Lista de exercícios 01

Os seguintes exercícios são para aprofundar no estudo da matéria. Nos casos em que fala Exercícios e problemas (com números), se referem aos exercícios/problemas da 3^{ra} edição do livro de texto principal (T. Cormen, C. Leiserson, R. Rivest, C. Stein. "Introduction to Algorithms").

Técnicas e escrita de demonstração

1. Exercícios: 2.3-3

2. Encontre a uma fórmula fechada para a seguinte forma soma e demostre-a:

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2^n}.$$

3. A razão ϕ áurea e seu conjugado $\hat{\phi}$ são as raízes da equação

$$x^2 = x + 1$$

Mostre por indução que o i-ésimo número de Fibonacci F_i satisfaz a igualdade

$$F_= \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}.$$

4. Considere a sequência $1, 1, 3, 5, 11, ...$ dada pela seguinte expressão:

$$a_n = \begin{cases} 1 & \text{se } n = 1, 2, \\ a_{n-1} + 2a_{n-2} & \text{se } n \geq 3. \end{cases}$$

Mostre por indução que a_n satisfaz

$$a_n = \frac{2^n - (-1)^n}{3}.$$

5. Mostre que as regiões formadas por n círculos no plano podem ser coloridas com duas cores de forma que que quaisquer regiões vizinhas tenham cores distintas.

6. O princípio das caixas de pombo (na sua forma mais simples) afirma o seguinte: se $n+1$ bolas ($n \geq 1$) são colocada dentro de n caixas, então pelo menos uma caixa conterá mais de uma bola. Demonstre esse afirmação por indução.

7. Mostre por indução que, dado um inteiro em sua representação decimal, ele é divisível por três se e somente se a soma de seus dígitos é divisível por três. (Dica: é mais fácil provar um resultado

mais forte em que o resto da divisão do número por três é igual ao resto da divisão da soma de seus dígitos por três.)

8. Encontre uma expressão para a soma da i-ésima linha do seguinte triângulo, que é chamado de triângulo de Pascal, e prove que sua afirmação está correta usando indução. Os lados do triângulo são 1s e cada um dos outros itens tem valor igual à soma dos dois itens imediatamente acima.

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

9. Considere a sequência 1, 2, 3, 4, 5, 10, 20, 40, ..., que começa como uma progressão aritmética, mas após os cinco primeiros termos é progressão geométrica. Mostre que todo inteiro positivo pode ser escrito como a soma de números *distintos* dessa sequência.

Notação assintótica

10. Exercícios: 3.1-1, 3.1-2, 3.1-3, 3.1-4, 3.1-5, 3.1-7, 3.2-2, 3.2-3

11. Em cada uma das situações abaixo, indique se $f = O(g)$, ou se $f = \Omega(g)$, ou ambos (quando $f = \Theta(g)$). [Justifique formalmente as respostas dos itens (g), (n) e (q). Dica: para (q), suponha que k é constante e compare o quanto cada função cresce para de n para $n + 1$.]

	$f(n)$	$g(n)$
(a)	$n - 100$	$n - 200$
(b)	$n^{1/2}$	$n^{2/3}$
(c)	$100n + \log n$	$n + (\log n)^2$
(d)	$n \log n$	$10n \log 10n$
(e)	$\log 2n$	$\log 3n$
(f)	$10 \log n$	$\log(n^2)$
(g)	$n^{1.01}$	$n \log^2 n$
(h)	$n^2 / \log n$	$n(\log n)^2$
(i)	$n^{0.1}$	$(\log n)^{10}$
(j)	$(\log n)^{\log n}$	$n / \log n$
(k)	$\sqrt{n}$	$(\log n)^3$
(l)	$n^{1/2}$	$5^{\log_2 n}$
(m)	$n2^n$	3^n
(n)	2^n	2^{n+1}
(o)	$n!$	2^n
(p)	$(\log n)^{\log n}$	$2^{(\log_2 n)^2}$
(q)	$\sum_{i=1}^{n} i^k$	n^{k+1}

Recorrências

12. Exercícios: 4.3-1, 4.3-2, 4.3-3, 4.3-7, 4.3-9, 4.4-1, 4.4-6, 4.4-8, 4.4-9, 4.5-1, 4.5-4,

13. Encontre a solução da seguinte relação de recorrência. É suficiente encontrar o comportamento assintótico de $T(n)$. Você deve dar argumentos convincentes de que a função $f(n)$ que você

encontrou satisfaz $f(n) = \Theta(T(n))$.

$$T(n) = 2T\left(\left\lfloor \frac{n}{\log n} \right\rfloor\right) + 3n, \quad (n > 2), \quad T(1) = 1, \quad T(2) = 2.$$

(Dica: compare essa recorrência com alguma outra recorrência mais fácil de resolver)

14. Os números de Fibonacci $F(n)$ podem ser estendidos para valores negativos de n usando a mesma definição: $F(n+2) = F(n+1) + F(n)$, e $F(1) = 1$ e $F(0) = 0$. Assim, temos $F(-1) = 1$, $F(-2) = -1$ e assim por diante. Seja $G(n) = F(-n)$. Escreva uma relação de recorrência para $G(n)$ e sugira como resolvê-la. Prove que $G(n) = (-1)^{n+1}F(n)$.

Invariantes de laço e demonstração de correção

15. Exercícios: 2.1-3,

16. Problemas: 2-2, 2-3

17. Modifique o algoritmo Converte_Binário de tal forma que ele converta um número dado em base 6 para um número binário. A entrada é um vetor de dígitos na base 6 e a saída é um vetor de bits. Mostre a correção de seu algoritmo utilizando uma invariante de laço.

18. Seja $f(x)$ uma função real contínua e suponha que ela tem uma ou mais raízes entre a, b (uma raiz é um número $r \in (a, b)$ com $f(r) = 0$). Dado $\varepsilon > 0$, uma ε-aproximação de uma raiz, é um número $x \in (a, b)$ tal que $x \in (r - \varepsilon, r + \varepsilon)$ em que r é uma raiz. O método de aproximação da raiz baseado em busca binária executa é descrito no algoritmo a seguir:

```
   input  : a, b, ε
   output : ε-aproximação de uma raiz
 1 begin
 2 |  while b-a > ε do
 3 |  |  if f((a+b)/2) ≤ 0 then
 4 |  |  |  a ← (a+b)/2
 5 |  |  end
 6 |  |  else
 7 |  |  |  b ← (a+b)/2
 8 |  |  end
 9 |  end
10 |  return (a+b)/2
11 end
```

Algorithm 1: Bisect(a, b, ε)

(a) Analise o tempo de execução desse algoritmo em termos $b - a$ e ε.

(b) Dê condições (suficientes) sobre o entrada para que o algoritmo termine com uma resposta correta. Depois demonstre que o algoritmo está correto (quando dada uma entrada válida).

19. Considere o algoritmo a seguir:

```
   input  : a, b
   output : máximo divisor comum de a e b
 1 begin
 2 |  r ← a mod b
 3 |  while r ≠ 0 do
 4 |  |  a ← b
 5 |  |  b ← r
 6 |  |  r ← a mod b
 7 |  end
 8 |  return b
 9 end
```

Você deverá mostrar formalmente que o algoritmo está correto utilizando uma invariante de laço.

(a) Escreva uma invariante de laço adequada para o algoritmo.

(b) Demonstre a invariante de laço.

(c) Demonstre que o algoritmo está correto utilizando a afirmação acima.

Projeto de algoritmos recursivos e divisão e conquista

Projeto de algoritmos recursivos

20. O quebra-cabeça das torres de Hanoi é um exemplo de um problem não trival que tem uma solução simples recursiva. Há n discos colocado em uma estaca em ordem decrescente de tamanho. Há duas estacas livres. O objetivo do quebra-cabeças é mover todos os discos, um por vez, da primeira estaca até outra estaca da seguinte maneira. Discos são movidos do topo de uma estaca para o topo de outra. Um disco só pode ser movido se for menor do que todos os outros discos na estaca de destino. Em outras palavas, a ordenação dos discos em ordem decrescente deve ser mantida em todos os momentos. O objetivo é mover todos os discos com o menor número de movimentos.

(a) Projete um algoritmo (por indução) para encontrar uma sequência mínima de movimentos que resolve as torres de Hanoi para o problema com n discos.

(b) Quantos movimentos são realizados pelo seu algoritmo? Construa uma relação de recorrência para o número de movimentos e resolva-a (exatamente).

(c) Mostre que o número de movimentos da parte (b) é ótimo, i.e., mostre que não pode existir outro algoritmo que realiza menos movimentos.

21. O seguinte é uma variação do problema das torres de Hanoi. Não assumimos mais que os discos estão em uma única estaca. Eles podem estar distribuídos entre as três estacas, desde que estejam ordenados em cada uma. O propósito dessa variante continua sendo mover todos os discos para uma estaca especificada, com as mesmas restrições do problema original, com tão poucos movimentos quanto possível. Projete um algoritmo para encontrar uma menor sequência de movimentos para essa versão das torres de Hanoi para n discos.

22. Seja $x_1, x_2, \ldots, x_n$ uma sequência de números reais (não necessariamente positivos). Projete um algoritmo $O(n)$ para encontrar a subsequência $x_i, x_{i+1}, \ldots, x_j$ (de elementos consecutivos) de tal forma que o produto dos números é máximo entre todas as subsequências consecutivas. O produto de uma subsequência vazia é definido como 1.

23. Sobre os problemas a seguir, marque certo ou errado. Jusitique:

(a) É possível calcular o valor de um polinômio de quarto grau calculando o valor de um outro polinômio de terceiro grau relacionado.

(b) Existe algoritmo para calcular o valor de um polinômio $P(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ que faz menos de $3 + 2 + 1 = 6$ multiplicações.

(c) Em um algoritmo recursivo, fortalecer a hipótese da indução significa restringir o conjunto de instâncias que o algoritmo resolve.

(d) Em um algoritmo recursivo projetado por indução, a solução de um problema de tamanho $n+1$ deve ser obtida a partir da solução do problema de tamanho n.

(e) Defina como *impopular* em um grupo de pessoas, alguém que conhece todos no grupo, mas que não é conhecida por mais ninguém. Sempre há no máximo uma impopular no grupo.

(f) Defina como *celebridade* em um grupo de pessoas, alguém que é conhecida por todos no grupo, mas que não conhece ninguém. Sempre há pelo menos uma celebridades no grupo.

(g) Se A *conhece* B ou B *não conhece* A, então A não é uma celebridade.

(h) Na sequência $(2, -3, 1, 3, -2, 1, -1, 4, -4, 1, 2)$, uma subsequência **consecutiva** de soma máxima é $(1, 3, -2, 1, -1, 4)$.

(i) Na sequência $(2, -3, 1, 3, -2, 1, -1, 4, -4, 1, 2)$ uma subsequência **crescente** mais longa é $(2, 1, 3, 1, 4, 1, 2)$.

(j) Se quisermos encontrar os dois maiores números de um grupo de n elementos, devemos primeiro buscar o maior entre os n números e depois buscar o maior entre os $n-1$ números restantes, totalizando pelo menos $n-1+n-2$ comparações.

Divisão e conquista

24. Considere o problema de encontrar o máximo em um vetor de n números. Projete um algoritmo de divisão e conquista para o problema. Experimente duas abordagens: usar um subproblema de tamanho $n-1$ e dividir o subproblema em dois subproblemas de tamanho aproximadamente $n/2$. Qual a melhor abordagem?

25. Suponha que você está prestando consultoria em uma empresa de investimentos e gostaria de saber os melhores dias de comprar uma ação e vendê-la posteriormente, isso é, em que dia se deve comprar e em que dia se deve vendê-la para maximizar o lucro? Suponha que você tenha a estimativa dos preços de n dias. Escreva um algoritmo que execute em tempo $O(n \log n)$ baseando-se no princípio da divisão e conquista.

26. Suponha que você tenha um vetor A de n números naturais. Suponha que esse vetor tem a seguinte propriedade:

(a) Se o máximo ocorre em $A[p]$ para um índice p, então $A[1] \leq A[2] \leq \ldots A[p]$ e $A[p] \geq A[p+1] \geq \ldots A[n]$.

(b) Se $A[i] = A[j] = x$, então $A[k] = x$ sempre que $i \leq k \leq j$.

Seu objetivo é encontrar o valor máximo nesse vetor. Escreva o algoritmo mais eficiente que conseguir. Justifique a complexidade e a correção.

27. Leia a seção sobre o algoritmo de Strassen (seção 4.2).

Ordenação

Visão geral de algoritmos de ordenação

28. Dada uma matriz de números retangular, ordene cada linha da matriz. Em seguida, ordene cada coluna da matriz. Mostre que as linhas da matriz continuam ordenadas.

29. Em alguns casos, a entrada de um algoritmo de ordenação já está quase ordenada, o que significa que o número de elementos fora de ordem é pequeno. Descreva como os algoritmos de ordenação que você conhece se comportam com sequências quase ordenadas. Que algoritmo você usaria? (Você é encorajado a projetar o seu próprio algoritmo)

30. Suponha que queremos ordenar um conjunto de n registros armazenados em disco. Se n for relativamente pequeno, a melhor estratégia costuma ser carregar todos os dados na memória RAM, ordená-los e gravá-los de volta. Se n for muito grande, então estratégias diferentes precisam ser consideradas. Considere duas situações: quando os dados são comparados por uma rotina dada como caixa-preta; e quando se deve ordenar por identificadores únicos, que são números de 1 até n. Pesquise e disserte brevemente (200/400 palavras) sobre algoritmos de ordenação nessas situações. Escreva um texto coeso (i.e., escreva um texto contínuo evitando dar duas respostas independentes) e considere as operações necessárias para os algoritmos citados e as implicações da sua estratégia com relação ao tempo de latência e cópia em disco, quantidade de memória RAM disponível, etc.

31. A entrada são d sequências de elementos tais que cada sequência já está ordenada e há um total de n elementos. Projete um algoritmo que executa em tempo $O(n \log d)$ para juntar todas as sequências em uma única sequência ordenada.

32. Explique como você classificaria um baralho de cartas com a restrição de que as únicas operações permitidas são olhar os valores das duas primeiras cartas, inverter a ordem das duas cartas no topo e mover a carta mais acima para o fundo do baralho.

(a) Escreva um algoritmo para ordenar as cartas em ordem decrescente de valor e demonstre que o algoritmo está correto.

(b) Analise a complexidade do algoritmo e mostre que sua análise é justa, i.e., construa uma instância do problema e mostre que ela corresponde a um pior caso.

Fila de prioridade e Heapsort

33. Exercícios: 6.1-1, 6.1-2, 6.1-3, 6.1-4, 6.1-5, 6.1-6, 6.2-1, 6.2-2, 6.2-6, 6.3-2, 6.4-1, 6.4-3,

34. A entrada é um *heap* de tamanho n (em que o maior elemento está no topo), dado como um vetor, e um número real x. Projete um algoritmo para determinar se o k-ésimo maior elemento no heap é menor ou igual a x. No pior caso, seus algoritmo deve executar em tempo $O(k)$, independente do tamanho do heap. Você pode usar espaço de tamanho $O(k)$. (Note que você não tem que encontrar o k-ésimo maior elemento; você só precisa determinar sua relação com x.)

35. Uma árvore binária completa com n nós pode ser representada por um vetor B indexado por elementos $1, 2, \ldots, n$. Nesta questão, queremos representar uma árvore d-ária completa usando um vetor D, indexado de 1 a n. Uma árvore d-ária completa representada em um vetor tem as seguintes propriedades:

- cada nó que não é folha tem um número constante d de filhos;
- todas as folhas estão no nível mais abaixo;
- percorrer o vetor da esquerda para a direita é equivalente a percorrer a árvore em largura.

(a) Dado um índice j, qual o índice de D corresponde ao pai de j? E quais índices correspondem aos filhos de j? Demonstre isso.

(b) Escreva um algoritmo linear para, dado um vetor D com n elementos, criar um *maxheap* d-ário. Defina a propriedade de *heap* correspondente e demonstre a correção do seu algoritmo e sua complexidade.

Algoritmos aleatorizados e ordenação por particionamento

Revisão de conceitos básicos de probabilidade

36. Exercícios: 5.1-1, 5.2-1, 5.2-2, 5.2-3, 5.2-4, 5.3-2,

37. Use variáveis aleatórias indicadoras para resolver o seguinte problema, conhecido como problema do hat-check. Cada um dos n clientes entrega um chapéu a um verificador de chapéu em um restaurante. A pessoa que confere o chapéu devolve os chapéus aos clientes em uma ordem aleatória. Qual é o número esperado de clientes que recebem seu próprio chapéu?

Ordenação por particionamento

38. Exercícios: 7.1-1, 7.1-2, 7.2-2, 7.2-3, 7.2-4 (compare com a lista de ordenação em que os itens estavam quase ordenados), 7.2-5, 7.2-6(*), 7-4

39. Construa um exemplo para o qual *quicksort* realiza $\Omega(n^2)$ comparações quando o pivo é escolhido tomando a mediana do primeiro, do último e do elemento do meio de uma sequência.

40. Um professor irá fazer um jantar de confraternização com os estudantes de sua sala. Como o restaurante é muito caro, apenas alguns estudantes sorteados receberão convite; os outros precisam pagar se quiserem ir. Um estudante decide pagar pelo jantar se, e apenas se, pelo menos outros dois estudantes, um mais velho e um mais novo, confirmarem antes (assim ele não fica chateado de ser o mais novo ou o mais velho na confraternização). Suponha que todos os estudantes sorteados decidam participar e que cada estudante que ainda não decidiu tem a mesma chance em um sorteio. A estratégia do professor para que todos participem é sortear um convite entre os indecisos até que todos decidam participar. Quantos convites o professor precisa distribuir? Você pode supor que todas as idades são distintas.

(a) Descreva a estratégia do professor na forma de um algoritmo aleatorizado que recebe as idades dos estudantes, $e_1, e_2, \ldots, e_n$ e devolve o conjunto de estudantes que receberão convites.

(b) Estime o número esperado de convites (assintoticamente) que serão sorteados.

Dicas: considere primeiro a situação em que um estudante decide pagar se, e apenas se, algum estudante mais velho tiver confirmado presença antes.

Cota inferior para ordenação e ordenação em tempo linear

41. Exercícios: 8.1-1, 8.1-2, 8.2-1, 8.2-4, 8.3-1, 8.3-3, 8.4-1, 8.4-2,

42. A entrada é um conjunto S com n número reais. Projete um algoritmo de tempo $O(n)$ para encontrar um número que *não* está no conjunto. Mostre que $\Omega(n)$ é um limitante inferiores no número de passos para esse problema.

43. Dado um vetor de inteiros $A[1..n]$, tal que, para todo i, $1 \leq i < n$, temos $|A[i] - A[i+1]| \leq 1$. Seja $A[1] = x$ e $A[n] = y$, tais que $x < y$. Projete um algoritmo de busca eficiente para encontra j tal que $A[j] = z$ para um valor z, $x \leq z \leq y$. Qual é o número de comparações com Z que seu algoritmo faz.

44. Mostre usando árvore de decisão que o algoritmo que você desenvolveu no exercício anterior é ótimo no pior caso (ou melhore seu algoritmo até que você possa provar que ele é ótimo).

45. O Sr. B. C. Dull acredita ter desenvolvido uma nova estrutura de dados para filas de prioridade que suporta as operações de Inserção, Máximo e Extrair-Máximo, todas com tempo de execução $O(1)$ no pior caso. Prove que ele está enganado.

46. Suponha que o vetor $A[1..n]$ tenha apenas números de $\{1, \ldots, n^2\}$, mas que no máximo $\log \log n$ desses números aparecem no vetor. Projete um algoritmo que ordene A em tempo substancialmente menor que $O(n \log n)$. Calcule a complexidade e demonstre a correção do algoritmo.

Estatísticas de ordem

47. Exercícios: 9.1-1, 9.2-1, 9.2-2, 9.2-4, 9.3-1, 9.3-2, 9.3-5, 9.3-7, 9.3-9,

48. Suponha que usamos RandomizedSelect para selecionar o menor elemento de um vetor $\langle 3; 2; 9; 0; 7; 5; 4; 8; 6; 1 \rangle$. Descreva uma sequência de partições que resulte em uma execução de pior caso de RandomizedSelect.

49. Problemas: 9-1

50. Projete um algoritmo de divisão e conquista para encontra o menor e o maior elementos de um conjunto. O algoritmo deve usar no máximo $3n/2$ comparações (para $n = 2^k$). Você pode apontar a razão desse algoritmo requerer menos que $2n - 3$ comparações do algoritmo trivial?

51. A entrada é um conjunto S contendo n números reais e um número real x.

(a) Projete um algoritmo para determinar se há dois elementos de s cuja soma é exatamente x. O algoritmo deve executar em tempo $O(n \log n)$.

(b) Suponha agora que o conjunto S é dado de forma ordenada. Projete um algoritmo para resolver esse problema em tempo $O(n)$.

52. Diz-se que um vetor $A[1..n]$ tem um elemento majoritário se mais da metade de suas entradas forem iguais. Dado um vetor, a tarefa é projetar um algoritmo eficiente para determinar se o vetor possui um elemento majoritário e, em caso afirmativo, para encontrar esse elemento. Os elementos do vetor não são necessariamente de algum domínio ordenado como os inteiros e, portanto, não pode haver comparações da forma "$A[i] > A[j]$?". (Pense nos elementos da matriz como arquivos GIF, digamos). No entanto, você pode responder perguntas do tipo: "$A[i] = A[j]$?" em tempo constante.

(a) Mostre como resolver este problema em tempo $O(n \log n)$. Para isso, divida a matriz A em duas matrizes A_1 e A_2 com a metade do tamanho. Conhecer um elemento majoritário de A_1

e A_2 ajuda você a descobrir o elemento maioritário de A? Se for assim, então você pode usar a abordagem de divisão-e-conquista.

(b) Você pode dar um algoritmo de tempo linear? Aqui está outra abordagem de divisão e conquista:

- Pareie elementos de A e obtenha $n/2$ pares.
- Olhe para cada par: se os dois elementos forem diferentes, descarte os dois; se eles são os mesmos, mantenha apenas um deles

Formalize e mostre que, após este procedimento, há no máximo $n/2$ elementos restantes e que eles têm um elemento maioritário se e somente se A tiver. Depois mostre como resolver o problema em tempo linear.

53. Os k-ésimos quantis de um conjunto de elementos são os $k-1$ elementos que dividem o conjunto ordenado em k conjuntos de tamanhos quase iguais (a diferença de tamanho entre quaisquer dois conjuntos é no máximo um). Dê um algoritmo de tempo $O(n \log k)$ para listar os k-ésimos quantis de um conjunto.

Programação dinâmica

54. Exercícios: 15.2-1, 15.2-2, 15.2-3, 15.3-1, 15.3-2, 15.3-3, 15.3-4 (3ed), 15.3-5 (2ed), 15.4-1, 15.4-2, 15.4-4, 15.5-2,

55. Problemas: 15-5, 15-6 (3ed),

56. Relembre o problema da mochila: dados um inteiro K e n itens de pesos diferentes, $S = \{p_1, p_2, \ldots, p_n\}$, encontrar um subconjunto $S' \subseteq S$ cuja soma dos pesos é exatamente K, ou determine que tal conjunto não existe.

Na tabela seguinte, temos uma tabela de programação dinâmica para o problema da mochila parcialmente preenchida. O itens têm pesos $p_1 = 2$, $p_2 = 7$, $p_3 = 5$ e $p_4 = 2$. Preencha os dados que faltam para os itens p_3 e p_4. O símbolo **O** significa existe solução para o subproblema da mochila correspondente ao item, sem usar o item. O símbolo **I** significa que existe solução para a respectiva mochila, usando o respectivo item. O símbolo **-** significa que não existe solução para a respectiva mochila.

Tam. Mochila $\rightarrow$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$p_1 = 2$	O	-	I	-	-	-	-	-	-	-	-	-	-	-	-
$p_2 = 7$	O	-	O	-	-	-	-	I	-	I	-	-	-	-	-
$p_3 = 5$															
$p_4 = 2$															

57. Melhore o uso de espaço do algoritmo para o problema da mochila apresentado em sala. Há necessidade de usar uma matriz completa $n \times K$? Qual a complexidade de espaço do algoritmo melhorado?

58. O problema da mochila inteira é uma variante do problema da mochila binária. Nesse problema, cada item corresponde a um tipo e pode ser utilizado várias vezes, i.e., uma solução

pode não conter itens de um tipo, conter apenas um, ou conter diversas cópias do mesmo item. Resolva este problema por programação dinâmica.

59. Considere a seguinte versão bidimensional do problema da subsequência consecutiva máxima. Considere uma matriz M, de dimensões $m \times n$ com números inteiros (positivos ou negativos). O objetivo é encontrar uma submatriz consecutiva N cuja soma dos elementos é máxima. Uma submatriz consecutiva é exatamente isso que você está pensando.

60. Cortando uma string.

Uma certa linguagem de processamento de strings permite que uma programadora corte uma string em duas partes. Como esta operação copia toda string, ela custa n unidades de tempo para cortar uma string com n caracteres em duas partes. Suponha que uma programadora queira quebrar uma string em muitos pedaços. A ordem em que os cortes ocorrem pode afetar o tempo gasto. Por exemplo, suponha que a programadora deseja quebrar uma sequência de 20 caracteres após os caracteres 2, 8 e 10 (numerando os caracteres a partir de 1 e começando na extremidade esquerda). Se ela programar os cortes para ocorrerem na ordem da esquerda para a direita, o primeiro corte custará 20 unidades de tempo, o segundo 18 unidades de tempo (quebrando a string dos caracteres 3 a 20 no caractere 8) e o terceiro 12 unidades de tempo, totalizando 50 unidades de tempo. Se ela programa os cortes para ocorrerem na ordem da direita para a esquerda, no entanto, o primeiro custará 20 unidades de tempo, o segundo 10 unidades de tempo e o terceiro 8 unidades de tempo, totalizando 38 unidades de tempo. Em um outra ordem de cortes, ela poderia partir primeiro em 8 (custando 20), depois quebrar a parte esquerda em 2 (custando 8) e, finalmente, a parte direita em 10 (custando 12), para um custo total de 40.

Projete um algoritmo que, dados os números de caracteres após os quais se deve cortar a string, determina a sequencia de cortes de menor custo. Mais formalmente, dada uma string S com n caracteres e um vetor $L[1...m]$ contendo os pontos de interrupção, calcule o menor custo para uma sequência de intervalos, juntamente com uma sequência de intervalos que tem esse custo.

61. Roberto tem uma sorveteria e quer fazer o planejamento de reabastecimento do estoque para um certo período. Ele deseja reduzir os custos de frete e armazenamento. Roberto sempre sabe com antecedência o custo do frete para encomendar sorvete e a demanda de sorvete dos próximos n dias. O valor de uma encomenda em um dia não muda independentemente do número de potes enviados. Ele também sabe qual o custo de manter cada pote de sorvete por uma noite no freezer. O problema é planejar quantos potes de sorvete devem ser encomendados em cada dia.

(a) Formalize o problema: descreva a entrada, uma solução e defina a função-objetivo.

(b) Escreva um algoritmo de programação dinâmica que resolva o problema (i.e., encontra um planejamento de custo mínimo.)

Algoritmos Gulosos

62. Exercícios: 16.1-1, 16.1-2, 16.1-3, 16.1-4, 16.2-1, 16.2-2, 16.2-4, 16.2-5, 16.2-7, 16.3-2, 16.3-3, 16.3-4, 16.3-8

63. Problemas: 16-1, 16-4a,

64. Uma caixa d-dimensional com lados $(x_1, \ldots, x_d)$ cabe numa caixa $(y_1, \ldots, y_d)$ se existe uma

permutação π de $1, \ldots, d$ tal que

$$x_{\pi_1} < y_1, \ \cdots, \ x_{\pi_d} < y_d.$$

Dê um algoritmo eficiente para determinar se $(x_1, \ldots, x_d)$ cabe em $(y_1, \ldots, y_d)$. Prove que este algoritmo está correto.

65. São dados n livros, $1, 2, \ldots, n$ com pesos $p_1, p_2, \ldots, p_n$, respectivamente. Os pesos satisfazem a condição $0 < p_i < 1$, para $i = 1, \ldots, n$. Deseja-se acondicionar os livros em um número mínimo de envelopes satisfazendo as condições abaixo:

1. Cada envelope contém no máximo dois livros.
2. Em nenhum envelope o peso dos livros ultrapassa 1. Descreva um algoritmo com *número de comparações* $O(n \log n)$ que acha um acondicionamento ótimo de n livros dados. Demonstre que o acondicionamento encontrado por seu algoritmo é ótimo, i.e., ele usa o menor número possível de envelopes.

66. Seja $N = 2^k$ e $S = \{x_1, \ldots, x_s\}$, onde x_i é potência de 2, $x_i \leq N$ e $\sum_{i=1}^{s} x_i \geq N$. Então existe um conjunto $S' \subseteq S$ tal que $\sum_{x' \in S'} x' = N$. Prove este resultado e apresente um algoritmo para encontrar tal S'.

67. Uma empresa de esquadrias metálicas precisa de n_i vigas de tamanho 2^i, $i = 0, \ldots, k$. Mas a metalúrgica que vende as vigas para a empresa de esquadrias só vende vigas de tamanho M, M é um inteiro positivo e $2^i \leq M$, $i = 0, \ldots, k$. Assim, a empresa de esquadrias precisa comprar o menor número de vigas de tamanho M. Projete um algoritmo para resolver este problema de forma ótima, i.e., usando o menor número de vigas grandes. Prove que ele devolve a solução ótima. O valor dos n_i e M são dados. (Você pode usar o algoritmo da questão anterior como subrotina).

68. Suponha que em um problema de mochila 0-1, a ordem dos itens quando classificados por aumento de peso é a mesma ordem que quando classificados por valor decrescente. Dê um algoritmo eficiente para encontrar uma solução ótima para essa variante do problema de mochila e argumente que seu algoritmo está correto.

69. Generalize o algoritmo de Huffman para palavras de código ternárias (i.e., palavras de código usando os símbolos 0, 1 e 2) e prove que ele produz códigos ternários ótimos.

70. O cadeado de Alice, que é de combinação de n números como o da figura abaixo, enferrujou-se e ficou com o seguinte defeito: toda vez que gira um número, o número imediatamente acima gira junto. O seu objetivo é ajudar Alice a obter a combinação da sua senha pessoal: uma sequência de n zeros! Como o cadeado está enferrujado, deve-se girar o menor número de vezes possível. Os números do cadeado vão de 0 até 9 e podem ser girados tanto para a esquerda como para a direita. Por exemplo, se a combinação atual, de baixo para cima, for $\{5, 4, 2, 1\}$ e girarmos o número 4 *três vezes* para a esquerda, obteremos a combinação $\{5, 1, 9, 1\}$.

(a) Escreva um algoritmo linear que receba a combinação atual do cadeado de baixo para cima e instrua Alice a abrir o cadeado com o menor número de giros possível.

(b) Demonstre que o algoritmo está correto.