

Fontes

Algoritmos

Pedro Hokama

- [crls] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest, Ronald L. Rivest e Clifford Stein.

- [timr] Algorithms Illuminated Series, Tim Roughgarden

Apresentação Baseada:

- Stanford Algorithms

<https://www.youtube.com/playlist?list=PLXFMmlk03Dt7Q0xr1PIAriY5623cKiH7V>

<https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZ17A5HEK6>

- Conjunto de Slides dos Professores Cid C. de Souza, Cândida N. da Silva, Orlando Lee, Pedro J. de Rezende

- Conjunto de Slides do Professores Cid C. de Souza para a disciplina MO420

Qualquer erro é de minha responsabilidade.

1 / 12

2 / 12

Representação de Caracteres

- Tradicionalmente caracteres são representados em binários através de uma codificação de tamanho fixo, ou seja, todos os caracteres têm o mesmo número de bits, tradicionalmente 8 bits.
- Para simplificar, considere um alfabeto bem pequeno, de 6 letras { a, b, c, d, e, f }.
- Para representar esse alfabeto poderíamos usar uma representação de 3 bits.

| a | b | c | d | e | f |
|-----|-----|-----|-----|-----|-----|
| 000 | 001 | 010 | 011 | 100 | 101 |

- Para representar a palavra “babaca” precisaríamos de 18 bits:

001000001000010000

- Para representar um texto de 100.000 caracteres, precisaríamos de 300.000 bits.

3 / 12

Compressão de Dados

- Suponha agora que queremos comprimir o texto, de forma a ocupar uma quantidade de bits menor.
- Uma ideia é utilizar uma codificação de tamanho variável, ao invés da de tamanho fixo.
- Considere a seguinte codificação (Spoiler: Não vai funcionar)

| a | b | c | d | e | f |
|---|---|----|----|----|----|
| 0 | 1 | 00 | 01 | 10 | 11 |

- Para representar a palavra “babaca” precisaríamos agora de apenas de 7 bits:

1010000

- Para representar um texto de 100.000 caracteres, precisaríamos de menos de 200.000 bits.

4 / 12

Porém não funcionaria!

| | | | | | |
|---|---|----|----|----|----|
| a | b | c | d | e | f |
| 0 | 1 | 00 | 01 | 10 | 11 |

- Não é possível decodificar uma palavra. Por exemplo:

1010000

- Poderia ser “babaca”. Mas também poderia ser “eeca” ou “bdaaaa”
- O Problema é que o código de um caractere é prefixo de outro caractere.

5 / 12

Solução

- Utilizando essa codificação em um texto de 100.000 caracteres, em que cada um aparece com a seguinte frequência:

| | | | | | |
|-----|-----|-----|-----|------|------|
| a | b | c | d | e | f |
| 0 | 101 | 100 | 111 | 1101 | 1100 |
| 45% | 13% | 12% | 16% | 9% | 5% |

- Seriam necessários:

$$(45 \times 1 + 13 \times 3 + 12 \times 3 + 16 \times 3 + 9 \times 4 + 5 \times 4) \times 1000 = 224.000$$

- Compressões de até 90%
- Como escolher uma boa codificação?

7 / 12

Solução

- Codificação de tamanho variável livre de prefixo.

| | | | | | |
|---|-----|-----|-----|------|------|
| a | b | c | d | e | f |
| 0 | 101 | 100 | 111 | 1101 | 1100 |

- Dessa forma uma codificação não será ambígua.

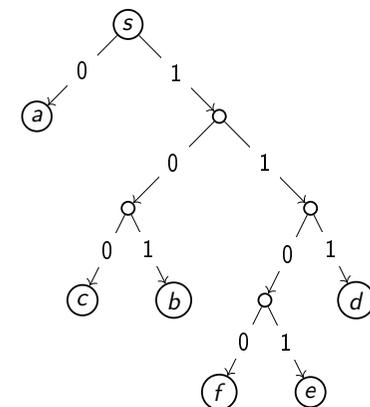
101010101000

- Com essa codificação a palavra “babaca” precisa de 12 caracteres.

6 / 12

Código de Huffman

- Criaremos uma árvore binária que representa uma codificação.
- A cada ramificação para a esquerda temos um bit 0 e cada ramificação a direita temos um bit 1.
- As folhas representam os caracteres.



8 / 12

Código de Huffman

- Construindo a árvore.

| a | b | c | d | e | f |
|-----|-----|-----|-----|----|----|
| 45% | 13% | 12% | 16% | 9% | 5% |

Complexidade

9 / 12

Código de Huffman

Algoritmo 1: Algoritmo Aleatorizado de Contração

Entrada: Um alfabeto C e suas frequências

Saída: Um código de tamanho variável livre de frequência

```
1 para cada  $c$  em  $C$  faça
2    $Q.inserir(c)$ ;
3 enquanto  $|Q| > 1$  faça
4   Criar novo Nó  $V$ ;
5    $V.esq = ExtraiMin(Q)$ ;
6    $V.dir = ExtraiMin(Q)$ ;
7    $V.freq = V.esq.freq + V.dir.freq$ ;
8    $Q.inserir(V)$ ;
9 devolva Árvore construída;
```

10 / 12

Corretude

11 / 12

12 / 12