
C	2
D	1
E	1
F	9
G	2
H	3

Exercício 3

- Implementar um algoritmo de codificação de textos que recebe como entrada um String contendo o texto a ser codificado, e uma matriz que representa a tabela de codificação. O algoritmo deve produzir como resultado o texto codificado.