Algoritmos Gulosos: Códigos de Huffman

Marcus Vinicius Martins Melo

Departamento de Computação
Universidade Federal do Ceará

5 de agosto de 2021

UNIVERSIDADE FEDERAL DO CEARÁ

Conceitos

- O algoritmo de Huffman tem como objetivo principal a compressão de dados.
- O algoritmo guloso de Huffman usa uma tabela que diz com qual frequência cada caractere ocorre para construir um caminho ótimo para representar cada caractere como uma string binária.
- Características importantes
  - Compressões pode variar de 20% a 90%.
  - É um algoritmo guloso que cria soluções dependendo das entradas de dados a serem manipuladas.
  - Utiliza código de comprimento variável para representar dados frequentemente acessados.

- Suponha que nós temos um arquivo com 100.000 caracteres que nós desejamos armazenar compactamente.
- Observamos que os caracteres no arquivo ocorrem com a seguinte frequências

Caractere	a	b	c	d	e	f
Frequência (em milhares)	45	13	12	16	9	5

- Utilizando código de caracteres binários, podemos usar os códigos de tamanho fixo ou variável.
- Comprimento fixo (3 bits por caractere): $(45\times3 + 13\times3 + 12\times3 + 16\times3 + 9\times3 + 5\times3)\times1.000=300.000$ bits.

Caractere	a	b	c	d	e	f
Frequência (em milhares)	45	13	12	16	9	5
Palavras em comprimento Fixo	000	001	010	011	100	101

- Comprimento variável: A cadeia de 1 bit 0 representa a, e a cadeia 1100 representa f.
- $(45\times1 + 13\times3 + 12\times3 + 16\times3 + 9\times4 + 5\times4)\times1.000=224.000$ bits.

	a	b	c	d	e	f
Frequency (in thousands)	45	13	12	16	9	5
Fixed-length codeword	000	001	010	011	100	101
Variable-length codeword	0	101	100	111	1101	1100

- Considerando que não deve haver palavras de código que são prefixos de outras palavras de código.
- Como nenhuma palavra é prefixo de outra, a palavra de código que inicia um arquivo codificado não é ambígua.
- Códigos de prefixo são desejáveis porque simplificam a decodificação.

Algoritmo de Huffman

- O Algoritmo de Huffman utiliza uma estrutura de arvore para definição/escolha dos prefixos de comprimento variável.
- Arvore binaria completa, com n nos externos e n-1 nos internos.
- Os nós externos(folhas) são rotulados com as frequências.

UNIVERSIDADE FEDERAL DO CEARÁ

- Comece com *n* arvores disjuntas, cada uma com um único nó, com o simbolo e sua frequência.

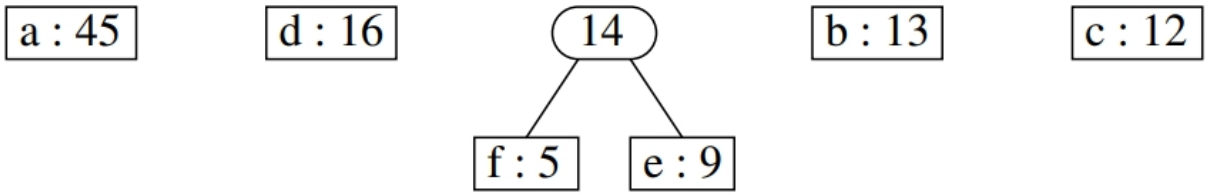
a : 45	d : 16	b : 13	c : 12	e : 9	f : 5

Algoritmo de Huffman

Etapas

- Comece com *n* arvores disjuntas, cada uma com um único nó, com o simbolo e sua frequência.

- A cada iteração, escolha as duas arvores de frequência menor e junte-as, com frequência somada (elemento de menor valor na esquerda).

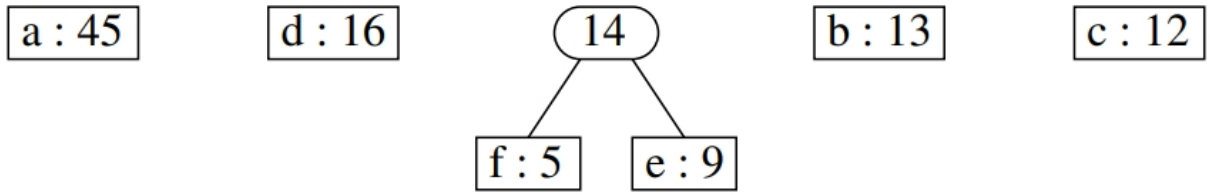


Algoritmo de Huffman

Etapas

- Comece com *n* arvores disjuntas, cada uma com um único nó, com o simbolo e sua frequência.

- A cada iteração, escolha as duas arvores de frequência menor e junte-as, com frequência somada (elemento de menor valor na esquerda).



- Procedimento continua ate restar uma única arvore.

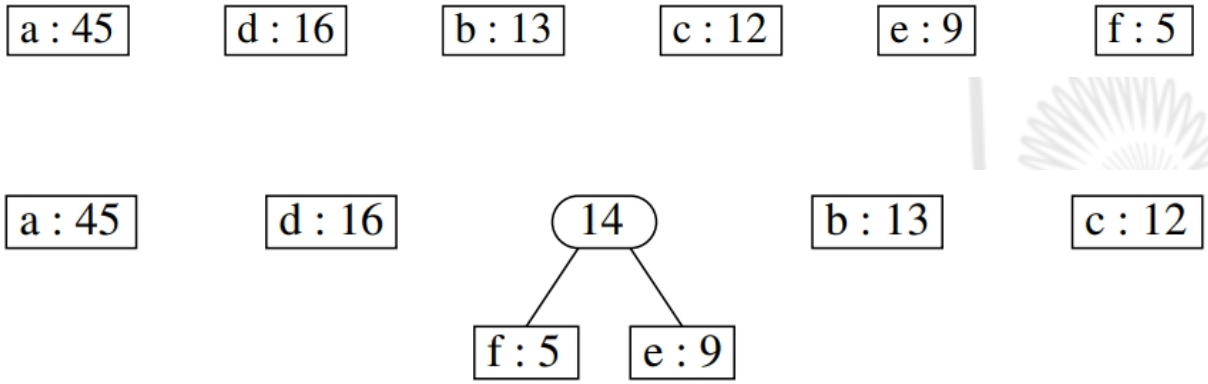
Algoritmo de Huffman

Exemplo

a : 45	d : 16	b : 13	c : 12	e : 9	f : 5

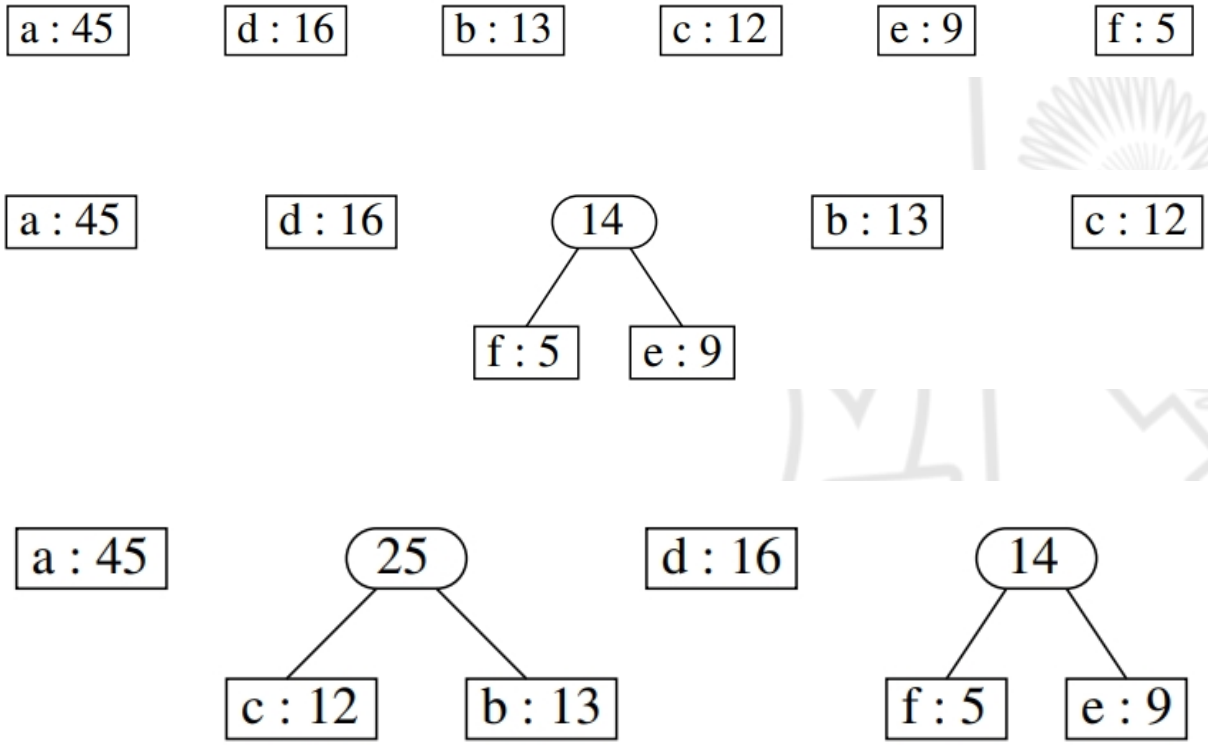
Algoritmo de Huffman

Exemplo



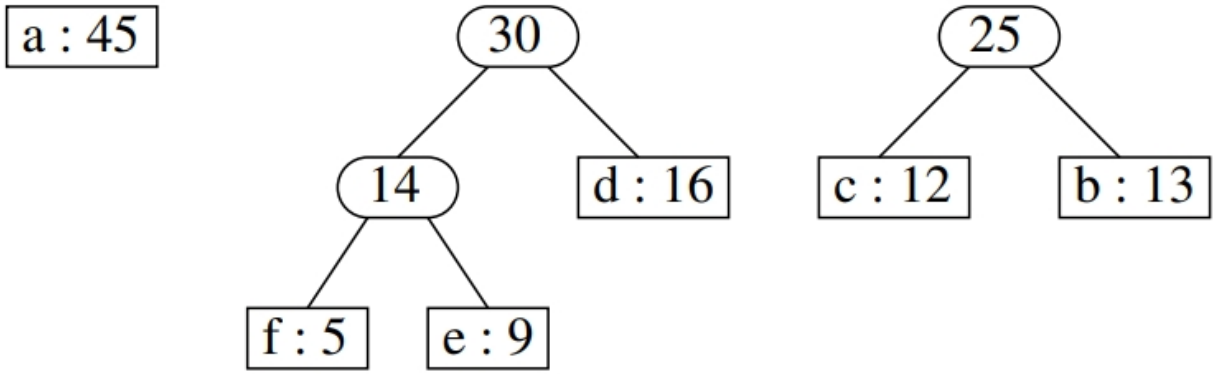
Algoritmo de Huffman

Exemplo



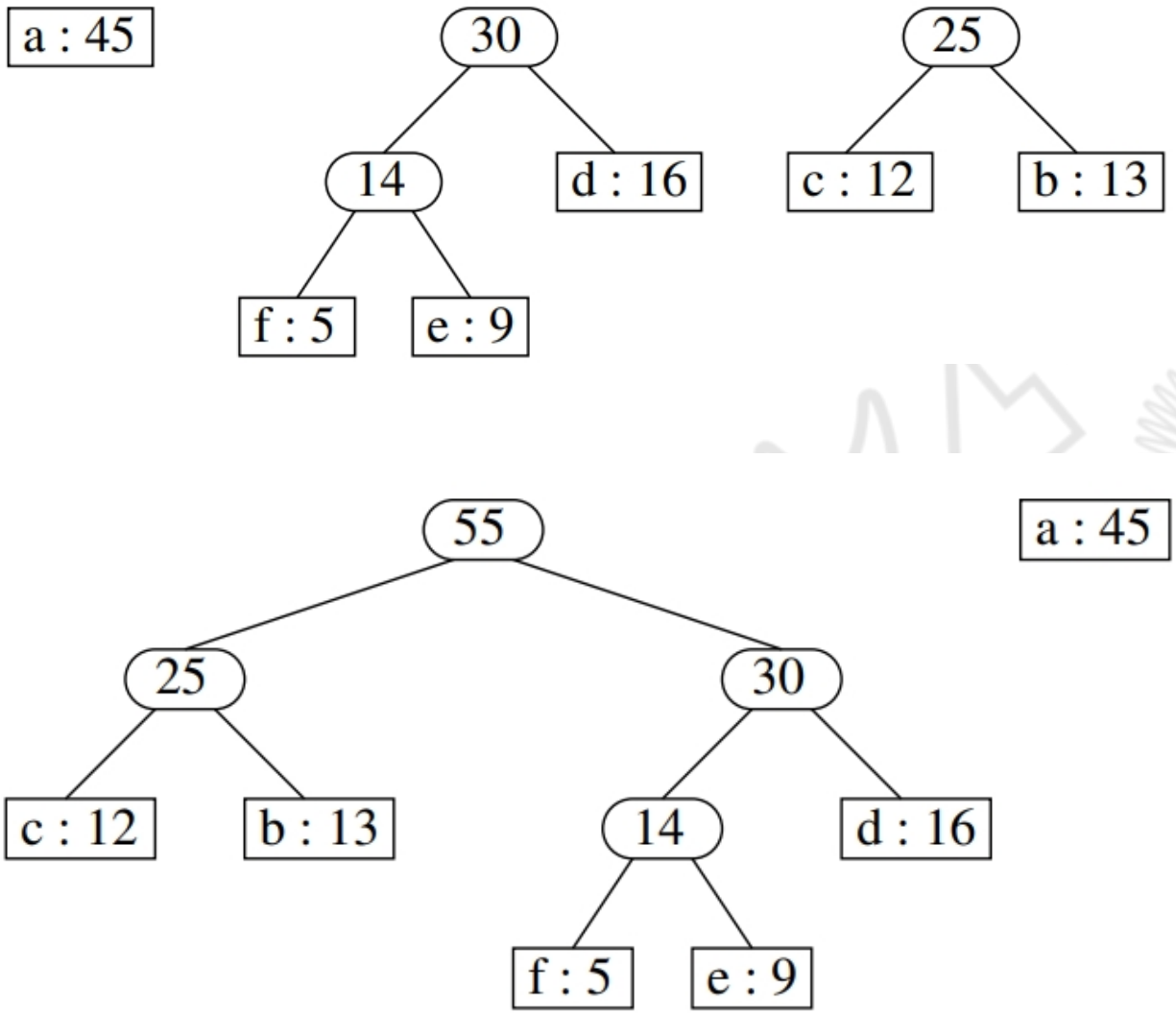
Algoritmo de Huffman

Exemplo



Algoritmo de Huffman

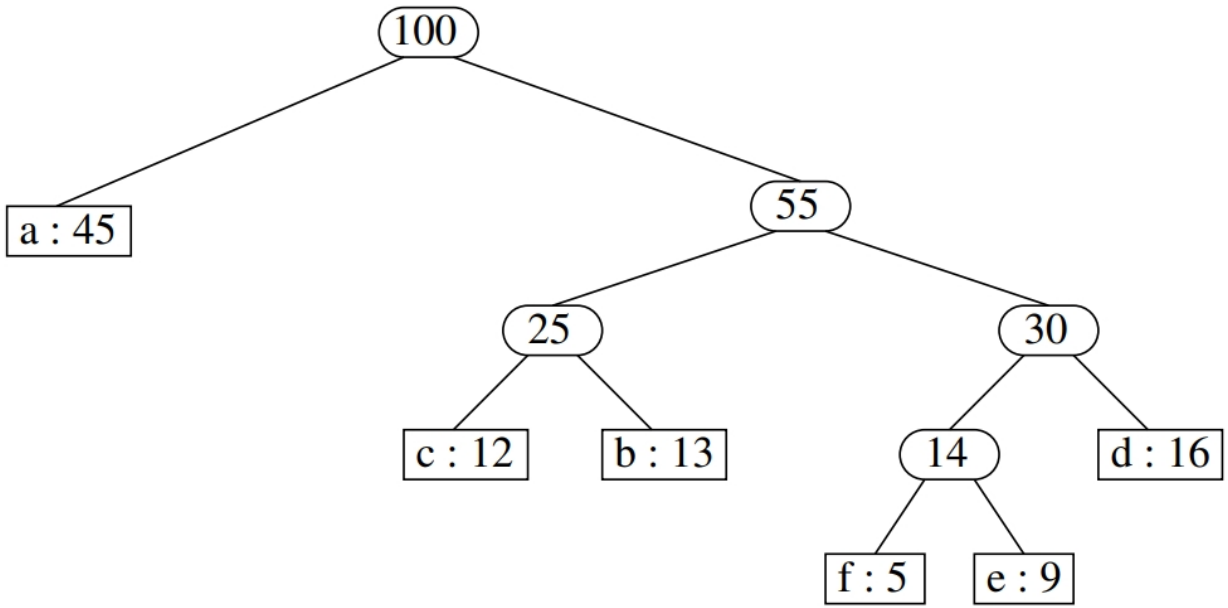
Exemplo



UNIVERSIDADE FEDERAL DO CEARÁ

Algoritmo de Huffman

Exemplo



UNIVERSIDADE FEDERAL DO CEARÁ

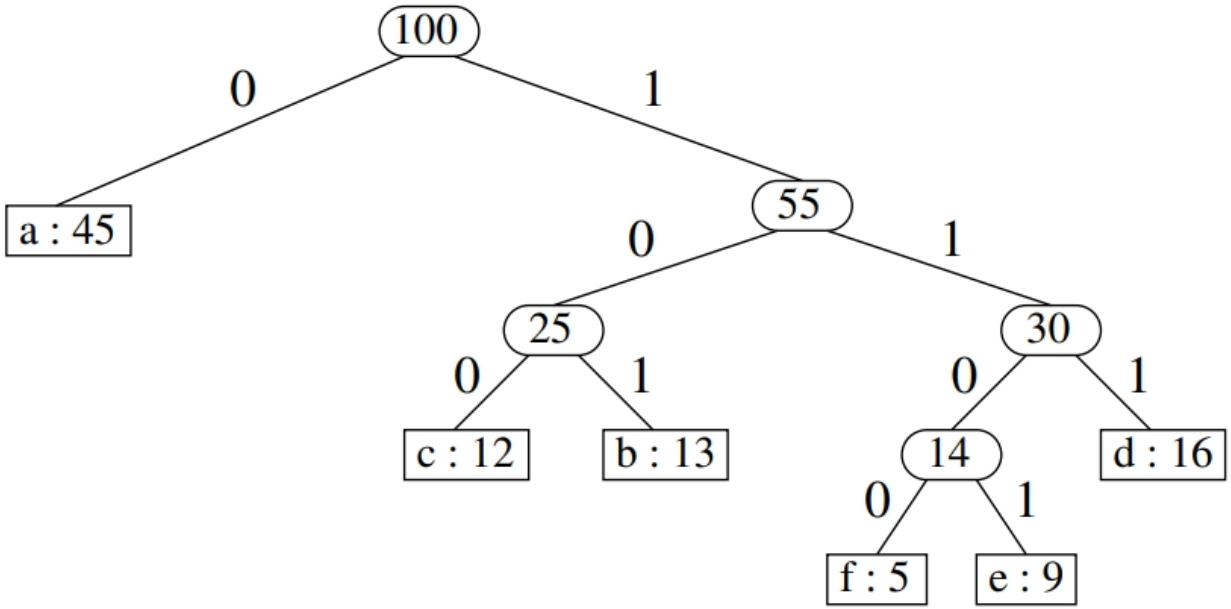
Algoritmo de Huffman

Exemplo

- Como obter os códigos a partir da arvore?
- O processo de decodificação do código de Huffman para cada caractere distinto é similar a decodificação de uma mensagem.
- Dado um nó folha (caractere), parte-se da raiz até alcançá-lo.
  - Se desceu pelo filho da esquerda, acrescenta 0 ao código
  - Se desceu pelo filho da direita, acrescenta 1

Algoritmo de Huffman

Etapa



	a	b	c	d	e	f
Frequência (em milhares)	45	13	12	16	9	5
Codificação	0	101	100	111	1101	1100

UNIVERSIDADE FEDERAL DO CEARÁ

- Código de comprimento fixo:
  - $(45\times3 + 13\times3 + 12\times3 + 16\times3 + 9\times3 + 5\times3)\times1.000$=300.000 bits.
- Código de comprimento variável:
  - $(45\times1 + 13\times3 + 12\times3 + 16\times3 + 9\times4 + 5\times4)\times1.000$=224.000 bits.
- Economia de aproximadamente 25%.

Caractere	a	b	c	d	e	f
Frequência (em milhares)	45	13	12	16	9	5
Palavras em comprimento Fixo	000	001	010	011	100	101
Palavras em comprimento Variável	0	101	100	111	1101	1100

Algoritmo de Huffman

Pseudocódigo

```
HUFFMAN (A, f, n)
 1  Q ← BUILD-MIN-HEAP(A, f, n)
 2  para i ← 1 até n − 1 faça
 3      x ← EXTRACT-MIN(Q)
 4      y ← EXTRACT-MIN(Q)
 5      z ← NOVA-CEL()
 6      esq[z] ← x
 7      esq[z] ← y
 8      f[z] ← f[x] + f[y]
 9      INSEREHEAP(Q, z, f[z])
10  devolva EXTRACT-MIN(Q)
```


UNIVERSIDADE FEDERAL DO CEARÁ

Algoritmo de Huffman

Pseudocódigo

```
HUFFMAN (A, f, n)
 1  Q ← BUILD-MIN-HEAP(A, f, n)
 2  para i ← 1 até n − 1 faça
 3      x ← EXTRACT-MIN(Q)
 4      y ← EXTRACT-MIN(Q)
 5      z ← NOVA-CEL()
 6      esq[z] ← x
 7      esq[z] ← y
 8      f[z] ← f[x] + f[y]
 9      INSEREHEAP(Q, z, f[z])
10  devolva EXTRACT-MIN(Q)
```

$O(nlogn)$: tempo de execução utilizando com conjunto de n caracteres implementado com heap de mínimo binário.

Obrigado(a) pela Atenção!

UNIVERSIDADE FEDERAL DO CEARÁ