



- ▶ Procure um identificador pelo seu nome e não pela categoria à qual ele pertence. Por exemplo, não tente encontrar *função printf()*; em vez disso, procure diretamente *printf()*.
- ▶ Número de página em negrito significa que a respectiva informação procurada está em nota de rodapé.

Símbolos

- ^ (acento circunflexo) disjunção exclusiva de bits, operador de 700
- | (barra vertical) disjunção de bits, operador de 700
- & (e comercial) conjunção de bits, operador de 700
- (final de prova) q.e.d. 162
- Σ (letra grega sigma maiúscula) alfabeto 452
- >> (maior maior) deslocamento de bits à direita, operador de 701
- << (menor menor) deslocamento de bits à esquerda, operador de 701
- [] (pisso esquerdo piso direito) função piso (matemática) 159
- [] (teto esquerdo teto direito) função teto (matemática) 159
- ~ (til), complemento, operador de 700

A

- "a+b", modo de abertura de arquivo 98, 115
- "a", modo de abertura de arquivo 98
- "a+", modo de abertura de arquivo 98, 115
- "a+t", modo de abertura de arquivo 98, 115

- "ab", modo de abertura de arquivo 98
- abertura de arquivo 96–100
- AbreArquivo(), função 96, 558, 658
- ABS, macro 256
- acerto de cache 74
- acesso a instrução, localidade de 81
- acesso aleatório a arquivo 106
- acesso direto a arquivo 106, 111–115
- acesso direto a memória 69
- acesso sequencial vs. acesso direto 64–65
- AcrescentaElementoIdx(), função 148, 152, 157
- AcrescentaFuncaoAFiltro(), função 407
- AcrescentaItemAFiltro(), função 407
- adaptador SCSI/SATA 69
- Adelson-Velskii e Landis 217
- Afunila(), função 244
- afunilamento de nó 234
 - alvo 234
 - ascendente 239
 - com configuração heterogênea 236–237
 - com configuração homogênea 235–236
 - descendente 239–241
 - exemplos de 237–239
 - filho da raiz 235
 - implementação de 244–246
 - zag 235
 - zag-zag 235–236
 - zag-zig 236–237
 - zig 235
 - zig-zag 236–237
 - zig-zig 235–236
- AFUNILAMENTODESCENDENTE, algoritmo 239

- agregado, método de análise amortizada de 271, 275–276, 278–279. *V. também* análise amortizada
- agrupamento
 primário 378
 secundário 379
- alarme falso 404
- alfabeto 452
- algoritmo. *Procure um algoritmo específico pelo nome dele*
 de dispersão 360
 de divisão e conquista 624. *V. também* **MERGESORT**, algoritmo; *V. também* **QUICKSORT**, algoritmo
 funcionalmente equivalente 80
 de inserção 143
 de leitura sequencial de arquivo 110
 de redução e conquista 777
 de remoção 143
- alocação dinâmica de memória 125, 296
- alteração de registro de arquivo 115
- altura
 de árvore AVL 230
 de árvore B 349–350
 de árvore binária 255
 de trie 487, 499
- AlturaArvoreBB2()**, função 255
- AlturaAVL()**, função 230
- AlturaB()**, função 349
- AnalisaColisoes()**, função 403
- análise amortizada
 análise assintótica vs. 271
 de array dinâmico 277–281. *V. também* análise amortizada de array dinâmico
 de árvore afunilada 281–285. *V. também* análise amortizada de árvore afunilada
 comparação com análise assintótica de caso médio 271
 comparação de métodos de 274
 custo amortizado em 270
 método contábil de 272–273, 276, 279–280
 método de agregado de 272, 275–276, 278–279
 método de potencial de 273–274, 276–277, 280, 281–285
 moeda virtual de 272
 quando usar 271
 sequência de operações em 270
 de tabela de destruição 275
- análise amortizada de array dinâmico 277–278
 efeito de crescimento geométrico na 280–281
 usando método contábil 279–280
 usando método de agregado 278–279
 usando método de potencial 280
- análise amortizada de árvore afunilada
 de caso zig 282
 de caso zig-zag 283
 de caso zig-zig 284
 custo temporal de afunilamento 282
 custo temporal de inserção/remoção 285
 função potencial em 281
 posto de nó em 281
 tamanho de nó em 281
- análise assintótica 270, 271, 277, 344. *V. também* análise de algoritmo
- análise de algoritmo
 amortizada. *V. análise amortizada*
 de array dinâmico. *V. análise amortizada de array dinâmico*
 de árvore afunilada. *V. análise amortizada de árvore afunilada*
 de árvore AVL 232–233
 de árvore B 329–331
 de árvore B+ 344–345
 de árvore binária ordinária de busca 213
 Boyer e Moore, de casamento de 472
 de **BUBBLESORT** 585–586
 de **BUCKETSORT** 617
 de busca binária 162–164
 de busca sequencial 151–152
 de **COUNTINGSORT** 613–614
 de dispersão com encadeamento 375–376
 de dispersão com endereçamento aberto 384–386
 de dispersão cuco 398–399
 de dispersão estática em memória secundária 427
 de dispersão extensível em memória secundária 445–446
 espacial. *V. custo espacial*
 de fila de prioridade 548
 força bruta, de casamento de strings por 457–458
 de heap binário 547–548
 de **HEAPSORT** 609–610
 Horspool, de casamento (ou simplificação) de 476–477
 de inserção massiva em árvores B+ 683
 de **INSERTIONSORT** 590–591
 de intercalação binária de arquivos 659
 de intercalação múltipla de arquivos 672–673
 de intercalação múltipla de arrays 662
 Karp e Rabin, de casamento de 484–485
 Knuth, Morris e Pratt, de casamento de 463–464
 limite inferior de ordenação externa e 673–674
 de lista com saltos 178–180
 em memória principal vs. em memória secundária 84–86
 em memória secundária 297–298
 de **MERGESORT** 604–605
 notação usada em memória secundária para 298
 de **QUICKSORT** 596–599
 de **RADIXSORT** 621
 de redimensionamento de dispersão 388
 regra da soma de 157, 232, 605
 regra do produto de 586, 605
 de **SELECTIONSORT** 588
 tabela de busca encadeada, de operação em 155–156
 tabela de busca indexada, de operação em 151–152
 temporal. *V. custo temporal*
 de trie 497–499
- análise de dispersão 399–400
- análise espacial. *V. custo espacial*
- análise temporal. *V. custo temporal*
- antecessor imediato (nó) 203
- aplicativo 143
- ApresentaHora()**, função 569
- arquitetura de computador 67–71

- arquivo 64, 94
 - abertura de 96–100
 - acesso direto a 106, 111–115
 - atualização de registro de 121–124
 - binário. *V.* arquivo binário
 - buffer associado a 95
 - cluster de 64
 - conversão de texto para binário 124–125
 - cópia de 120–121
 - CSV 503
 - de dados. *V.* arquivo de dados
 - erro em processamento de 100–101
 - escrita em 107–115
 - escrita para leitura, passagem de 114
 - FASTA 697
 - fechamento de 99–100
 - formato de 94
 - fragmentação de 65
 - grande 117–118
 - indicador de erro em processamento de 95
 - indicador de final de 95
 - indicador de posição de 95, 111–115, 116
 - índice de byte em 118
 - índice de registro em 118
 - leitura em 107–115
 - leitura para escrita de, passagem de 114
 - organização de 65
 - posicionamento em 111–115, 116
 - processamento de. *V.* processamento de arquivo
 - rebatismo de 105
 - registro de. *V.* registro de arquivo
 - remoção de 105
 - temporário 104–105, 650, 674
 - de texto. *V.* arquivo de texto
 - texto para binário, conversão de 124–125
- arquivo binário 94. *V. também* arquivo de dados
 - criação de 124–125
 - escrita em 107–110
 - leitura em 107–115
- arquivo de dados 693–697
 - CensoMEC.bin** 696–697
 - CEPs.bin** 694–696
 - DNA.txt** 697
 - Machado.txt** 697
 - Tudor.txt** 693–694
- arquivo de texto 94. *V. também* arquivo de dados
 - atualização de 121–124
 - conversão em arquivo binário de 124–125
 - escrita em 107–111
 - em formato FASTA 697
 - leitura em 107–111
 - linha vazia de 118
 - salto de linha de 118–120
- array bidimensional 79–80
 - acesso por coluna de 80
 - acesso por linha de 79
 - padrão de referência em processamento de 80–81
- array dinâmico 145, 148, 277–281
- árvore afunilada 233–234
 - afunilamento de 234–241. *V. também* afunilamento de nó
 - análise amortizada de. *V.* análise amortizada de árvore afunilada
 - análise de 249. *V. também* análise amortizada de árvore afunilada
 - busca em 241–242
 - implementação de 244–249
 - inserção em 242–243
 - posto de nó de 281
 - remoção em 243–244
 - tamanho de nó de 281
- árvore autoajustável 233. *V. também* árvore afunilada; *V. também* árvore AVL
- árvore AVL 217–218
 - altura de 232
 - análise de 232–233
 - árvore perfeitamente balanceada vs. 217
 - balanceamento de nó 217, 255–256
 - busca em 229
 - caminho de inserção e remoção remoção em 218
 - desbalanceamento de 219–228. *V. também* desbalanceamento de árvore binária de busca
 - implementação de 228–232
 - inserção em 218–225
 - propagação de desbalanceamento em 226
 - rebalanceamento de 219–228
 - remoção em 225–228
 - verificação de balanceamento de 255–256
- árvore B 309
 - altura de 349–350
 - análise de 329–331
 - busca em 309, 329–331
 - divisão de nó de 310
 - grau mínimo de 319
 - inserção em 309–313
 - junção de nós de 314–315
 - maior e menor chaves de 347–348
 - número de chaves de 349
 - número de nós de 348–349
 - persistência de dados em 328–329
 - profundidade de 349–350
 - programa-cliente de 329
 - remoção em 313–319
 - com tendência direita 310
 - com tendência esquerda 310
 - variante de 345–346
- árvore B* 345–347
- árvore B# 346
- árvore B+ 331–332
 - análise de 344–345
 - bulkloading de 674–683
 - busca de intervalo em 342–343, 350–352
 - busca em 332
 - conjunto de índices de 332
 - conjunto sequencial de 332

- custo de transferência de 344–345
- implementação de 343–344
- inserção em 332–336
- inserção massiva em 674–683
- primeira chave de folha de 336
- remoção em 336–342
- árvore binária
 - afunilada 233–234. *V. também* árvore afunilada AVL 217–218. *V. também* árvore AVL
 - de busca. *V. árvore binária de busca completa* 537. *V. também* árvore binária completa de busca 196
 - degenerada 196
 - inclinada 216
 - perfeitamente balanceada 216
 - repleta 609
 - rubro-negra xlix, 196
- árvore binária completa
 - array, implementada com 538
 - filho de nó de 539
 - como heap binário 538. *V. também* heap binário
 - pai de nó de 537
- árvore binária de busca 196
 - afunilada 233–234. *V. também* árvore afunilada AVL 217–218. *V. também* árvore AVL
 - ordinária 196–198. *V. também* árvore binária ordinária de busca
 - rubro-negra xlix, 196
- árvore binária ordinária de busca 196–198
 - análise de 213
 - busca em 200–201
 - caminhamento em ordem infixada em 197–198
 - implementação de 203–212
 - iniciação de 204
 - inserção em 198–200
 - maior chave de 251–252
 - menor chave de 251–252
 - ordenação de chaves de 197–198, 252–253
 - remoção em 201–203
 - validação de 253–255
- árvore multidirecional ascendente 311. *V. também* árvore B; *V. também* árvore B+
- árvore multidirecional descendente de busca 288–290, 298
 - análise de 295
 - balanceada 289
 - busca em 290–291, 302–303
 - caminhamento em 347
 - chave de 288
 - dimensionamento de grau de 298–300
 - escrita de nó de 301
 - filho de chave de 288
 - folha de 289
 - grau de 288
 - implementação em memória principal de 294–295
 - implementação em memória secundária de 298–309
 - inserção em 292–293, 303–306
 - leitura de nó de 301
 - maior/menor chave de 347–348
 - nó completo/incompleto de 289
 - ordem de 288
 - remoção em 293–294, 306–309
 - semifolha de 289
 - tratamento de exceção em 300–301
- árvore rubro-negra xlix, 196
- árvores binárias de busca, comparação de 249–251
- árvores multidirecionais de busca, comparação de 346
- árvore ternária de busca 499
- ASSEGURA**, macro 300
- AssociaPonteiros()**, função 633
- "at"**, modo de abertura de arquivo 98
- ATENDIMENTO**, constante 567
- atoi()**, função de biblioteca 162
- AtualizaArquivo()**, função 123
- atualização de registro de arquivo de texto 121–124
- ATUALIZAÍNDICEEMÁRVOREB+**, algoritmo 340
- avaliação de algoritmo. *V. análise de algoritmo*
- avaliação experimental
 - de intercalação binária de arquivos 658–659
 - de lista com saltos 180–181
 - de método de ordenação 627–631

B

- backup (de dados) 67
- baixo nível, operação de 699–712. *V. também* bit; *V. também* operador de bits
- BalanceamentoAVL()**, função 230
- balanceamento de árvore binária 216–217. *V. também* árvore AVL
 - cálculo de 230
 - manutenção de. *V. árvore AVL*
 - de nó 217
 - perfeito 216
 - restabelecimento de. *V. árvore AVL*
 - verificação de 255–256
- bandeira holandesa, problema da 634–635
- barramento 67, 68
- barra vertical (símbolo) disjunção exclusiva de bits 700
- BASE**, constante 483, 522
- base numérica 699–700
- BASE_RADIX**, constante 619
- bastão de memória 66
- Bernstein, função de dispersão de 714
- bit
 - consulta de 706–707
 - cópia de 701–703
 - desligação de 705–706
 - deslocamento de 701
 - inversão de 706
 - ligação de 705–706
 - LSB (bit menos significativo) 703
 - mais significativo 704–705
 - maskamento de 701–703
 - menos significativo 703

- bit (*continuação*)
 - MSB (bit mais significativo) 704–705
 - operador de. *V.* operador de bits
 - representação de valor do tipo **int** 707–708
- blocagem 81–83
- bloco
 - de disco magnético 64–65, 82, 296
 - de memória principal 109–111
- BM**, algoritmo 467. *V. também* Boyer e Moore, algoritmo de casamento de
- BMH**, algoritmo 475. *V. também* Horspool, algoritmo de casamento (ou simplificação) de
- bolha, método de ordenação da. *V.* **BUBBLESORT**, algoritmo
- bom sufixo 465–466
- Boyer e Moore, algoritmo de casamento de 464–468
 - análise de 472
 - heurística de bom sufixo de 465–467
 - heurística de mau caractere de 465
 - implementação de 468–472
- bsearch()**, função de biblioteca 160–161, 253
- BUBBLESORT**, algoritmo 584–585, 634
 - análise de 585–586
 - implementação de 585
- BubbleSort()**, função 585
- bucket (coletor) 360, 420. *V. também* **BUCKETSORT**, algoritmo
- BUCKETSORT**, algoritmo 615–616
 - análise de 617
 - implementação de 616–617
- BucketSort()**, função 616
- buffer
 - área de 95, 101
 - descarga de 99, 117, 666
 - de entrada 665, 666
 - fflush()** e 101
 - recarga de 666
 - de saída 652, 665, 666
 - stream de atualização e 102
 - stream de entrada e 102
 - stream de saída e 101
 - tamanho de (em ordenação em memória secundária) 672
 - tipos de 101
 - uso de 658
- buffering 101
- buffering e bloco de disco magnético 65
- bulkloading 674. *V. também* inserção massiva em árvores B+
- BULKLOADINGDEÁRVORES B**, algoritmo 780
- busca 140
 - algoritmo de 141
 - em árvore afunilada 241
 - em árvore AVL 229
 - em árvore B 309
 - em árvore B+ 332
 - em árvore binária ordinária de busca 200–201
 - em árvore multidirecional descendente 290, 302–303
 - bem-sucedida 141
 - binária 159–161. *V. também* busca binária
 - chave de 141
 - com chave secundária 184–187
 - dedilhada 142
 - em dispersão com encadeamento 371
 - em dispersão com endereçamento aberto 380
 - em dispersão cuco 392
 - em dispersão estática em memória secundária 421
 - em dispersão extensível em memória secundária 432
 - externa 142
 - de Fibonacci 181–184
 - hierárquica em memória principal 196. *V. também* árvore AVL; *V. também* árvore binária ordinária de busca; *V. também* árvore afunilada
 - hierárquica em memória secundária 288. *V. também* árvore B; *V. também* árvore B+
 - interna 141
 - por interpolação 161
 - de intervalo 142, 188, 350–352, 446
 - linear 140. *V. também* busca linear
 - em lista com saltos 167–168
 - de maior chave 143
 - malsucedida 141
 - em memória principal 140
 - de menor chave 142
 - de piso de chave 142, 187
 - sequencial 145. *V. também* busca sequencial
 - sequencial com movimentação para início 154–155
 - sequencial com transposição 153–154
 - em tabela encadeada 153–156
 - em tabela indexada 145, 159–161
 - em tabela ordenada 159–161. *V. também* busca binária
 - de teto de chave 142
 - em trie 488–489
- BuscaArvoreBB()**, função 204
- BuscaArvoreFunil()**, função 246
- BuscaB()**, função 323
- busca binária 159–160
 - análise de 162–164
 - usando **bsearch()** 160–161
 - implementação de 160
 - número máximo de comparações de chaves em 163
 - overflow em 160
- BUSCABINÁRIA**, algoritmo 159
- BuscaBinariaIdx()**, função 160, 162
- BuscaComMovimentoLSE()**, função 155
- BuscaComTransposicaoLSE()**, função 154
- BuscaCuco()**, função 395
- BuscaDEA()**, função 383
- BUSCADEFIBONACCI**, algoritmo 181
- BuscaDE()**, função 373
- busca de intervalo 142, 446
 - em árvore B+ 342, 350–352
 - em lista simplesmente encadeada 188
- BuscaDEst()**, função 424
- BuscaDExt()**, função 438
- BUSCAEMÁRVOREAFUNILADA**, algoritmo 241
- BUSCAEMÁRVOREB+**, algoritmo 332
- BUSCAEMÁRVOREBINÁRIADEBUSCA**, algoritmo 201

BUSCAEMÁRVOREMULTIDIRECIONAL, algoritmo 290
BUSCAEMDISPERSÃOCOMENCADEAMENTO, algoritmo 372
BUSCAEMDISPERSÃOCOMENDEREÇAMENTOABERTO, algoritmo 380
BUSCAEMDISPERSÃOCUCO, algoritmo 392
BUSCAEMLISTACOMALTOS, algoritmo 167–168
BuscaEmNoMultiMS(), função 302, 320
BUSCAEMTABELADEDISPERSÃOESTÁTICA, algoritmo 421
BUSCAEMTABELADEDISPERSÃOEXTENSÍVEL, algoritmo 432
BUSCAEMTRIE, algoritmo 489
BuscaEmTrie(), função 495, 513
BuscaFibonacciIdx(), função 183
BuscaInterpolacaoIdx(), função 162
BuscaIntervaloBM(), função 350
BuscaIntervaloLSE(), função 188
busca linear

- binária 159–161. *V. também* busca binária de Fibonacci 181–184
- por interpolação 161–162
- sequencial 145
- em tabela ordenada 159–166. *V. também* busca binária em tabela sem ordenação 145. *V. também* busca sequencial

BuscaListaComSaltos(), função 172
BuscaListaSE(), função 374
BuscaMultiMS(), função 302
BuscaPisoIdx(), função 187
busca por interpolação 161–162
BuscaSecundariaLSE(), função 185, 188
busca sequencial 146–148
BUSCASEQUENCIAL, algoritmo 145
BuscaSequencialIdx() 146
BuscaTrieMachado(), função 506
byte, índice de 118

C

cache 73–76
caching 73–76
cadeia de coletores excedentes 421
cadeia de DNA 513
caminhamento infix

- em árvore binária de busca 197, 214
- em árvore multidirecional de busca 347

CaminhamentoInfixoBB(), função 252
CaminhamentoInfixoB(), função 347
CaraOuCoroa(), função 175
CarregaBufferMS(), função 658, 671
CasaFluxoContínuo(), função 523
casamento

- de chaves 141
- de padrões. *V.* casamento de strings
- de palavras 500
- de strings 452. *V. também* casamento de strings

CasamentoBM(), função 471
CasamentoBMH(), função 476
casamento de palavras vs. casamento de strings 500–501

casamento de strings

- alfabeto em 452
- borda em 453
- de Boyer e Moore (**BM**). *V.* Boyer e Moore, algoritmo de casamento de
- casamento de palavras vs. 500–502
- comparação de algoritmos de 485–486
- deslizamento de padrão em 453
- falso 477
- em fluxo contínuo 523–525
- de força bruta (**FB**). *V.* força bruta, algoritmo de casamento de strings por
- de Horspool (**BMH**). *V.* Horspool, algoritmo de casamento (ou simplificação) de
- por impressão digital 477. *V. também* Karp e Rabin, algoritmo de casamento de
- de Karp e Rabin (**KR**). *V.* Karp e Rabin, algoritmo de casamento de
- de Knuth, Morris e Pratt (**KMP**). *V.* Knuth, Morris e Pratt, algoritmo de casamento de
- léxico 519–522
- padrão em 452
- prefixo em 453
- prefixo próprio em 453
- com retrocesso 454
- salto em 453
- substring em 452, 453
- sufixo em 453
- sufixo próprio em 453
- tamanho de alfabeto em 452
- texto em 452
- visualização de 453–454

CasamentoFB(), função 456
CasamentoKMP(), função 462
CasamentoKR(), função 482
casamento léxico de strings 519–522
CasamentoLexico(), função 521
censo escolar 696
CensoMEC.bin, arquivo de dados 696–697
CEPs.bin, arquivo de dados 694–697
CHAR_BIT, constante de biblioteca 560, 616, 707
chave 140, 580

- de árvore multidirecional descendente de busca 288
- de busca 141
- composta 365
- duplicada 636
- externa 141
- filho direito/esquerdo de 288
- interna 141, 696, 697
- piso de 187
- primária 140
- secundária 140
- valor de 140

CHEGADA, constante de enumeração 567
Clang, compilador 118
classificação 580. *V. também* ordenação
clearerr(), função de biblioteca 101

- cliente, programa- 143
- CODIFICAARQUIVO**, algoritmo 554
- codificação de Huffman 551–566
- árvore de 554
 - cabeçalho de 552
 - canônica 552
 - decodificação da 563–566
 - fila de prioridade em 554
 - lista de códigos canônicos em 555
 - metadado de 552
 - obtenção de código em 555–563
 - padrão 552
- CodificaHuff()**, função 556
- código canônico 553
- ColetaPares()**, função 685
- coletor 360, 420. *V. também* **BUCKETSORT**, algoritmo; *V. também* dispersão com endereçamento aberto
- camarada 432
 - excedente 420, 434–436
 - primário 420
- colisão 362
- em dispersão com encadeamento 371
 - em dispersão com endereçamento aberto 377–380
 - em dispersão cuco 389–392
 - em dispersão em memória principal 362
 - em dispersão em memória secundária 420–421, 428–431
 - Karp e Rabin, no algoritmo de 477
 - resolução de 362, 377–380
- CompactaNoMultims()**, função 308, 327
- comparação
- entre os algoritmos **BM** e **BMH** 473
 - de algoritmos de casamento de strings 485
 - entre análise amortizada e análise assintótica de caso médio 271
 - entre árvores binárias de busca 249–251
 - entre árvores multidirecionais de busca 346
 - entre busca de Fibonacci e a busca binária 184
 - de chaves 580
 - entre codificação padrão e codificação canônica de Huffman 553
 - entre dispersão em memória secundária e árvores da família B 446
 - entre busca por interpolação e busca binária 165
 - função de 157, 547, 555, 561
 - de implementações de filas de prioridade 548
 - entre intercalação binária e múltiplice 668
 - entre lista com saltos e outras tabelas de busca 180
 - entre memórias SRAM e DRAM 57
 - de métodos de análise amortizada 274
 - de métodos de sondagem 386
 - ordenação baseada em 582–583, 615, 617, 621–624
 - entre tabela de dispersão e tabela indexada 361
 - entre tries e outros tipos de árvores de busca 498
- ComparaCEPs()**, função 157
- ComparaInts2()**, função 657
- ComparaInts()**, função 254, 547, 628
- ComparaIntsInv()**, função 628
- ComparaNosHeapHuff()**, função 555
- ComparaStr()**, função 686
- ComparaTamanhosHuff()**, função 561
- compilador
- Clang 118
 - GCC 118
- complemento, operador de 700
- complexidade de algoritmo. *V. análise de algoritmo*
- ComprimentoListaSE()**, função 186
- condição de erro em arquivo 100–101
- condição de exceção 116. *V. também* tratamento de exceção
- confusão em função de dispersão 368
- conjunção de bits, operador de 700
- conjunto de índices de árvore B+ 332
- conjunto sequencial de árvore B+ 332
- constante simbólica. *Procure uma constante simbólica específica pelo nome dela*
- notação para escrita de 719
- CONSTRÓIÁRVOREDEHUFFMAN**, algoritmo 551
- ConstroiNoArvoreBB()**, função 207, 255
- ConstroiNoArvoreHuff()**, função 558
- ConstroiTabelaIdx()**, função 158
- ConstroiTrieMachado()**, função 508
- construção. *V. criação*
- consulta. *V. também* busca
- de bit 706–707
 - em heap binário 543
 - de pertinência 404
- ConsultaBitEmArray()**, função 558
- ConsultaBit()**, função 558, 707
- contábil, método de análise amortizada 271, 272–273, 276, 279–280. *V. também* análise amortizada
- contagem de frequência 583
- ContemDuplicatasListaIdx()**, função 636
- ContemDuplicatasListaIdxOrd()**, função 636
- conteúdo de memória de buffer 99, 101
- controlador de disco 69
- controlador de USB 68
- conversão entre bases hexadecimal e binário 700
- CopiaArquivo()**, função 120, 658
- CopiaChavesB()**, função 321
- cópia de arquivo 120–121
- CopiaRestoArquivo()**, função 658
- corolário
- 3.1 (pior caso de construção de tabela indexada) 152
 - 3.2 (custo temporal de construção de tabela encadeada) 155
 - 3.3 (pior caso de busca binária) 164
 - 3.4 (custo temporal de busca por interpolação) 165
 - 4.1 (pior caso de operação em árvore binária ordinária de busca) 213
 - 5.1 (custo temporal amortizado de busca em árvore afunilada) 284
 - 5.2 (custo temporal amortizado de inserção em árvore afunilada) 285
 - 5.3 (custo temporal amortizado de remoção em árvore afunilada) 285
 - 6.1 (número de chaves num nível de árvore B) 330

- corolário (*continuação*)
 - 7.1 (nós visitados em inserção em dispersão com encadeamento com lista ordenada) 375
 - 11.1 (pior caso do algoritmo **QUICKSORT**) 597
- CORRECAO**, macro 343
- corrupção de memória usando **fgets()** 108
- COUNTINGSORT**, algoritmo 611–614
 - análise de 613–614
 - implementação de 612–613
- CountingSort()**, função 612
- CPU 67
- CriaArquivoApenasCEPs()**, função 628
- CriaArquivoBin()**, função 124
- CRIAÁRVOREB+COMBULKLOADING**, algoritmo 675
- CriaArvoreBM()**, função 679
- criação
 - de arquivo temporário 104–105
 - de árvore B 328
 - de árvore binária de busca 204
 - de árvore B+ usando inserção massiva 674–683
 - de árvore de codificação de Huffman 551
 - de coletor de tabela de dispersão extensível 434–435, 438
 - de filtro de Bloom 404
 - de heap binário 542–543, 608–610
 - de lista com saltos 167
 - de nó de árvore multidirecional em memória secundária 304, 320
 - de nó interno de árvore B+ 335
 - de nova raiz de árvore B 312
 - de série de intercalação binária em memória secundária 651
 - de série de intercalação múltipla em memória secundária 667
 - de tabela de bons sufixos 467
 - de tabela de busca com dispersão extensível 437
 - de tabela de busca indexada 146, 150–151
 - de tabela de busca indexada ordenada 166
 - de tabela de dispersão com encadeamento 373
 - de tabela de dispersão com endereçamento aberto 382
 - de tabela de dispersão cuco 394
 - de tabela de maiores bordas de string 459–460
 - de tabela de maus caracteres 467
 - de tabela de saltos 473
 - de trie 508–509
- CriaEvento()**, função 569
- CriaFiltro()**, função 406
- CriaHeapHuff()**, função 558
- CriaListaCanonicaHuff()**, função 560
- CriaPilha()**, função 328
- CRIASÉRIES**, algoritmo 651
- CriaSeriesMS()**, função 652
- CriaTabelaApenasCEPs()**, função 628
- CriaTabelaCuco()**, função 394
- CriaTabelaDEA()**, função 382
- CriaTabelaDE()**, função 373
- CRIA TABELA DESALTOSBMH**, algoritmo 475
- CriaTabelaIdx()**, função 146
- CRIA TB BONS SUFFIXOS**, algoritmo 467
- CRIA TB MAUS CARACTERES**, algoritmo 467
- CRIA TMB**, algoritmo 460
- CriaTMB()**, função 461, 524
- CSV, formato de arquivo 503, 696
- custo amortizado 270. *V. também* análise amortizada
- custo de entrada e saída 288. *V. também* custo de transferência
- custo de transferência 288
 - de árvore B 329–330
 - de árvore B+ 344
 - de tabela de dispersão extensível 445
- custo espacial
 - do algoritmo **BMH** 476
 - de árvore AVL 232
 - de árvore binária de busca 232
 - BUCKETSORT**, do algoritmo 617
 - COUNTINGSORT**, do algoritmo 614
 - de dispersão com endereçamento aberto 386
 - de dispersão cuco 399
 - de dispersão extensível em memória secundária 445–446
 - de intercalação de duas listas 604
 - de lista com saltos 179
 - em memória interna 84
 - em memória secundária 84, 85
 - MERGESORT**, do algoritmo 605
 - de **MERGESORT** para lista encadeada 603
 - QUICKSORT**, do algoritmo 598
 - RADIXSORT**, do algoritmo 621
 - de tabela de busca encadeada 156
 - de tabela de dispersão extensível 445
 - de trie 497, 498
- custo temporal
 - amortizado 270. *V. também* custo temporal amortizado
 - árvore afunilada, de operação em 249, 251. *V. também* custo temporal amortizado
 - árvore AVL, de operação em 232
 - árvore binária ordinária de busca, de operação em 200, 213
 - BUBBLESORT**, do algoritmo 585
 - BUCKETSORT**, do algoritmo 617
 - de busca binária 164
 - de busca de Fibonacci 184
 - de busca por interpolação 165
 - de busca sequencial 151
 - de cálculo de comprimento de MSC 517
 - de casamento de palavras usando trie 510
 - de casamento de strings de Boyer e Moore (**BM**) 472
 - de casamento de strings de Horspool (**BMH**) 476
 - de casamento de strings de Karp e Rabin (**KR**) 484
 - de casamento de strings de Knuth, Morris e Pratt (**KMP**) 463
 - de casamento de strings por força bruta (**FB**) 457
 - chave duplicada em lista indexada, de verificação de 636
 - COUNTINGSORT**, do algoritmo 613
 - de dispersão com encadeamento 371, 375, 376
 - de dispersão com endereçamento aberto 385–386
 - em dispersão cuco 399
 - de distância de edição 519
 - fila de prioridade, de operação em 537, 548

custo temporal (*continuação*)
 de função de dispersão 364
 heap binário, de inserção/remoção em 547
HEAPSORT, do algoritmo 609
INSERTIONSORT, do algoritmo 590, 591
 limite inferior para algoritmo baseado em comparações 621–624
 linear de ordenação 610–621
 linear logarítmico de ordenação 592–610
 lista com saltos, de operação em 179
MERGESORT, do algoritmo 604
 de ordenação de listas encadeadas com **MERGESORT** 603, 604
 de ordenação em memória secundária 650
 quadrático de ordenação 584–591
QUICKSORT, do algoritmo 597–598
RADIXSORT, do algoritmo 621
realloc(), da função de biblioteca 152
 de rotação em árvore binária 216
SELECTIONSORT, do algoritmo 588
 tabela de busca encadeada, de operação em 155, 156
 tabela de busca indexada, de operação em 151, 152, 157, 277. *V. também* custo temporal amortizado
 tabela de maiores bordas, de construção de 463
 tabela indexada ordenada, de inserção em 165
 trie, de operação em 497
 trivial de casamento de strings 457

custo temporal amortizado
 de afunilamento 282
 árvore afunilada, de operação em 284, 285
 de tabela de destruição 276
 tabela indexada dinâmica, de inserção em 279

D

DECODIFICAARQUIVO, algoritmo 563
DecodificaHuff(), função 565
 dependência de implementação
 arquivos abertos simultaneamente, do número de 97
 de **EOF** (constante de biblioteca) 100
 de **FILE** (tipo de biblioteca) 95
 desalojamento de bloco em caching 74
 desalojamento de cache 75
 desbalanceamento de árvore binária de busca
 direita-direita 221–222, 226
 direita-esquerda 222, 226
 esquerda-direita 220–221, 225–226
 esquerda-esquerda 219–220, 225
 inserção, devido a 219–225
 propagação de 226
 remoção, devido a 225–227

Desempilha(), função 328
DesligaBit(), função 705
 desligação de bit 705–706
DeslocaBitsEsquerda(), função 562
 deslocamento de bit, operador de 701
 à direita 701
 à esquerda 701

DespachaEvento(), função 570
DestroiArvoreBB(), função 255
DestroiArvoreHuff(), função 558
DestroiFiltro(), função 407
DestroiHeap(), função 544
DestroiHeapHuff(), função 558
DestroiListaSE(), função 186, 275
DestroiTabelaCuco(), função 398
DestroiTabelaIdx(), função 146
 destruição 143
 de árvore binária 255, 558
 de filtro de Bloom 407
 de heap binário 544, 558
 de lista simplesmente encadeada 186, 275
 tabela de 275–277
 de tabela de busca indexada 146
 de tabela de dispersão cuco 398

dicionário 140
difftime(), função de biblioteca 86
 difusão de função de dispersão 368
 digital de Rabin 365, 434, 478, 480
DigitalRabinErrada(), função 478
DigitalRabin(), função 479
 disco de estado sólido (SSD) 65
 disco magnético 58–65
 atraso rotacional de 62
 bloco de 64–65
 cabeça de leitura e escrita de 58
 capacidade de 60–61
 cluster de 64
 controlador de 69
 densidade de 60
 espaço ocioso em 64
 formatação de 61
 fragmentação de 65
 geometria de superfície de 59–60
 latência rotacional de 62
 MTTF de 64
 prato de 58
 SATA 69
 SCSI 69
 seek time de 61
 setor de 59–60
 superfície de 58
 tamanho de bloco de 82
 taxa de transferência de dados de 63
 tempo de acesso de 61
 tempo de posicionamento de 61
 tempo de transferência de 62
 tempo médio de falha de 64
 tempo médio de latência de 62
 trilha de 59

disco rígido. *V.* disco magnético
 disco SSD 65
 disjunção de bits 700
 disjunção exclusiva de bits (xor) 366, 700

- dispersão
- algoritmo de 360
 - análise de 399–400
 - colisão em 362
 - cuco 388. *V. também* dispersão cuco
 - com encadeamento 362. *V. também* dispersão com encadeamento
 - com endereçamento aberto 362. *V. também* dispersão com endereçamento aberto
 - estática. *V. dispersão* estática em memória secundária
 - extensível. *V. dispersão* extensível em memória secundária
 - fator de carga de 362
 - função de 360. *V. também* função de dispersão em memória principal. *V. dispersão* em memória principal em memória secundária. *V. dispersão* em memória secundária
 - método de 360
 - modular 365
 - polinomial 366
 - redimensionamento de 386. *V. também* redimensionamento de dispersão
 - com sondagem 362. *V. também* dispersão com endereçamento aberto
 - tabela de 360
 - valor de 360
- dispersão com encadeamento 362, 371
- análise de 375–376
 - busca em 371–372
 - fator de carga em 371, 376
 - implementação de 372–375
 - inserção em 372
 - redimensionamento de tabela de. *V. redimensionamento* de dispersão
 - remoção em 372
- dispersão com endereçamento aberto 376–377
- análise de 384–386
 - busca em 380
 - comparação de métodos de sondagem de 386
 - custo espacial de 386
 - custo temporal de 385–386
 - fator de carga de 376, 378, 385
 - implementação de 382–384
 - inserção em 381
 - redimensionamento de tabela de. *V. redimensionamento* de dispersão
 - remoção em 381
 - resolução de colisão em 377–380
 - sondagem dupla em 379–380
 - sondagem linear em 377–378
 - sondagem quadrática em 378–379
- dispersão cuco 388–392
- análise de 398–399
 - busca em 392
 - custo temporal amortizado de 399
 - desalojamento em 388
 - desvantagem de 399
 - implementação de 394–398
 - inserção em 392–393
 - pássaro cuco, analogia com 388
 - remoção em 393–394
 - vantagem de 399
- DispersaoDJB2()**, função 714
- DispersaoDJB()**, função 714
- dispersão em memória principal
- aplicações de 362–363
 - cuco 388–399. *V. também* dispersão cuco
 - com encadeamento 371–376. *V. também* dispersão com encadeamento
 - com endereçamento aberto 376–386. *V. também* dispersão com endereçamento aberto
 - fator de carga de 362. *V. também* fator de carga
 - redimensionamento de 386–388. *V. também* redimensionamento de dispersão
 - com sondagem 376–386. *V. também* dispersão com endereçamento aberto
- dispersão em memória secundária 420
- avaliação de 446
 - estática 420–427. *V. também* dispersão estática em memória secundária
 - extensível 427–446. *V. também* dispersão extensível em memória secundária
- dispersão estática em memória secundária 420
- análise de 427
 - busca em 421
 - coletor de 420
 - coletor excedente/primário de 420
 - implementação de 422–427
 - inserção em 420–421
 - número de coletores primários vazios em 446–447
 - remoção em 422
- dispersão extensível em memória secundária 427–428
- análise de 445–446
 - busca em 432
 - coletor camarada de 432
 - coletor excedente de 434–436
 - custo espacial de 445–446
 - desvantagem de 445
 - uso de digital de Rabin em 434
 - diretório de 427
 - implementação de 437–445
 - índice de 427
 - inserção em 428–431
 - maior profundidade local em 447
 - profundidade de diretório de 428
 - profundidade global de 428
 - profundidade local de 428
 - remoção em 432–433
 - tabela de 427
 - vantagem de 445
- DispersaoFNV()**, função 715
- DispersaoJOAAT()**, função 714
- DispersaoJSW()**, função 716
- DispersaoMisturaJunior()**, função 710
- DispersaoMisturaMedio()**, função 711
- DispersaoMultiplicacao()**, função 367

dispersão por mistura 708–712
DISPERSÃO POR MISTURA, algoritmo 708
 dispersão rotativa 368
DispersaoRotativa(), função 368
DispersaoSAX(), função 715
 dispersão xor 367–368
DispersaoXOR(), função 367
DispersoesLexico(), função 520
 dispositivo de entrada 68
 dispositivo de saída 68
 dispositivo periférico 94
 arquivo e 94
 buffer associado a 99
 de entrada 94
 padrão 102
 de saída 94
 stream e 94
 distância de edição 517–519
DistanciaDeEdicao(), função 518
 distância de Levenshtein 518
 distribuição exponencial 550
DIVIDE COLETOR DE DISPERSÃO EXTENSÍVEL, algoritmo 431
DivideColetorDExt(), função 440
DivideNoB(), função 322
DIVIDE NÓ EM ÁRVORE B, algoritmo 313
DIVIDE NÓ INTERNO B+, algoritmo 676
 divisão de nó
 de árvore B 310, 313
 de árvore B+ 332–336, 676
 divisão e conquista, algoritmo/paradigma de 624
 divisão modular 366–367
 DJB 714
 DJB2 714
 DMA 69
DNA.txt, arquivo de dados 697
 DRAM 56–57. *V. também* memória
DRand(), função 569

E

e comercial (símbolo) conjunção de bits, usado em operador de 700
 EEPROM 57. *V. também* memória
EhArvoreAVL(), função 255
EhArvoreBinDeBusca(), função 253
EhNoFinalDeTrie(), função 495
EhNoFinalTrieMachado(), função 505
EhNoVazioDeTrie(), função 495
 elemento ativo (em intercalação múltipla de arquivos) 665
ElementoTopo(), função 328
Empilha(), função 328
EncontraCaminhoB(), função 320
EncontraNoMultiMS(), função 305, 310
 end of file (EOF) 100
 entrada, dispositivo de 68
 entrada e saída
 meio de 102

 operação de 69–71
EOF, constante de biblioteca 100–101, 103, 117
 EPROM 57. *V. também* memória
 equação de recorrência. *V. relação de recorrência*
 erro falso-negativo 404
 erro falso-positivo 404
ERRO_FREAD, macro 301
ERRO_FWRITE, macro 301
ERRO_OPEN, macro 300
ERRO_POSICAO, macro 301
ERRO_SEEK, macro 300
ERRO_STREAM_NULL, macro 301
ERRO_TELL, macro 301
EscreveCabecalhoHuff(), função 563
EscreveColetorDExt(), função 424, 440
EscreveColetorDExt(), função 440
EscreveEmArquivoArvoreBB(), função 252
EscreveNoMultiMS(), função 301, 320
EscreveRaizB(), função 329
 escrita de dados
 em arquivo binário 107–110, 124–125
 em arquivo de texto 107–111
 formatada 102–103. *V. também* escrita formatada
 escrita formatada 102
 fprintf() e, função de biblioteca 103
 printf() e, função de biblioteca 103
 sprintf() e, função de biblioteca 103–104
 estado interno de valor de dispersão 368
EstaVazialSE(), função 186
 estilo de escrita de identificador 719–720
 estrutura de dados
 em memória interna 140. *V. também* estrutura de dados em memória principal
 em memória principal 140, 360, 486, 536. *V. também* estrutura de dados em memória principal
 em memória secundária 84–86, 288, 420. *V. também* estrutura de dados em memória secundária
 probabilística 167
 estrutura de dados em memória principal
 árvore AVL 217–233, 255–256. *V. também* árvore AVL
 árvore binária afunilada 233–249. *V. também* árvore afunilada
 árvore binária ordinária de busca 196–213, 251–254. *V. também* árvore binária ordinária de busca
 vs. estrutura de dados em memória secundária 297
 fila de prioridade 536–537, 548. *V. também* fila de prioridade; *V. também* heap binário
 heap binário 537–548, 605–610, 659–662, 662–673. *V. também* heap binário
 lista com saltos 166–181
 lista simplesmente encadeada 603–604, 632–633. *V. também* tabela de busca encadeada
 tabela de destruição 275–277
 tabela de dispersão com encadeamento 371–376. *V. também* dispersão com encadeamento
 tabela de dispersão com endereçamento aberto 376–386. *V. também* dispersão com endereçamento aberto

estrutura de dados em memória principal (*continuação*)
 tabela de dispersão cuco 388–399. *V. também* dispersão cuco
 tabela indexada ordenada 156–166, 181–184. *V. também* tabela de busca indexada
 tabela indexada sem ordenação 145–152, 184–188. *V. também* tabela de busca indexada
 tabela simplesmente encadeada 152–156. *V. também* tabela de busca encadeada
 trie 486–500, 504–510. *V. também* trie
 estrutura de dados em memória secundária 295–298
 árvore B 309–331. *V. também* árvore B
 árvore B* 345–346
 árvore B+ 331–345, 350–352, 674–683. *V. também* árvore B+
 árvore multidirecional descendente de busca 288–295, 298–309, 347–348. *V. também* árvore multidirecional descendente de busca
 bloco de 296
 encadeada 297
 vs. estrutura de dados em memória principal 297
 modelo para análise de 297–298
 página de 296
 tabela de dispersão estática 420–427. *V. também* dispersão estática em memória secundária
 tabela de dispersão extensível 427–446, 447. *V. também* dispersão extensível em memória secundária
 estrutura (variável), preenchimento de 298–300
EsvaziaPilha(), função 328
 evento de simulação discreta 549
 exceção 116. *V. também* tratamento de exceção
EXCEDENTE, constante de enumeração 422
 exemplo de programação
 altura de árvore B 349–350
 atualizando de registro de arquivo de texto 121–124
 bandeira holandesa, problema da 634–635
 busca com chave secundária 184–188
 busca de Fibonacci 181–188
 busca de intervalo 188
 busca de intervalo em árvore B+ 350–352
 busca de piso de chave 187–188
 caminhamento em árvore multidirecional de busca 347
 casamento de strings em fluxo contínuo 523–525
 casamento léxico 519–522
 chaves duplicadas 636
 checando árvore binária de busca 253–255
 codificação de Huffman 551–566
 coletores excedentes em tabela de dispersão estática 446–447
 conferindo balanceamento AVL 255–256
 conversão de arquivo de texto em binário 124–125
 cópia de arquivo 120–121
 distância de edição 517–519
 exibição ordenada de chaves de árvore binária de busca 252–253
 filtro de Bloom 404–408
 HeapBurger, lanchonete 566–572
 Huffman, codificação de 551–566
 indexação de palavras do romance Ressurreição 504–510

lanchonete HeapBurger 566–572
 lista bitônica 636–637
 lista simplesmente encadeada, ordenação de 632–633
 Machado de Assis 504–510
 maior prefixo comum (MPC) a um conjunto de strings 510–513
 maior profundidade local em tabela de dispersão extensível 447
 maior subsequência comum (MSC) a dois strings 513–517
 medindo tempo de execução 86
 menor e maior chaves de árvore binária de busca 251–252
 menor e maior chaves de árvore multidirecional de busca 347–348
 MPC (maior prefixo comum) 510–513
 MSC (maior subsequência comum) 513–517
 número de chaves de árvore B 349
 número de nós de árvore B 348–349
 ordenação de arquivos por indexação 684–687
 ordenação de lista simplesmente encadeada 632–633
 ordenação de ponteiros 633–634
 ordenação em memória secundária, teste de 687
 saltando linhas de um arquivo de texto 118–120
 separando um string em partes (tokens) 502–504
 testando função de dispersão pronta 400–404
 teste de ordenação em memória secundária 687
ExibeConteudoNoArvoreBB(), função 253
ExibeEncontrados(), função 186
ExibeMSC(), função 516
ExibeRegistrosOrdenados(), função 686
ExibeResultadoDeTeste(), função 629
 extração de bit 703–705

F

fase de coleta/distribuição de **BUCKETSORT** 615
 fase de pré-processamento de string 485
 fase de processamento de string 485
 FASTA, formato de arquivo 697
 fator de carga 362, 386
 em dispersão com encadeamento 371
 em dispersão com endereçamento aberto 376
FB, algoritmo 456. *V. também* força bruta, algoritmo de casamento de strings por
fclose(), função de biblioteca 99, 117
FechaArquivo(), função 99, 117, 558, 658
 fechamento de arquivo 99–100
feof(), função de biblioteca 100–101, 111
ferror(), função de biblioteca 100–101, 111, 117
fflush(), função de biblioteca 101–102, 114, 117
fgetc(), função de biblioteca 107, 111, 116
fgets(), função de biblioteca 107, 108, 111, 115
 fila de prioridade 536–537
 análise de 548
 aplicações de 536
 ascendente 536
 comparação de implementações de 548
 descendente 536

- fila de prioridade (*continuação*)
 - enfileiramento/denfileiramento em 536
 - implementação com árvore binária de busca balanceada 537
 - implementação com heap binário. *V.* heap binário
 - implementação com lista encadeada 537
 - implementação com lista indexada 537
 - inserção em 536
 - remoção em 536
- FILENAME_MAX**, constante de biblioteca 97
- FILE**, tipo de biblioteca 95
- FILHO_D**, macro 544
- FILHO_E**, macro 544
- filhos esquerdo e direito de chave 288
- filtro de Bloom 404–408
 - consulta de pertinência em 404
 - falso-negativo em 404
 - falso-positivo em 404
- final de arquivo 95, 100–101
- firmware 57
- fitas magnéticas 67
- F**, macro 343
- FNV**, função de dispersão 715
- folha
 - de árvore multidirecional 289
 - inserção em 292–294, 309–311
- FOLHA**, constante de enumeração 343
- fopen()**, função de biblioteca 69, 95, 96–97, 99, 117
- força bruta, algoritmo de casamento de strings por 455–456
 - análise de 457–458
 - implementação de 456–457
- formato de arquivo 94
 - binário 94
 - CSV 503, 696
 - FASTA 697
 - de texto 94
- Fowler, Noll e Vo, função de dispersão de 715
- fprintf()**, função de biblioteca 103, 107
- fputc()**, função de biblioteca 107, 120
- fputs()**, função de biblioteca 107, 109, 115
- fragmentação e bloco de disco magnético 65
- fread()**, função de biblioteca 107, 109–111, 115, 296, 300
- FREE**, macro 497
- fscanf()**, função de biblioteca 103, 104, 107
- fseek()**, função de biblioteca 101, 107, 111–115
- fsetpos()**, função de biblioteca 117
- ftell()**, função de biblioteca 107, 112–115, 117
- função. *Procure uma função específica pelo nome dela*
 - denominação de 719–720
 - de dispersão 360. *V. também* função de dispersão; *V. também* função de dispersão pronta
 - genérica 546
 - nome de 719
 - de posicionamento em arquivo 111–115
 - de sondagem 377
- função de dispersão 360, 363–370
 - anatomia de 368–369
 - avalanche de 364
 - bem elaborada 368–370
 - para chave composta 365
 - para chave inteira 365
 - para chave real 365
 - combinação/confusão de 369
 - determinismo de 364
 - eficiência de 364
 - independência de 364
 - método de cálculo de 366–368. *V. também* método de cálculo de dispersão
 - modular 365
 - primária 377
 - pronta 713–717. *V. também* função de dispersão pronta
 - propriedades desejáveis de 363–364
 - recomendações práticas para criação de 370
 - rolante 480
 - secundária 377
 - simplicidade de 364
 - de sondagem 377
 - para string 365
 - tabular 366
 - teste de 370, 400–404
 - uniformidade de 364
- função de dispersão pronta 713–717
 - avaliação de 716–717
 - DJB 714
 - DJB2 714
 - FNV 715
 - JOAAT 713–714
 - JSW 715–716
 - SAX 715
- fwrite()**, função de biblioteca 107, 109–111, 111, 115, 296

G

- GCC, compilador 118
- genoma humano 697
- G**, macro 300, 343

H

- H_ABERTURA**, constante 567
- hashing 360. *V. também* dispersão
- HD. *V.* disco magnético
- heap. *V.* heap binário
- heap binário 537–539
 - acréscimo de prioridade em 543
 - análise de 547–548
 - ascendente 537
 - consulta em 543
 - criação de 542–543
 - decréscimo de prioridade em 543
 - descendente 537
 - filho de nó de 539
 - implementação de 544–547
 - inserção em 539–541
 - de máximo/mínimo 537

heap binário (*continuação*)
 numeração de nós de 538–539
 ordenação usando 605–610. *V. também* **HEAPSORT**, algoritmo
 pai de nó de 538
 percolação ascendente em 540
 percolação descendente em 541
 propriedade de ordenação de 537
 propriedade estrutural de 537
 remoção de elemento específico em 543–544
 remoção em 541–543
 HeapBurger, lanchonete 566–572
HeapCheio(), função 545
HEAPSORT, algoritmo 605–610
 análise de 609–610
 implementação de 608–609
HeapSort(), função 608
HeapVazio(), função 547
HeapVazioHuff(), função 558
 heurística 75
 de bom sufixo 464
 de caching 75
 localidade de referência e 77
 de mau caractere 464
 de movimentação para início 152
 de transposição 152
 HG-19 (genoma) 697
 hierarquia de memória 71–73
 Horner, método de 366
 Horspool, algoritmo de casamento (ou simplificação) de
 472–474
 vs. algoritmo **BM** (diferença) 473
 análise de 476–477
 implementação de 475–476
 melhor caso de 477
 pior caso de 477
 tabela de saltos de 473
 Huffman, codificação de 551–566. *V. também* codificação de
 Huffman

I

identificador
 notação para escrita de 719–720
 implementação
 de afunilamento de nó 244–246
 de árvore afunilada 244–249
 de árvore AVL 228–232
 de árvore B 319–328
 de árvore B+ 343–344
 de árvore binária ordinária de busca 203–212
 de árvore multidirecional descendente de busca em memória
 principal 294–295
 de árvore multidirecional descendente de busca em memória
 secundária 298–309
 de **BUBBLESORT** 585
 de **BUCKETSORT** 616–617
 de busca binária 160

de busca por interpolação 162
 de busca sequencial com movimentação para início 154–155
 de busca sequencial com transposição 153–154
 de **COUNTINGSORT** 612–613
 de Boyer e Moore, algoritmo de casamento de 468–472
 de força bruta, algoritmo de casamento de strings por
 456–457
 de **HEAPSORT** 608–609
 de Horspool, algoritmo de casamento (ou simplificação) de
 475–476
 de redimensionamento de dispersão 387–388
 de dispersão com encadeamento 372–375
 de dispersão com endereçamento aberto 382–384
 de dispersão cuco 394–398
 de dispersão estática em memória secundária 422–427
 de dispersão extensível em memória secundária 437–445
 de heap binário 544–547
 de inserção massiva em árvores B+ 679–683
 de **INSERTIONSORT** 589–590
 de intercalação binária de arquivos 651–652
 de intercalação múltipla de arquivos 668–671
 de intercalação múltipla de arrays 660–662
 Karp e Rabin, do algoritmo de casamento de 482–484
 Knuth, Morris e Pratt, do algoritmo de casamento de
 461–463
 de lista bitônica 636–637
 de lista com saltos 170–178
 de **MERGESORT** 601–603
 de ordenação de lista simplesmente encadeada 632–633
 de ordenação de ponteiros 633–634
 de **QUICKSORT** 595–596
 de **RADIXSORT** 619–621
 de rotação em árvore binária de busca 215
 de **SELECTIONSORT** 587
 de tabela de busca encadeada 153–155
 de tabela de busca indexada 145–151, 158
 de tabela de maiores bordas 461
 de trie 492–497
IncrementaBits(), função 562
 indexação
 em arquivo 111, 118
 de byte em arquivo 112, 118
 de linha em arquivo 115
 de registro em arquivo 111, 118
INDICE_CARACTERE, macro 493
 índice de byte/registo em arquivo 118
 indução (matemática) 163, 179, 197, 622
IniciaArvoreBB(), função 204
 iniciação. *V. também* criação
 de árvore binária ordinária de busca 204
 de coletor de tabela de dispersão com endereçamento aberto
 376
 de filtro de Bloom 405
 de heap binário 544, 660
 de lista com saltos 171–172
 de lista encadeada 373, 505
 de nó de trie 494

- iniciação (*continuação*)
 - de tabela de busca com dispersão estática 424
 - de tabela de dispersão com encadeamento 373
 - de tabela de dispersão com endereçamento aberto 382
 - de tabela de dispersão cuco 394
 - de tabela de dispersão extensível 434, 437
 - de trie 493
 - de valor de dispersão 370
- IniciaHeap()**, função 544
- IniciaHeapIM_Arr()**, função 660
- IniciaListaComSalto()**, função 175
- IniciaListaComSaltos()**, função 171
- IniciaListaSE()**, função 373, 505
- IniciaNoMultiMS()**, função 304, 320, 325
- IniciaTabDEst()**, função 424
- IniciaTabelaDExt()**, função 437
- IniciaTrie()**, função 493
- inserção
 - em árvore afunilada 242–243
 - em árvore AVL 218–225
 - em árvore B 309–313
 - em árvore B+ 332–336
 - em árvore binária ordinária de busca 198–200
 - em árvore multidirecional descendente de busca 292–293, 303–306
 - em dispersão com encadeamento 372
 - em dispersão com endereçamento aberto 381
 - em dispersão cuco 392–393
 - em dispersão estática em memória secundária 420–421
 - em dispersão extensível em memória secundária 428–431
 - em heap binário 539–543
 - em lista com saltos 168–169
 - massiva. *V.* inserção massiva em árvores B+ ordenada em tabela de busca indexada 158
 - de registro de arquivo 115
 - em tabela de busca encadeada 155
 - em tabela de busca indexada 148–149
 - em trie 490–492
- inserção massiva em árvores B+ 650, 674, 676–679
 - algoritmos de 675–676
 - análise de 683
 - desvantagem de 683
 - implementação de 679–683
- InserAcimaBM()**, função 681
- INSEREMÁRVOREB+**, algoritmo 675
- InserArvoreBB()**, função 204
- InserArvoreFunil()**, função 247
- InserAVL()**, função 229
- InserB()**, função 324
- InserColetorVazioDExt()**, função 443
- InserCuco()**, função 395
- InserDEA()**, função 383
- InserDE()**, função 374
- InserDEst()**, função 425
- InserDExt()**, função 439
- INSEREMÁRVOREAFUNILADA**, algoritmo 242
- INSEREMÁRVOREAVL**, algoritmo 224
- INSEREMÁRVOREB**, algoritmo 312
- INSEREMÁRVOREB+**, algoritmo 334
- INSEREMÁRVOREBINÁRIADEBUSCA**, algoritmo 198
- INSEREMÁRVOREMULTIDIRECIONALDESCENDENTE**, algoritmo 292
- INSEREMDIRETÓRIODEDISPERSÃOEXTENSÍVEL**, algoritmo 431
- InserEmDiretorioDExt()**, função 442
- INSEREMDISPERSÃOCOMENCADEAMENTO**, algoritmo 372
- INSEREMDISPERSÃOCOMENDEREÇAMENTOABERTO**, algoritmo 381
- INSEREMDISPERSÃOCUCO**, algoritmo 392
- InserEmFolhaMultiMS()**, função 306, 310, 321
- INSEREMHEAPASCENDENTE**, algoritmo 540
- InserEmHeap()**, função 545, 570
- InserEmHeapHuff()**, função 558
- INSEREMLISTACOMSALTOS**, algoritmo 168, 169
- InserEmNoB()**, função 321
- InserEmOrdem()**, função 165
- INSEREMTABELADEDISPERSÃOESTÁTICA**, algoritmo 421
- INSEREMTABELADEDISPERSÃOEXTENSÍVEL**, algoritmo 431
- INSEREMTRIE**, algoritmo 490
- InserEmTrie()**, função 494
- InserLinhaEmTrieMachado()**, função 509
- InserListaComSaltos()**, função 173
- InserListaSE()**, função 275, 374
- InserMultiMS()**, função 303
- InserTrieMachado()**, função 506
- INSERTIONSORT**, algoritmo 589
 - análise de 590–591
 - implementação de 589–590
- InsertionSort()**, função 589
- Intercala2Tabelas()**, função 601
- IntercalaBinMS()**, função 654
- intercalação
 - de arquivos. *V.* intercalação binária de arquivos; *V. também* intercalação de arquivos; *V. também* intercalação múltipla de arquivos
 - de arrays 599–605, 659. *V. também* intercalação múltipla de arrays; *V. também* **MERGESORT**, algoritmo de listas encadeadas 603–604, 632–633
- INTERCALAÇÃOBINÁRIA**, algoritmo 651
- intercalação binária de arquivos 651–652
 - análise de 659
 - custo de transferência de 659
 - implementação de 652–659
- intercalação de arquivos
 - análise de 672–673
 - binária 651–659. *V. também* intercalação binária de arquivos
 - buffer de entrada em 652
 - buffer de entrada vazio em 666
 - buffer de saída em 652
 - buffer repleto em 666
 - descarga de buffer de saída de 666
 - fases de 650
 - implementação de 652–659, 668–671
 - múltipla 662–673. *V. também* intercalação múltipla de arquivos

intercalação de arquivos (*continuação*)
 passagem de 650
 série de 650

intercalação múltipla de arquivos 662–690
 análise de 672–690
 custo de transferência de 672–690
 exemplo de 664–668
 implementação de 668–690

INTERCALAÇÃO MÚLTIPLEX DE ARQUIVOS, algoritmo 663

intercalação múltipla de arrays 659
 algoritmo de 660
 análise de 662
 implementação de 660–662

INTERCALAÇÃO MÚLTIPLEX DE ARRAYS, algoritmo 660

IntercalaDuasSeriesMS(), função 655

IntercalaMultiArq(), função 669

IntercalaNArrays(), função 661

INTERCALA TABELAS, algoritmo 600

INTERNO, constante de enumeração 343

interrupção, sinal de 69

INTERVALO_CHEGADA, constante 567

IntervaloExponencial(), função 569

inversão de bit 706

inversão, estado de ordenação de 583

InverteBit(), função 706

InverteTabela(), função 628

ItemEstaEmFiltro(), função 408

J

JOAAT, função de dispersão 369, 713–714

JSW, função de dispersão 366, 715–716

Julienne Walker, função de dispersão de 715–716

junção de nós
 de árvore B 314–319
 de árvore B+ 336–342

JuntaNosB(), função 328

JUNTA NÓS DE ÁRVORE B, algoritmo 314

K

Karp e Rabin, algoritmo de casamento de 477–482
 análise de 484–485
 casamento falso em 477, 479
 digital de Rabin usada com 478
 função de dispersão rolante usado com 480
 implementação de 482–484
 método de Horner em 479
 ocorrência de overflow em 479

KMP, algoritmo 460. *V. também* Knuth, Morris e Pratt, algoritmo de casamento de

Knuth, Morris e Pratt, algoritmo de casamento de 458–461, 519–522
 análise de 463–464
 função de falha de 458
 implementação de 461–463
 tabela de maiores bordas de 458–459

tabela de prefixo de 458

KR, algoritmo 482, 519–522. *V. também* Karp e Rabin, algoritmo de casamento de

L

lanchonete HeapBurger 566–572

lapso de cache 74

LeCabecalhoHuff(), função 563

LeColetorDEst(), função 423, 439

LeColetorDExt(), função 439

leitura antecipada de bloco de disco magnético 65

leitura de dados em arquivo
 com acesso direto 111–115
 com acesso sequencial 107–111
 binário com acesso direto 107–115
 formatada 103
 de texto 107–111

LEITURA SEQUENCIAL DE ARQUIVO, algoritmo 110

LeLinhaIlimitada(), função 509

lema
 3.1 (piso de logaritmo) 162
 3.2 (piso de logaritmo de $n - 1$) 162
 3.3 (tamanho de tabela em busca binária) 163
 5.1 (custos real e amortizado de sequência de operações) 273
 5.2 (relação entre logaritmos) 282
 9.1 (número de janelas em texto) 457
 9.2 (custo temporal de construção de tabela de maiores bordas) 463
 9.3 (custo temporal de pré-processamento do algoritmo **KR**) 484
 11.1 (número de comparações de chaves no algoritmo **SELECTIONSORT**) 588
 11.2 (número máximo de comparações de chaves no algoritmo **INSERTIONSORT**) 590
 11.3 (caso médio de comparações e atribuições no algoritmo **INSERTIONSORT**) 590
 11.4 (número máximo de comparações de chaves no algoritmo IntercalaTabelas) 604
 11.5 (relação satisfeita pelo custo temporal do algoritmo **MERGESORT**) 604
 11.6 (convergência de série de potências) 609
 11.7 (número máximo de folhas de árvore binária) 622
 11.8 (altura mínima de árvore binária) 623

LeNo(), função 301

LeNome(), função 186

LeNoMultiMS(), função 297, 301, 320

LeRaizB(), função 328

Levenshtein, distância de 518

liberação de memória dinâmica 99. *V. também* destruição

LigaBit(), função 705

ligação de bit 705–706

limite inferior de algoritmo
 de ordenação baseado em comparação 621–624
 de ordenação externa 673–674

<limits.h>, cabeçalho de biblioteca 560, 707

linha de arquivo de texto, salto de 118–120

linha de cache 74
LinhaEmRegistro(), função 124, 125
 lista
 bitônica 636–637
 encadeada 166, 603–604, 632–633. *V. também* lista com saltos; *V. também* tabela de busca encadeada; *V. também* dispersão com encadeamento
 indexada. *V.* tabela de busca indexada
 com saltos 166–181. *V. também* lista com saltos
 lista com saltos 166–167
 análise de 178–180
 avaliação experimental de 180–181
 busca de 167–168
 implementação de 170–178
 inserção em 168–169
 nível de 166
 número de níveis vs. nível de 170
 perfeita 167
 quase perfeita 167
 real 167
 remoção em 169–170
 lista encadeada 107. *V. também* lista com saltos; *V. também* tabela de busca encadeada
 circular com cabeça 170
 simples 153
 localidade de referência
 de acesso a instrução 81
 blocagem e 81–83
 espacial 77–81
 padrão de referência sequencial e 77
 princípio de Pareto em 77
 de programa 84
 recomendação sobre 84
 uso de registrador e 83–84
 regra 80/20 e 77
 temporal 77–81
long int, tipo 117
 largura do 117
long long int, tipo 118, 479
 LSB (bit menos significativo) 703
L_tmpnam, constante de biblioteca 105

M

Machado de Assis 504, 697
Machado.txt, arquivo de dados 697
 macro. *Procure uma macro específica pelo nome dela*
 maior chave
 de árvore B 347–348
 de árvore binária de busca 251–252
 de árvore multidirecional de busca 347–348
MaiorChaveArvoreBB(), função 251
MaiorChaveB(), função 348
 maior prefixo comum (MPC) a um conjunto de strings 510–513
MaiorPrimo(), função 483, 522
 maior (símbolo) deslocamento à direita, usado em operador de 701
 maior subsequência comum (MSC) a dois strings 513–517
MAIOR_VALOR, macro 256
MARGEM, constante 567
 máscara de bits 702
 mascaramento, operação de 701–703
 mau caractere 465
MAX_BITS, constante 560
MAX_BYTES, constante 560
MAX_NOME, constante 122
MAX_PESSOAS_GRUPO, constante 567
MAX_REGISTROS, constante 653
MAX_SAND, constante 567
M, constante 422
 Mean Time To Failure (MTTF) 64
 mediana 594
 medida de tempo de execução 86
MedidaDeTempo(), função 86, 402
 meio de armazenamento 56–67. *V. também* memória
 disco magnético 58–65. *V. também* disco magnético
 fita magnética 67. *V. também* fita magnética
 pen drive 66
 SSD 65–66
 meio de entrada/saída 102
 memória
 acesso direto a 69
 bastão de 66
 bloco de 106
 cache 57, 73–76
 cache fria 78
 caching e 73–76
 contiguidade em 109
 dispositivo periférico e 110
 DRAM 56–57
 EEPROM 57
 EPROM 57
 flash 65
 interna 72
 módulo/pente de 56
 não volátil 57
 principal 57, 67, 72
 PROM 57
 RAM 56–57
 SRAM 56–57
 virtual 82
 volátil 57
 memória interna vs. memória principal 140
 menor chave
 de árvore B 347–348
 de árvore binária de busca 251–252
 de árvore multidirecional de busca 347–348
MenorChaveArvoreBB(), função 249, 251
MenorChaveMultims(), função 308, 347
MenorDe3(), função 518
MenorNoArvoreBB(), função 249

menor (símbolo) deslocamento à esquerda, usado em operador de 701

MERGE_SORT, algoritmo 599–601, 634

análise de 604–605

implementação de 601–603

ordenação de lista encadeada com 603–604

MergeSortAux(), função 602

MERGE_SORT_DE_LISTA_ENCADEADA, algoritmo 604

MergeSort(), função 603

método contábil 272–273, 276, 279–280. *V. também* análise amortizada

método de agregado 272, 275–276, 278–279. *V. também* análise amortizada

método de cálculo de dispersão

aditivo 366

de disjunção exclusiva 367–368

de divisão modular 366–367

por mistura 368, 708–712

de multiplicação 367

passo de combinação de 369

passo de confusão de 369

polinomial 366

de resto de divisão 366–367

rotativo 368

tabular 366

xor 367–368

método de dispersão 360. *V. também* método de cálculo de dispersão

método de Horner 479

método de ordenação da bolha. *V. BUBBLE_SORT*, algoritmo

método de potencial 273–274, 276–277, 280, 281–285. *V. também* análise amortizada

mod 361, 481

moda 583

modelo de memória externa padrão 297

modf(), função de biblioteca 365, 367

modo de abertura de arquivo

"a" 98

"a+" 98, 115

"a+b" 98, 115

"ab" 98

"a+t" 98, 115

"at" 98

para atualização 98

"r" 98

"r+" 98, 115

"r+b" 98, 115

"rb" 98

"r+t" 98, 115

"rt" 98

"w" 98

"w+" 98, 115

"w+b" 98, 115

"wb" 98, 121

"w+t" 98, 115

"wt" 98

módulo de biblioteca stdio 94, 117

moeda virtual 272

MoveApontador(), função 113, 302, 438, 658

MPC(), função 511

MSB (bit mais significativo) 704–705

MSBits(), função 616

Murphy, lei de 116, 300

N

NAleatorio(), função 570

N_BITS_SIG, constante 616

N_CADEIRAS, constante 567

N_COLETORES, constante 616

NColetoresExcedentesDEst(), função 446

níveis de hierarquia de memória 72

nó

alvo 234

antecessor de 203, 243

de árvore afunilada 243, 244

de árvore AVL 228

de árvore B 299, 309

de árvore B+ 331, 343

de árvore binária ordinária de busca 204

de árvore multidirecional descendente de busca 289, 294, 298

completo 289

dimensionamento de 299–300, 343

divisão de 310–313, 334–336

escrita de 302

filho de 197, 201, 210, 213–216, 539

final 487

folha 289, 331, 488

grau de 299

incompleto 289

junção de 314–319, 336–342

leitura de 301

de lista com saltos 166, 170

de lista simplesmente encadeada 153

pai de 213–214, 538

pivô de rotação 213

posto de 281

raiz de rotação 213

rotação de 213–216

semifolha 289

sucessor de 210, 243

de tabela de destruição 275

tamanho de 281

de trie 487–492

visitação de 197, 214, 252–253

nome

de constante de enumeração 719

de constante simbólica 719

de função 719

de macro 719

de rótulo de estrutura 719

de tipo definido pelo programador 719

NoNovo(), função 255

notação
 adotada neste livro 719
 análise de algoritmo em memória secundária, usada em 298
 camelo 719
 de identificador 719–720
NO_VAZIO, constante de enumeração 343
NovoColetorDEst(), função 423
NovoColetorDExt(), função 438
NovoNoDeTrie(), função 494, 505
NovoNoTrieMachado(), função 505
NumeroDeChavesB(), função 349
NumeroDeChavesTrieMachado(), função 508
NumeroDeFilhosDeNoDeTrie(), função 512
NumeroDeNosB(), função 348
NumeroDeRegistros(), função 304

O

ObtemApontador(), função 113, 438, 658
ObtemBitsDeChave(), função 439
ObtemCodigosCanonicosHuff(), função 562
OBTEM CÓDIGOS DE HUFFMAN, algoritmo 553
ObtemElementoIdx(), função 147, 160
ObtemLSBs(), função 704
ObtemMinimoHeap(), função 546
ObtemMinimoIM_Arr(), função 661
ObtemMSBs(), função 705
ObtemTamanhosHuff(), função 561
ObtemTokens(), função 503
OCUPADO, constante de enumeração 382, 394
 one-at-a-time, função de dispersão 713–714
 operação de entrada/saída 69–71
 de entrada 69
 de saída 69
 sequência de eventos em 69
 operador de bits 700
 ordenação 580. *V. também* ordenação em memória principal; *V. também* ordenação em memória secundária
 adaptativa 582
 aplicações de 583–584
 da bolha. *V. BUBBLESORT*, algoritmo
 chave de 580
 por comparação 582
 por distribuição 582
 com espaço adicional 581
 estado de 583
 estável 580, 581
 externa 581, 650. *V. também* ordenação em memória secundária
 in loco 581
 por inserção. *V. INSERTIONSORT*, algoritmo
 instável 581
 interna 581. *V. também* ordenação em memória principal
 inversão em 583
 limite para algoritmo baseado em comparação de 621–624
 de lista encadeada 603–604
 de lista simplesmente encadeada 632–633

em memória principal. *V. ordenação em memória principal*
 em memória secundária. *V. ordenação em memória secundária*
 método de 582–583
 offline/online 582
 de ponteiros 633–634
 de propósito específico 582
 de propósito geral 582
 por seleção. *V. SELECTIONSORT*, algoritmo
 ordenação em memória principal
 aplicações de 583–584
 por base. *V. RADIXSORT*, algoritmo
 por borbulhamento. *V. BUBBLESORT*, algoritmo
 com coletor. *V. BUCKETSORT*, algoritmo
 por contagem. *V. COUNTINGSORT*, algoritmo
 com custo temporal linear 610–621
 com custo temporal linear logarítmico 592–610
 com custo temporal quadrático 584–591
 com heap. *V. HEAPSORT*, algoritmo
 por inserção. *V. INSERTIONSORT*, algoritmo
 por intercalação. *V. MERGESORT*, algoritmo
 usando **QUICKSORT**. *V. QUICKSORT*, algoritmo
 por seleção direta. *V. SELECTIONSORT*, algoritmo
 ordenação em memória secundária
 por indexação 684–687
 por intercalação binária 651–659
 por intercalação múltipla 662–673
 limite inferior para 673–674
 tamanho de buffer em 672
 teste de 687
OrdenaHeap(), função 546, 609, 661
OrdenaHeapIM_Arr(), função 661
OrdenaListaSE(), função 632
OrdenaPonteiros(), função 634
OrdenaTabelaIdx(), função 157
 organização de arquivo e bloco de disco magnético 65
 organizador prévio li
 otimização dinâmica 514

P

padrão de acesso sequencial/direto 64–65
 padrão de referência sequencial 77
 padrão (string) 452
 página 81, 296. *V. também* bloco: de disco magnético
PAI, macro 544
 par chave/valor 140
 Pareto, princípio de 77
PARTIÇÃO DE QUICKSORT, algoritmo 592
PEDIDO, constante de enumeração 567
 pen drive 66
 percolação de heap
 ascendente 540
 descendente 541
PERIODO, constante 567
 permutação 622
perorr(), função de biblioteca 102

persistência de dados 328
PilhaVazia(), função 328
 piso de chave 142, 187
PodeDividirColetorDext(), função 442
 política de substituição de cache
 aleatória 75
 FIFO 75
 MRU 76
 ponte de entrada e saída 67
 ponto bitônico 636
PontoBitonico(), função 636
 portabilidade
 de arquivo 117–118
 de **fseek()**, função de biblioteca 113
 do tipo **long int** 117
POSICAO_NULA, constante 299
 posicionamento em arquivo 111–115
 potencial, método de análise amortizada de 271, 273–274, 276–277, 280, 281–285. *V. também* análise amortizada
PREC0, constante 567
 preenchimento de estrutura 298–300
 pré-processamento de padrão 458–459, 464–466, 473–474, 484
PRIMARIO, constante de enumeração 422
PRIMEIRO_DIGITO_RADIX, constante 619
PRIMO, constante 483, 522
 princípio
 de localidade 77
 de Pareto 77
 problema de seleção 583
ProcessaChegada(), função 571
 processamento de arquivo 94
 por acesso direto 106, 111–115
 por acesso sequencial 106, 107–115
 por bloco 106, 109–111
 por byte 106, 107–111
 por caractere 106, 107–111
 formatado 106
 com leitura sequencial 110–111
 por linha 106, 108–109
ProcessaPedido(), função 571
ProcessaSaida(), função 572
ProfLocalMaxDext(), função 447
 profundidade. *V. também* altura
 de árvore AVL 232
 de árvore B 349–350
 de diretório de dispersão extensível 428
 global de dispersão extensível 428
 local de dispersão extensível 428
 programa
 -cliente 143–194
 dirigido por entrada e saída 76
 dirigido por processamento 76
 programação dinâmica 514
 PROM 57. *V. também* memória

Q

qsort(), função de biblioteca 157, 561, 626, 628, 652, 686
 quebra de linha 94
 em arquivo de texto 94
 remoção em string de 108
Quick1(), função 595
Quick2(), função 596
Quick3(), função 596
QuickSort1(), função 596
QUICKSORT, algoritmo 592–595
 análise de 596–599
 implementação de 595–596
 mediana de três em 594, 596
 partição de 592
 pivô aleatório de 596
 pivô de 592
 ponto de corte de 594, 596

R

"**r+b**", modo de abertura de arquivo 98, 115
 "**r**", modo de abertura de arquivo 98
 "**r+**", modo de abertura de arquivo 98, 115
RADIXSORT, algoritmo 617–619
 análise de 621
 base em 618
 implementação de 619–621
 representação posicional em 617
RadixSort(), função 620
 RAM 56–57. *V. também* memória
rand(), função de biblioteca 175, 364, 569
RAND_MAX, constante de biblioteca 569
 "**rb**", modo de abertura de arquivo 98
realloc(), função de biblioteca 277, 388
 rebalanceamento de árvore binária de busca. *V. desbalanceamento de árvore binária de busca*
 rebatismo de arquivo 105
 recomendação sobre
 arquivo aberto para atualização 114
 constante simbólica, uso de 719
 EOF, uso de 101
 estilo de programação 719–720
 fechamento de arquivo 100
 feof(), uso de 100, 101
 ferror(), uso de 101
 função de processamento de arquivo, uso de 117
 identificador, escrita de 719
 leitura além de final de arquivo, teste de 100, 101
 processamento de arquivo 96, 101
 rewind(), uso de 116
ReconstróiArvoreHuff(), função 564
 recorrência. *V. relação de recorrência*
 recuperação de informação 141
RedimensionaCuco(), função 397
RedimensionaHeap(), função 545

- redimensionamento de dispersão 386–388
 - análise de 388
 - fator de carga em 386
 - implementação de 387–388
- RedimensionaTabDEA()**, função 387
- REDIMENSIONATABELACUCO**, algoritmo 393
- redução de alfabeto 499
- ReduzHeapIM_Arr()**, função 661
- register**, palavra-chave 83
- registrador, uso de 83–84
- registro 580
 - de arquivo. *V.* registro de arquivo
 - de ativação 598
 - conceito de 140
 - preenchimento de estrutura e 298–300
- registro de arquivo 111
 - alteração de 115
 - atualização de 121–124
 - conversão de texto para binário de 124–125
 - índice de 118
 - inserção/remoção de 115
 - preenchimento de estrutura e 298–300
 - tamanho fixo, de 115
- regra 80/20 77
- relação de recorrência
 - de árvore AVL 232
 - de busca binária 163–164
 - de distância de edição 519
 - de lista com salto 179
 - de **MERGE SORT** 604
 - de ordenação em memória secundária 673
 - de **QUICK SORT** 596–597
- remoção
 - de arquivo 99–100, 105
 - em árvore afunilada 243–244
 - em árvore AVL 225–228
 - em árvore B 313–319
 - em árvore B+ 336–342
 - em árvore binária ordinária de busca 201–203
 - em árvore multidirecional descendente de busca 293–294, 306–309
 - em dispersão com encadeamento 372
 - em dispersão com endereçamento aberto 381
 - em dispersão cuco 393–394
 - em dispersão estática em memória secundária 422
 - em dispersão extensível em memória secundária 432–433
 - em heap binário 541–543
 - de indicação de erro em arquivo 101
 - em lista com saltos 169–170
 - de quebra de linha em string 108–109
 - de registro de arquivo 115
 - em tabela de busca indexada 149–150
 - em trie 490–492
- RemoveArvoreBB()**, função 207
- RemoveAVL()**, função 230
- RemoveChaveB()**, função 326
- RemoveChaveMultiMS()**, função 306
- RemoveCuco()**, função 398
- RemoveDEA()**, função 384
- RemoveDE()**, função 375
- RemoveDEst()**, função 426
- RemoveDExt()**, função 444
- REMOVEEMÁRVOREAFUNILADA**, algoritmo 243
- REMOVEEMÁRVOREAVL**, algoritmo 228
- REMOVEEMÁRVOREB**, algoritmo 313
- REMOVEEMÁRVOREB+**, algoritmo 339
- REMOVEEMÁRVOREBINÁRIADEBUSCA**, algoritmo 203
- REMOVEEMÁRVOREMULTIDIRECIONALDESCENDENTE**, algoritmo 293
- REMOVEEMDISPERSÃOCOMENCADEAMENTO**, algoritmo 372
- REMOVEEMDISPERSÃOCOMENDEREÇAMENTOABERTO**, algoritmo 381
- REMOVEEMDISPERSÃOCUCO**, algoritmo 393
- RemoveEmFolhaB()**, função 328
- REMOVEEMFOLHADEÁRVOREB**, algoritmo 314
- REMOVEEMHEAPASCENDENTE**, algoritmo 542
- REMOVEEMLISTACOMALTOS**, algoritmo 169
- REMOVEEMTABELADEDISPERSÃOESTÁTICA**, algoritmo 422
- REMOVEEMTABELADEDISPERSÃOEXTENSÍVEL**, algoritmo 433
- REMOVEEMTRIE**, algoritmo 492
- RemoveEmTrieAux()**, função 496, 507
- RemoveEmTrie()**, função 496, 507
- remove()**, função de biblioteca 105
- RemoveListaComSaltos()**, função 177
- RemoveListaSE()**, função 375
- RemoveMinHeap()**, função 545, 570
- RemoveMinHeapHuff()**, função 558
- RemoveNoArvoreFunil()**, função 247
- REMOVEDO**, constante de enumeração 382
- rename()**, função de biblioteca 105
- representação binária de valor do tipo **int** 707
- RepresentacaoBinaria()**, função 707
- representação posicional 617
- resolução de colisão 362
 - em dispersão com encadeamento 371
 - em dispersão com endereçamento aberto 377–380
 - em dispersão cuco 389
 - sondagem e 377–380
 - em tabela de dispersão estática 420–422
 - em tabela de dispersão extensível 427–431
- Ressurreição (romance) 697
- resto de divisão 366–367
- ResultadoBuscaEmTrie()**, função 513
- rewind()**, função de biblioteca 101, 106, 112, 116, 117
- RK. *V.* Karp e Rabin, algoritmo de casamento de robustez 116–117
- RotacaoDireitaArvoreBB()**, função 215
- rotação em árvore binária de busca 213–216
 - direita 213
 - esquerda 214
 - pivô de 213
 - preservação de ordem em 214
 - raiz de 213
 - zag 235

rotação em árvore binária de busca (*continuação*)
 zag-zag 235–236
 zag-zig 236–237
 zig 235
 zig-zag 236–237
 zig-zig 235–236

RotacaoEsquerdaArvoreBB(), função 216

rotNoABB, rótulo de estrutura 204

rotNoArvoreHuff, rótulo de estrutura 555

rotNoAVL, rótulo de estrutura 229

rotNoListaBloom, rótulo de estrutura 406

rotNoLSE, rótulo de estrutura 153, 275, 373, 616, 619

rotNoLS, rótulo de estrutura 170

rotNoMulti, rótulo de estrutura 294

rotNoTrieMachado, rótulo de estrutura 505

rotNoTrie, rótulo de estrutura 493

rotTabelaIdx, rótulo de estrutura 145, 146

rótulo de estrutura. *Procure um rótulo de estrutura específico pelo nome dele*

"**rt**", modo de abertura de arquivo 98

"**r+t**", modo de abertura de arquivo 98

S

saída
 dispositivo de 68
 formatada 102–103. *V. também* escrita formatada
 meio de 102

SAIDA, constante de enumeração 567

SaltaLinhas(), função 119

salto de linha de arquivo de texto 118–120

SATA 69

SAX, função de dispersão 715

scanf(), função 104

SCSI 69

SEEK_CUR, constante de biblioteca 112

SEEK_END, constante de biblioteca 112, 113

SEEK_SET, constante de biblioteca 112, 113

seek time 61

SELECTIONSORT, algoritmo 586–587
 análise de 588
 implementação de 587

SelectionSort(), função 587

semifolha 289

separação de string em partes (tokens) 502–504

SeparaEm3Particoes(), função 635

SeriesEstaoVaziasMS(), função 671

setor de disco magnético
 acesso a 61
 área de dados de 59
 cabeçalho de 59
 código de correção de erro de 59
 ECC de 59
 espaçador de 59
 extrato de 59
 identificação de 59
 informação sobre sincronização de 59
 intervalo de 59
 setor geométrico vs. 60
 trailer de 59

shift-add-xor (SAX), função de dispersão 715–717

simulação de evento discreto 566–572

simulação discreta dirigida por eventos 548–551
 distribuição exponencial em 550
 evento de 548, 549
 fila de prioridade em 548
 laço principal de 550
 lanchonete HeapBurger (exemplo) 566–572
 simulação contínua vs. 549

sinal de interrupção 69

site deste livro: www.ulysseso.com/ed2 li

SomaArrayBi1(), função 79

SomaArrayBi2(), função 80

SomaArray(), função 77, 81

sondagem em tabela de dispersão 376. *V. também* dispersão com
 endereçamento aberto
 agrupamento primário em 378
 com dispersão dupla 362, 379–380
 função de 377
 função de dispersão primária em 377
 função de dispersão secundária em 377
 índice de 376
 linear 362, 377–378
 passo de 376
 quadrática 362, 378–379
 sequência de 376

splay tree. *V. árvore afunilada*

sprintf(), função de biblioteca 103

SRAM 56–57. *V. também* memória

srand(), função de biblioteca 175, 364, 569

SSD 65–66

stderr, variável de biblioteca (stream) 102

stdin, variável de biblioteca (stream) 102

<**stdio.h**>, cabeçalho de biblioteca 94

stdio, módulo de biblioteca 94, 104

<**stdlib.h**>, cabeçalho de biblioteca 157, 160, 162

stdout, variável de biblioteca (stream) 102

strchr(), função de biblioteca 109

strcmp(), função de biblioteca 147, 253, 634

strcpy(), função de biblioteca 124

stream 94–95
 binário 97
 buffer associado a 95
 de entrada 99
 implementação de 95
 indicador de erro em 95
 indicador de final de 95
 indicador de posição em 95
 padrão 102
 ponteiro para 95
 de saída 99

stderr 102

stdin 102

stdout 102

stream (*continuação*)
de texto 97

strings

busca em. *V.* casamento de strings
casamento de. *V.* casamento de strings
casamento léxico de 519–522
distância de edição de 517–519

strings (*continuação*)

fopen(), usados por 96
maior prefixo comum (MPC) a um conjunto de 510–513
maior subsequência comum (MSC) a 513–517
MPC (maior prefixo comum) de 510–513
MSC (maior subsequência comum) de 513–517
separação em partes (tokens) de 502–504
strpbrk(), função de biblioteca 504
strtod(), função de biblioteca 124
strtok(), função de biblioteca 124, 502–504
substituição de cache 75
SubstituiMinimoIM_Arr(), função 661
substring 452, 453
sucessor imediato, nó 210

T

tabela de busca 140. *V. também* estrutura de dados
chave de 140
dicionário e 140
de dispersão 360. *V. também* tabela de dispersão
elemento de 140
encadeada 153–156. *V. também* tabela de busca encadeada
indexada 145. *V. também* tabela de busca indexada
índice de 142
organização de 143
par chave/valor de 140
tabela de busca encadeada 153–156
análise de 155–156
busca sequencial com movimentação para início em 154–155
busca sequencial com transposição em 153–154
implementação de 153–155
tabela de busca indexada
análise de 151–152
busca binária em 159–161. *V. também* busca binária
busca por interpolação 161–162
busca sequencial em 145
criação de 150–151
implementação de 145–151
inserção em 148–149
inserção em ordem em 158
remoção em 149–150
tabela de busca sequencialmente encadeada. *V.* tabela de busca encadeada
tabela de destruição 275–277
tabela de dispersão 360
diretamente endereçável 361
com encadeamento 371–376. *V. também* dispersão com encadeamento

com endereçamento aberto 376–386. *V. também* dispersão com endereçamento aberto
tabela de maiores bordas 458–459
tabela de ordenação 580, 583
TabelaDeSaltosBMH(), função 475
tabela de saltos do algoritmo **BMH** 473. *V. também* Horspool, algoritmo de casamento (ou simplificação) de
tabela de símbolos 362
tabela encadeada 153
TabelaEstaEmOrdemInversa(), função 630
TabelaEstaOrdenada(), função 629
tabela extensível 427. *V. também* dispersão extensível em memória secundária
tabela indexada dinâmica 277
TabelaMauCaractereBM(), função 468
tabela ordenada 580, 583
TabelaSufixosBM(), função 469, 470, 471
TAD (tipo abstrato de dado) 145–151
tAluno, tipo 122, 124, 634, 685, 694
TAM_ALFABETO, constante 493
tamanho
de array usado com **fgets()** 108–109
de bloco de disco magnético 82
de bloco de memória 101, 109–110
de partição de arquivo 106, 111
de registro de arquivo 111
do tipo **long int** 117
TamanhoHeap(), função 547
TamanhoHeapHuff(), função 558
TAMANHO_INICIAL_HEAP, constante 544
TAM_CHAVE_RADIX, constante 619
TAM_MAIOR_PALAVRA, constante 512
TAM_MATR, constante 122
TAM_REG, constante 653
tArvoreAVL, tipo 229
tArvoreBB, tipo 204
tArvoreBM, tipo 344
tArvoreHuff, tipo 555
tArvoreMultiMS, tipo 298
tArvoreMulti, tipo 294
TB, macro 300, 343
tBucket, tipo 616
tByteFreq, tipo 555
tByte, tipo 555
tCabecalhoHuff, tipo 563
tCaraCoroa, tipo 175
tCEP_Ind, tipo 145, 696
tCEP, tipo 145, 696
tChaveIndice, tipo 294, 298, 343, 697
tChaveRadix, tipo 619
tChave, tipo 294, 295, 343, 697
TCI, macro 300, 343
TC, macro 343
tColetorCuco, tipo 394
tColetorDEA, tipo 382
tColetorDEst, tipo 422
tColetorDExt, tipo 437

- tCompara, tipo 253
- tData, tipo 365
- técnica de dispersão, análise de 399–400
- tElemArray, tipo 660
- tempo de execução, medição de 86
- teorema
 - 3.1 (pior caso de busca sequencial) 151
 - 3.2 (melhor caso de busca sequencial) 151
 - 3.3 (custo temporal média de busca sequencial) 151
 - 3.4 (pior caso de inserção em tabela indexada) 151
 - 3.5 (pior caso de remoção em tabela indexada) 152
 - 3.6 (custo temporal de inserção em tabela encadeada) 155
 - 3.7 (melhor caso de remoção em tabela encadeada) 156
 - 3.8 (pior caso e caso médio de remoção em tabela encadeada) 156
 - 3.9 (custo espacial de tabela encadeada) 156
 - 3.10 (número máximo de comparações de chaves em busca binária) 163
 - 3.11 (caso médio de busca binária) 164
 - 3.12 (custo temporal de inserção em tabela indexada ordenada) 165
 - 3.13 (número de comparações de chaves em busca por interpolação) 165
 - 3.14 (número máximo esperado de níveis de lista com saltos) 178
 - 3.15 (custo temporal esperado de busca em lista com saltos) 179
 - 3.16 (custo temporal esperado de inserção em lista com saltos) 179
 - 3.17 (custo temporal esperado de inserção em lista com saltos) 179
 - 3.18 (custo espacial esperado de lista com saltos) 179
 - 4.1 (árvore binária de busca e caminhamento em ordem infixada) 197
 - 4.2 (pior caso de altura de árvore binária ordinária de busca) 213
 - 4.3 (altura de árvore AVL) 232
 - 4.4 (pior caso de operação em árvore AVL) 232
 - 5.1 (custo temporal de sequência de operações em tabela de destruição) 276
 - 5.2 (custo temporal de sequência de acréscimos em tabela indexada dinâmica) 279
 - 5.3 (custo temporal amortizado de afunilamento) 282–284
 - 6.1 (altura de árvore B) 329–330
 - 6.2 (custo de transferência de operação em árvore B) 331
 - 6.3 (custo de transferência de operação em árvore B+) 344
 - 7.1 (número de nós visitados em busca bem-sucedida em dispersão com encadeamento) 375
 - 7.2 (número de nós visitados em busca malsucedida em dispersão com encadeamento com listas desordenadas) 375
 - 7.3 (número de nós visitados em busca malsucedida em dispersão com encadeamento com listas ordenadas) 375
 - 7.4 (custo temporal de inserção em tabela de dispersão com encadeamento com lista desordenada) 376
 - 7.5 (custo temporal de remoção em tabela de dispersão com encadeamento) 376
 - 9.1 (pior caso do algoritmo **FB**) 457
 - 9.2 (melhor caso do algoritmo **FB**) 457
 - 9.3 (pior caso do algoritmo **KMP**) 463
 - 9.4 (custo espacial do algoritmo **BMH**) 476
 - 9.5 (custo temporal de pré-processamento do algoritmo **BMH**) 476
 - 9.6 (melhor caso do algoritmo **KR**) 484
 - 9.7 (pior caso do algoritmo **KR**) 484
 - 9.8 (pior caso de busca em trie) 497
 - 9.9 (custo temporal de remoção de string em trie) 497
 - 9.10 (custos temporal e espacial de inserção de string em trie) 497
 - 10.1 (índices de pai e filhos em árvore binária completa) 538
 - 10.2 (pior caso de inserção em heap) 547
 - 10.3 (pior caso de remoção em heap) 547
 - 11.1 (pior caso do algoritmo **BUBBLESORT**) 585
 - 11.2 (melhor caso do algoritmo **BUBBLESORT**) 585–586
 - 11.3 (caso médio do algoritmo **BUBBLESORT**) 586
 - 11.4 (custo temporal do algoritmo **SELECTIONSORT**) 588
 - 11.5 (número máximo de trocas no algoritmo **SELECTIONSORT**) 588
 - 11.6 (pior caso do algoritmo **INSERTIONSORT**) 590
 - 11.7 (melhor caso do algoritmo **INSERTIONSORT**) 590
 - 11.8 (caso médio do algoritmo **INSERTIONSORT**) 591
 - 11.9 (número de comparações do algoritmo **QUICKSORT** no pior caso) 596–597
 - 11.10 (número de trocas do algoritmo **QUICKSORT** no pior caso) 597
 - 11.11 (melhor caso do algoritmo **QUICKSORT**) 597
 - 11.12 (caso médio do algoritmo **QUICKSORT** básico) 598
 - 11.13 (custo temporal esperado do algoritmo **QUICKSORT** aleatório) 598
 - 11.14 (custo espacial do algoritmo **QUICKSORT** no melhor caso) 598
 - 11.15 (custo espacial do algoritmo **QUICKSORT** no pior caso) 598
 - 11.16 (custo temporal do algoritmo **MERGESORT**) 604
 - 11.17 (custo espacial do algoritmo **MERGESORT**) 605
 - 11.18 (custo temporal de criação de heap) 609
 - 11.19 (custo temporal do algoritmo **HEAPSORT**) 609
 - 11.20 (custo temporal do algoritmo **COUNTINGSORT**) 613–614
 - 11.21 (custo espacial do algoritmo **COUNTINGSORT**) 614
 - 11.22 (melhor caso do algoritmo **BUCKETSORT**) 617
 - 11.23 (custo espacial do algoritmo **BUCKETSORT**) 617
 - 11.24 (custo temporal do algoritmo **RADIXSORT**) 621
 - 11.25 (custo espacial do algoritmo **RADIXSORT**) 621
 - 11.26 (comparações feitas por um algoritmo de ordenação baseado em comparações) 623
 - 11.27 (limite inferior de algoritmo baseado em comparações) 623
 - 12.1 (custo de transferência do algoritmo de intercalação binária) 659

- teorema (*continuação*)
- 12.2 (custo temporal de intercalação múltipla de arrays) 662
 - 12.3 (custo de transferência de intercalação múltipla) 672
 - 12.4 (limite inferior para algoritmo de ordenação externa) 673
- TestaDispersao()**, função 403
- TestaMetodo()**, função 628
- TestaOrdenacaoMS()**, função 687
- teste de função de dispersão 370
- teste de unicidade 583
- teto de chave 142
- tEvento**, tipo 567
- tFComparaHuff**, tipo 555
- tFCompara**, tipo 544, 660, 669
- tFDispersaoCuco**, tipo 394
- tFDispersaoDEA**, tipo 382
- tFDispersao**, tipo 373, 406
- tFiltroBloomPtr**, tipo 406
- tFiltroBloom**, tipo 406
- TF**, macro 343
- tFOrdena**, tipo 628
- TG**, macro 343
- tHeapHuff**, tipo 555
- tHeapIM_Arr**, tipo 660
- tHeap**, tipo 544
- TI**, macro 300, 343
- time()**, função de biblioteca 86, 569
- <time.h>**, cabeçalho de biblioteca 86
- tipo. *Procure um tipo específico pelo nome dele*
- tipo abstrato de dado (TAD) 145–151
- tLanchonete**, tipo 567
- tListaBloom**, tipo 406
- tListaComSaltos**, tipo 171
- tListaNosAtivos**, lista 679
- tListaSE**, tipo 153, 275, 373, 619
- tMatricula**, tipo 694
- tmpfile()**, função de biblioteca 104
- tmpnam()**, função de biblioteca 104
- tNoArvoreBB**, tipo 204
- tNoArvoreHuff**, tipo 555
- tNoAVL**, tipo 229
- tNoBM**, tipo 344
- tNoCaminhoB**, tipo 319
- tNoFolha**, tipo 344
- tNoHeapHuff**, tipo 555
- tNoHeapIM_Arr**, tipo 660
- tNoHeap**, tipo 544, 567
- tNoInterno**, tipo 344
- tNoListaBloom**, tipo 406
- tNoListaCanHuff**, tipo 559
- tNoListaComSalto**, tipo 170
- tNoListaSE**, tipo 153, 275, 373, 616, 619, 632
- tNome**, tipo 694
- tNoMultiMS**, tipo 298
- tNoMulti**, tipo 294
- tNoPos**, tipo 679
- tNoTrieMachado**, tipo 505
- tNoTrie**, tipo 493
- token 502
- tOperacao**, tipo 252
- touch (comando do sistema operacional Unix) 82
- tPadraoLexico**, tipo 520
- tParChaveIndice**, tipo 685
- transferência de disco 288
- tratamento de exceção
- em abertura de arquivo 96
 - em árvore multidirecional de busca 300–301
 - em movimentação de apontador de arquivo 113
 - em processamento de arquivo 116–117
 - em tabela de busca 143
- tRegistroCEP**, tipo 695
- tRegistroMEC**, tipo 422, 437, 653, 669, 697
- tRegistro**, tipo 144
- trie 486–492
- altura de 499
 - análise de 497–499
 - aplicações de 487
 - busca em 488–489
 - compactada 500
 - filho de nó de 487
 - implementação de 492–497
 - implementações alternativas de 499–500
 - inserção em 490–492
 - nó final de 487, 488
 - nó folha de 488
 - nó redundante de 500
 - outros tipos de árvores de busca vs. 498
 - padrão 487
 - PATRICIA 500
 - remoção em 490–492
- TrocaGenerica()**, função 546, 584
- tStatusCuco**, tipo 394
- tStatusDEA**, tipo 382
- tTabelaCuco**, tipo 394
- tTabelaDEA**, tipo 382
- tTabelaDE**, tipo 373
- tTabelaDEExt**, tipo 437
- tTabelaIdx**, tipo 145
- tTipoDeColetorDEst**, tipo 422
- tTipoDeEvento**, tipo 567
- tTipoDoNo**, tipo 343
- TT**, macro 343
- tTrieMachado**, tipo 505
- tTrie**, tipo 493
- Tudor.bin**, arquivo de dados 124, 694
- Tudor.txt**, arquivo de dados 114, 118, 693–694

U

- UCHAR_MAX**, constante de biblioteca 560, 616
- ungetc()**, função de biblioteca 101, 105, 120
- unidade central de processamento 67
- unidade de transferência 64

Unix, sistema operacional
 conceito de arquivo no 94
 portabilidade de stream no 95
unsigned char, tipo 562, 616
USB, controlador/barramento de 68

V

ValidaPalavra(), função 510
valor de dispersão 360
valor inicial de dispersão 369
variável, nomenclatura de 719
VAZIO, constante de enumeração 382, 394

W

"**w+b**", modo de abertura de arquivo 98, 115
"**w+**", modo de abertura de arquivo 98, 115
"**w+t**", modo de abertura de arquivo 98, 115
"**wb**", modo de abertura de arquivo 98, 121
Windows, sistema operacional Microsoft
 portabilidade de stream no 95
"**w**", modo de abertura de arquivo 98
"**wt**", modo de abertura de arquivo 98
www.ulysseso.com/ed2, site deste livro li

X

xor 366, 700, 708–712