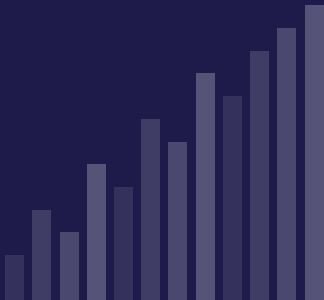

AULA 34 · BLOCO 4: GRAFOS

Algoritmos gulosos: seleção de atividades e Huffman

Seções 16.1 a 16.3

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Reconhecer a propriedade da escolha gulosa e a subestrutura ótima
- ▶ Resolver a seleção de atividades escolhendo sempre a que termina primeiro
- ▶ Construir códigos de Huffman com fila de prioridade
- ▶ Medir o comprimento médio de Huffman contra a entropia de Shannon



Teoria

Seleção de atividades, gulosos contra programação dinâmica, códigos de prefixo



Prática em C

Huffman em C com as frequências da Figura 16.3



Experimento

Huffman, entropia e código fixo para alfabetos de 2 a 4.096 símbolos



Conexões

Compressão, redes, agendamento e aprendizado de máquina

Seleção de atividades

Atividades $a_i = [s_i, f_i)$ disputam uma sala. Escolher o maior conjunto sem sobreposição.

- ▶ **Escolha gulosa:** a atividade que **termina primeiro** deixa a sala livre o mais cedo possível.
- ▶ **Teorema 16.1:** ela pertence a alguma solução ótima.
- ▶ Depois, o problema restante tem a mesma forma.
- ▶ Com as atividades ordenadas por término: $\Theta(n)$.

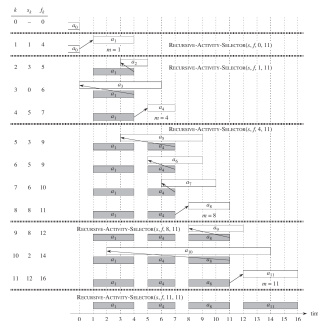


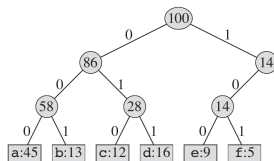
Figura 16.1 de Cormen et al. (2009) —

Recursive-Activity-Selector: em cada passo, a próxima atividade compatível que termina primeiro.

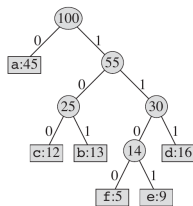
Códigos de prefixo

- ▶ **Código de prefixo:** nenhuma palavra é prefixo de outra; decodificar é só seguir a árvore.
- ▶ Custo de uma árvore T :

$$B(T) = \sum_c c.freq \cdot d_T(c).$$
- ▶ Símbolos frequentes devem ficar perto da raiz.
- ▶ Um código ótimo corresponde a uma árvore **cheia** (todo nó interno tem dois filhos).



(a)



(b)

Figura 16.4 de Cormen et al. (2009) — (a) código de tamanho fixo, 300 mil bits; (b) código ótimo, 224 mil bits.

Huffman

$\text{HUFFMAN}(C)$: coloque os símbolos numa fila de mínimo por frequência; $n - 1$ vezes, retire os dois menores x e y , crie z com $z.\text{freq} = x.\text{freq} + y.\text{freq}$ e filhos x , y , e reinsira z .

Custo $O(n \lg n)$ com heap (aula 17).

Lemas 16.2 e 16.3: a escolha gulosa é segura e o problema restante tem subestrutura ótima.

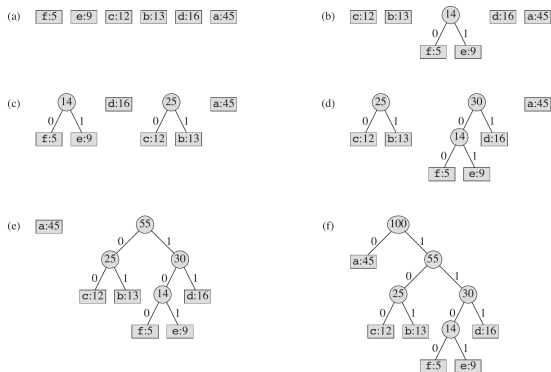


Figura 16.5 de Cormen et al. (2009) — construção do código de Huffman: juntar sempre os dois menos



Prática em C

- ▶ Frequências de a a f da Figura 16.3
- ▶ Fila de prioridade simples (vetor) para 6 símbolos
- ▶ Imprime o código de cada símbolo e o total: 224 mil bits

Códigos de Huffman em C (1/3)

```
1  #include <stdio.h>
2
3  /* Frequências (em milhares) de a..f, como na Figura 16.3. A fila de prioridade é
4     um vetor simples: com 6 símbolos, achar o mínimo por varredura basta. */
5  #define MAX 64
6  int freq[2 * MAX], esq[2 * MAX], dir[2 * MAX], vivo[2 * MAX], n_nos;
7  char codigo[MAX][MAX];
8
9  int extrai_min(void) {
10     int m = -1;
11     for (int i = 0; i < n_nos; i++)
12         if (vivo[i] && (m < 0 || freq[i] < freq[m])) m = i;
13     vivo[m] = 0;
14     return m;
15 }
```

Códigos de Huffman em C (2/3)

```
16
17 void gera(int x, char *prefixo, int prof, int n) { // percorre a árvore: 0 esq, 1 dir
18     if (x < n) { prefixo[prof] = '\\0'; sprintf(codigo[x], "%S", prefixo); return; }
19     prefixo[prof] = '0'; gera(esq[x], prefixo, prof + 1, n);
20     prefixo[prof] = '1'; gera(dir[x], prefixo, prof + 1, n);
21 }
22
23 int main(void) {
24     int n = 0;
25     while (n < MAX && scanf("%d", &freq[n]) == 1) { vivo[n] = 1; n++; }
26     if (n < 2) return 1;
27     n_nos = n;
28     for (int k = 1; k < n; k++) { // HUFFMAN: n - 1 junções
29         int x = extrai_min(), y = extrai_min();
```

Códigos de Huffman em C (3/3)

```
30     freq[n_nos] = freq[x] + freq[y];           // novo nó z com filhos x e y
31     esq[n_nos] = x;
32     dir[n_nos] = y;
33     vivo[n_nos++] = 1;
34 }
35 char prefixo[MAX];
36 gera(n_nos - 1, prefixo, 0, n);
37 long bits = 0;
38 for (int i = 0; i < n; i++) {
39     printf("%c:%s ", 'a' + i, codigo[i]);
40     for (int j = 0; codigo[i][j]; j++) bits += freq[i];
41 }
42 printf("\n%ld mil bits\n", bits);
43 return 0;
44 }
```



Experimento

- ▶ Alfabetos de 2 a 4.096 símbolos com frequências de Zipf
- ▶ Huffman com heap
- ▶ Verificação: $H \leq L < H + 1$ em todos os casos

Huffman contra o limite de Shannon

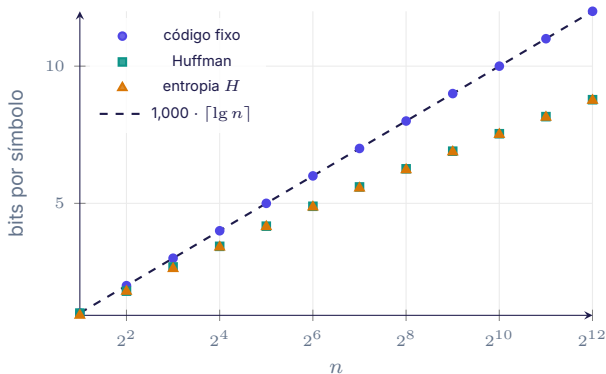
Hipótese. Para qualquer código de prefixo, o comprimento médio L é pelo menos a **entropia** $H = -\sum p_i \lg p_i$ (Shannon). O código de Huffman é ótimo e garante $H \leq L < H + 1$. O código de tamanho fixo usa $\lceil \lg n \rceil$ bits: Huffman nunca pode ser pior.

Método

1. Frequências proporcionais a $1/i$ (lei de Zipf)
2. $n = 2, 4, 8, \dots, 4.096$ símbolos
3. $L = \sum p_i \cdot$ profundidade da folha i
4. O programa aborta se $L \notin [H, H + 1)$

🧪 EXPERIMENTO

Bits por símbolo



n	medido	razão
2,000	1,000	1,000
8,000	2,682	0,894
32,0	4,168	0,834
128,0	5,598	0,800
512,0	6,901	0,767
4.096	8,777	0,731

razão = medido $\div$ $\lceil \lg n \rceil$

✓ sempre $\leq$ o limite

Huffman fica colado na entropia (diferença de 0,03 a 0,08 bit) e economiza 2 a 3 bits por símbolo em relação ao código fixo.

🧪 EXPERIMENTO

O que o experimento mostrou

Em todos os alfabetos, $H \leq L < H + 1$, e na prática $L - H < 0,1$ bit: Huffman fica praticamente no **limite teórico** da compressão sem perdas símbolo a símbolo. Um algoritmo guloso de cinco linhas atinge o ótimo, e a teoria da informação diz que ninguém pode fazer melhor.

➤ Reproduza: `gcc -O2 aula34_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Compressão
- ▶ Redes
- ▶ Agendamento
- ▶ Aprendizado de máquina

Onde isso aparece na computação

 Compressão

DEFLATE (ZIP, gzip, PNG), JPEG e MP3 usam códigos de Huffman na etapa final; formatos modernos usam codificação aritmética, que chega ainda mais perto de H .

 Redes

O HTTP/2 comprime cabeçalhos com HPACK, que usa uma tabela de Huffman fixa: cada requisição web economiza bytes.

 Agendamento

Escalonar tarefas por prazo (*earliest deadline first*) e alocar salas, frequências de rádio ou pistas de pouso são problemas gulosos parentes da seleção de atividades.

 Aprendizado de máquina

Árvores de decisão escolhem gulosamente o atributo de maior ganho de informação (entropia!); o *hierarchical softmax* usa árvores de Huffman.

Para casa

Exercícios

1. (16.1-1) Escreva a seleção de atividades por programação dinâmica e compare com a gulosa.
2. (16.1-3) Mostre que escolher a atividade mais curta (ou com menos conflitos) não é ótimo.
3. (16.2-2) Resolva a mochila 0-1 por programação dinâmica em $O(nW)$. Por que a gulosa falha?
4. (16.3-3) Qual o código de Huffman para frequências de Fibonacci 1, 1, 2, 3, 5, 8, 13, 21?
5. Comprima um livro em português (Domínio Público) com seu Huffman por caractere. Compare com gzip.

Leitura

Cormen et al., seções 16.1 a 16.3.

Código

josedihego.net/estruturas