

PROJETO E ANÁLISE DE ALGORITMOS (INF 2926)

Lista 1

1. Considere as seguintes funções:

- (a) $10.n^\pi$
- (b) $\log n$
- (c) $\log(n^2)$
- (d) $0.005.n^{0.00001}$
- (e) $1000.(\log n)^2$
- (f) $30.n^3.\log n$
- (g) $50.n.\log^2 n$
- (h) $(\log n)^{\log n}$
- (i) $\frac{n}{\log n}$
- (j) $70.n$
- (k) $\log \log n$
- (l) $(1.5)^{(\log n)^2}$
- (m) $90.n^2.\log n + n^3.\log n$

- Coloque as funções acima em ordem de crescimento assintótico, i.e. valor quando $n \rightarrow \infty$.
- Utilize pelo menos três vezes cada um dos símbolos O , Ω , Θ , o , e ω para indicar a relação existente entre pares das funções acima (não vale recíprocos).

2. Encontre contra-exemplos para as proposições abaixo.

- (a) Se $f(n) = O(s(n))$ e $g(n) = O(r(n))$, então $f(n) - g(n) = O(s(n) - r(n))$
- (b) Se $f(n) = O(s(n))$ e $g(n) = O(r(n))$, então $f(n)/g(n) = O(s(n)/r(n))$

3. Considere o seguinte problema:

[ELE] Dados um conjunto de n inteiros distintos $X = \{x_1, \dots, x_n\}$ e pesos positivos $w(x_j)$, $j = 1, \dots, n$, associados aos elementos de X , e um inteiro positivo V tal que $0 \leq V \leq W = \sum_{j=1}^n w(x_j)$; encontrar o elemento x_k tal que $\sum_{x_j < x_k} w(x_j) < V$ e $w(x_k) + \sum_{x_j < x_k} w(x_j) \geq V$.

- (a) Projete um algoritmo $O(n \log n)$ para resolver esse problema.

- (b) Utilize divisão e conquista para projetar um algoritmo $O(n)$ para resolver esse problema.
- (c) Apresente detalhadamente a análise da complexidade do item anterior.
4. Sejam u e v dois números de n bits (considere que n é potência de 2). A multiplicação tradicional de u por v utiliza $O(n^2)$ operações. Um algoritmo baseado em divisão e conquista divide os números em duas partes iguais, calculando o produto como: $uv = (a2^{n/2} + b)(c2^{n/2} + d) = ac2^n + (ad + bc)2^{n/2} + bd$. As multiplicações ac , ad , bc e bd são feitas usando este mesmo algoritmo recursivamente.
- Escreva este algoritmo
 - Determine a complexidade
 - Qual é a complexidade se $ad + bc$ é calculado como $(a + b)(c + d) - ac - bd$?
5. Verdadeiro ou Falso. Justifique.
- (a) $(\log n)^{100} = O(n^\epsilon)$, $\forall \epsilon > 0$.
- (b) $2^{n+1} = O(2^n)$.
- (c) Se $g(n, m) = m \log_d n$ onde $d = \lceil m/n + 2 \rceil$ ($\lceil x \rceil$ é o menor inteiro maior que x), $m = O(n^2)$, então $g(n, m) = O(m^{1+\epsilon})$ $\forall \epsilon > 0$.
6. Determine uma forma fechada para cada uma das seguintes recorrências:
- (a) $T(1) = 1$;
 $T(2) = 6$;
 $T(n) = T(n - 2) + 3n + 4$, $\forall n \geq 3$.
- (b) $T(1) = 1$;
 $T(2) = 6$;
 $T(n) = 2.T(n - 2) + 3$, $\forall n \geq 3$.
- (c) $T(1) = 1$;
 $T(n) = \sum_{i=1}^{n-1} (T(i) + T(n - i)) + 1$, $\forall n \geq 2$.
7. Seja $S = \{x_1, x_2, \dots, x_n\}$ uma seqüência de números reais (não necessariamente positivos). Projete dois algoritmos de complexidade $O(n)$, um iterativo e um que utilize divisão e conquista (ambos podem ser recursivos!), que determine uma subseqüência $S' = \{x_i, x_{i+1}, \dots, x_j\}$ de elementos consecutivos da seqüência S tal que o produto dos números que compõem S' é máximo dentre todas as subseqüências consecutivas possíveis.
8. Considere uma *heap* com n inteiros e um valor x . A *heap* possui o seu maior elemento no topo e está organizada em um vetor (de n posições). Proponha um algoritmo para decidir se o valor x é maior ou igual ao k -ésimo maior elemento na *heap* ou não. O algoritmo deve executar em $O(k)$.
- Dica: Observe que você não precisa necessariamente encontrar o k -ésimo maior elemento da *heap*.
9. Dados um multiconjunto C contendo n números reais (não necessariamente distintos) e um número real x :

- (a) Projete um algoritmo de complexidade $O(n \log n)$ que determine se existem dois elementos em C cuja soma seja igual a x .
- (b) Considere agora que os elementos do conjunto C estão ordenados crescentemente. Projete um algoritmo de complexidade $O(n)$ para resolver o mesmo problema.
10. O departamento de transporte de uma cidade deseja averiguar se seus motoristas respeitam os conceitos pregados pela técnica de direção defensiva. De acordo com o departamento, todos os veículos deveriam manter entre si uma distância maior ou igual d , definida como *distância de segurança*. Com o objetivo de realizar uma pesquisa, várias câmeras foram instaladas pela cidade, captando as posições de inúmeros veículos. A posição de um veículo é definida por um par ordenado $(x, y) \in R^2$. Seu objetivo é projetar um algoritmo de complexidade $O(n \log n)$ que obtenha uma cena contendo as posições de n veículos captadas por uma câmera e a distância de segurança d atualmente estabelecida pelo departamento, informando ao final da execução se existem veículos que não respeitam a distância de segurança. Seu algoritmo deve responder apenas *sim* ou *não*. A distância entre dois veículos com posições (x_1, y_1) e (x_2, y_2) é dada por $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.
11. O problema da torre de Hanói generalizado consiste em mover n discos de diâmetros distintos, posicionados em um haste denominada *origem*, para uma haste denominada *destino* utilizando $m \geq 1$ hastes denominadas *de trabalho*. Inicialmente, todos os discos encontram-se empilhados na haste de origem em ordem decrescente de tamanho, de baixo para cima. As demais hastes de trabalho e destino encontram-se vazias. Durante o processo de transferência é permitida a movimentação de apenas um disco por vez. Considere ainda que nenhum disco pode ser posicionado acima de um disco com diâmetro menor que o seu. Projete um algoritmo que resolva o problema da torre de Hanói generalizado utilizando o menor número de movimentos possível. Seu algoritmo deve relatar a ordem e a quantidade de movimentos a serem realizados.
12. Seja uma sequência de n inteiros distintos $T = \{t_1, \dots, t_n\}$. Uma subsequência crescente de T é um subconjunto não necessariamente consecutivo de T que preserva a ordem em T .
- (a) Considere o problema de encontrar a subsequência (não necessariamente consecutiva) crescente de T com maior número de elementos.
 Defina $T(j)$ como a sequência $\{t_1, \dots, t_j\}$. Assuma que para um dado j , $j \leq n$, conhece-se as subsequências de tamanho k , $0 \leq k \leq j$ (k não necessariamente vai até j), cujo último elemento tem o menor valor possível.
 Projete o algoritmo resultante da obtenção destas subsequências cada vez um novo elemento de T é considerado (j é incrementado). Analise a complexidade do algoritmo obtido. O seu algoritmo deve ter complexidade $O(n^2)$.
- (b) Considere, agora, que para cada elemento de T é associado um custo positivo c_i , $i = 1, \dots, n$. Proponha um algoritmo para encontrar a subsequência crescente de T onde a soma dos custos dos elementos é maximizada.
 Dica: Qual seria o custo em que esse problema fica equivalente ao do item (a).
- (c) Mostre como os algoritmos obtidos acima funcionam na sequência: 3, 17, 9, 12, 35, 6, 27, 8, 21, 26, 23, 11, 19, 13, 15. Execute o algoritmo do item (b) (só até o sexto elemento) com os custos 1, 2, 3, 4, 5, 6 e também com os custos 15, 14, 13, 12, 11, 10.

- (d) Analise a complexidade do algoritmo em (b). Diga se ele tem complexidade polinomial ou exponencial. No caso da resposta ser exponencial, indique se a complexidade é pseudo-polinomial.
- (e) O problema do item (b) pode ter algoritmo polinomial? No caso afirmativo, apresente um algoritmo polinomial. No caso negativo justifique.
13. Sejam S_1 e S_2 conjuntos cujos elementos são números inteiros e onde $|S_1| = |S_2| = n$.
- (a) Escreva um algoritmo para determinar a união de S_1 e S_2 . Nenhum elemento deve aparecer mais de uma vez. Analise sua complexidade em função de n (melhor caso e pior caso).
- (b) Este algoritmo deve executar em $O(n \log n)$.
14. Seja uma função convexa $f(x)$ definida no intervalo $[0, U]$. Dado um $\epsilon > 0$, deseja-se encontrar x^0 de modo que o intervalo $[x^0 - \epsilon, x^0 + \epsilon]$ contenha x^* onde $f(x^*)$ é o menor valor de f no intervalo dado.
- (a) Suponha que, para um dado x e um dado U , o cálculo de $f(x)$ se faz em tempo constante. Projete um algoritmo de complexidade $O(\log(U/\epsilon))$.
Dica: Lembre que a função é convexa e, para iterativamente se aproximar do mínimo, determine o valor da função em 4 pontos. O que você pode concluir sobre onde está o mínimo de f ?
- (b) A complexidade deste algoritmo é polinomial? Justifique.
15. Considere que os n itens de uma busca arqueológica estão organizados nos andares de uma pirâmide de modo que o item de maior valor está no andar mais alto, os 2 itens de maior valor seguintes estão no andar abaixo, os 4 itens de valores menores no andar abaixo do que tem 2 itens, e assim por diante. Isto é, sendo v_1, v_2, \dots, v_n e que $v_1 \geq v_2 \geq v_3 \geq \dots \geq v_n$, o item 1 está no andar mais alto, 2 e 3 no seguinte, os itens 4, 5, 6 e 7 logo abaixo, e assim por diante. Responda:
- (a) Quantos andares tem a pirâmide que guarda os itens?
- (b) Utilizando um vetor para armazenar o valor do item de maior valor de cada andar da pirâmide, proponha um algoritmo $O(\log \log n)$ para determinar o andar em que se encontra um item dado.
16. Analise a eficiência do algoritmo abaixo, calculando o número de passos executados pelo mesmo em função de um número n fornecido como entrada. Considere que todas as operações básicas envolvidas (atribuição, operações lógicas, operações aritméticas, entrada/saída) possuem custo constante ($c = O(1)$).

```

Leia(n);
x <- 0;
Para i <- 1 até n faça
  Para j <- i+1 até n faça
    Para k <- 1 até j-i faça
      x  x + 1;
Imprima(x);

```

17. Perdido em uma terra muito distante, você se encontra em frente a um muro de comprimento infinito para os dois lados (esquerda e direita). Em meio a uma escuridão total, você carrega um lampião que lhe possibilita ver apenas a porção do muro que se encontra exatamente à sua frente (o campo de visão que o lampião lhe proporciona equivale exatamente ao tamanho de um passo seu). Existe uma porta no muro que você deseja atravessar. Supondo que a mesma esteja a n passos de sua posição inicial (não se sabe se à direita ou à esquerda), elabore um algoritmo para caminhar ao longo do muro que encontre a porta em $O(n)$ passos. Considere que n é um valor desconhecido (informação pertencente à instância). Considere que a ação composta por dar um passo e verificar a posição do muro correspondente custa $O(1)$.
18. “Pseudo ou não-pseudo? Eis a questão.”
- (a) Defina os conceitos de algoritmo polinomial e algoritmo pseudopolinomial.
 - (b) Forneça um exemplo de um algoritmo polinomial e outro de um algoritmo pseudopolinomial.
 - (c) Prove ou refute: Todo algoritmo polinomial é pseudopolinomial.
 - (d) Prove ou refute: Todo algoritmo pseudopolinomial é polinomial.
19. Era uma vez um rei que não conhecia algoritmos. Durante um fatídico verão, seu reino fora invadido por um dragão, obrigando os conselheiros da ordem omega-theta a se reunirem e decidirem por contratar o cavaleiro mais corajoso (e ambicioso) do reino que fez uma proposta de recompensa associada ao peso do animal. Considerando que o peso do dragão seja n quilos, a cobrança será feita em moedas de ouro, seguindo a regra de $k \cdot 2^{(n-k)}$ moedas de ouro adicionais para o k -ésimo quilo do dragão. Para exemplificar, um dragão com 4 quilos custaria ao reino $1 \cdot 2^3 + 2 \cdot 2^2 + 3 \cdot 2^1 + 4 \cdot 2^0 = 26$ moedas de ouro.
- Elabore um algoritmo POLINOMIAL (LINEAR!!!) que, dado um inteiro positivo n , correspondente ao peso do dragão, calcule a quantidade de moedas a serem pagas ao cavaleiro. O algoritmo (projetado para ser executado pelos conselheiros da corte) deve conter apenas as operações aritméticas de soma, subtração, multiplicação e divisão de inteiros (considere que qualquer uma dessas operações é realizada em tempo constante).
20. No que se refere ao conceito de esquema de codificação de uma instância:
- (a) Prove por indução que um número inteiro positivo n ao ser codificado na base $b \geq 2$ faz uso de $(\log_b n)$ caracteres.
 - (b) Prove por indução que um número inteiro positivo n ao ser codificado na base unária faz uso de (n) caracteres.
 - (c) Utilizando o resultados anteriores prove que, um algoritmo que recebe como entrada um número inteiro positivo n codificado em uma base $b \geq 2$ e que executa em n passos é um algoritmo de complexidade exponencial em função do tamanho da instância.