Projeto e Análise de Algoritmos
Lista de Exercícios I

Data de Entrega: 6/4/2025 (moodle)
Valor: 5 pontos

1) Seja o algoritmo abaixo que ordena um vetor A de entrada, no qual A.length é o número de elementos do vetor:

```
BUBBLE-SORT(A)
    for i = 1 to A.length-1
        for j = A.length downto i + 1
            if A[j] < A[j-1]
                exchange A[j] with A[j-1]
```

   a) Como o algoritmo funciona?
   b) Utilize invariantes de loops para provar que ele sempre termina com o vetor A ordenado (ou seja, ele está correto)
   c) Qual a função de complexidade para o número de comparações? E para o número de trocas?

2) Considere o algoitmo iterativo abaixo que encontra o maior e o menor elemento de um vetor A de tamanho n. Considere ainda que os n elementos estão distribuídos aleatoriamente no vetor. Responda:

```
MaxMin(int A[1..n]){
    max = A[1];
    min = A[1];
    for(i=2; i<=n; i++)
        if (A[i]>max) max = A[i];
        else if(A[i]<min) min = A[i];
    imprime(min,max)
}
```

a) Mostre que o algoritmo está correto utilizando invariantes de loops
b) Qual é a função de complexidade do número de comparações de elementos no melhor e pior caso?
c) Utilizando análise probabilística, compute o número de comparações de elementos do vetor que serão realizadas no caso médio.

3) Considere uma generalização do Problema do Casamento Estável em que certos pares hospital-estudante são explicitamente proibidos. Por exemplo, um estudante pode se recusar a aceitar uma vaga de um determinado hospital mesmo estando livre, ou um hospital pode não aceitar contratar, em hipótese alguma, um determinado estudante. Em outras palavras, há um conjunto H de n hospitais, um conjunto S de n estudantes, e um conjunto $F \subseteq H \times S$ de pares que representam casamentos que não são permitidos. Cada hospital h ordena todos os estudantes s tal que $(h,s) \notin F$, e cada estudante s ordena todos os hospitais h tal que $(h, s) \notin F$. Considerando este cenário, responda:

a) Quais são as condições de instabilidade? Note que neste caso, os casamentos gerados podem não ser perfeitos.
b) Adapte o Algoritmo de Gale Shapley para funcionar neste novo cenário
c) Prove que ele funciona, ou seja, que ele produz casamentos estáveis.

4) Encontre um limite assintótico firme para as equações de recorrência abaixo usando os métodos indicados:

a) $T(n) = T(n/2) + 1$ usando o método da expansão de termos.
b) $T(n) = T(n-1) + n^2$ usando o método da substituição.
c) $T(n) = 2T(n/2) + n^3$ usando o Teorema Mestre.
d) $T(n) = 4T(n/2) + n^2\sqrt{n}$ usando o Teorema Mestre.

5) Determine um limite superior assintótico para as funções abaixo (de preferência, o mais apertado possível):

a) $2n^3 + n^4 - 1$
b) $2^n + 5 \log n + n^2$
c) $\log_{10} n + \log_3 10$
d) $n + n \log n + \log n$.
e) $4^n + 2^n + n$

6) Utilizando as definições formais para as notações assintóticas, prove se são verdadeiras ou falsas as afirmativas abaixo:

a) $3n^3 + 2n^2 + n + 1 = O(n^3)$
a) $6n^3 + 5n^2 \neq \Theta(n^2)$
b) $n^2 = \omega(n \log n)$
c) $9n^3 + 3\log n = \Omega(n^3)$
d) $n (\log n)^2 = o(n^2)$

7) Ordene as funções abaixo em ordem crescente de complexidade assintótica:

n^2 , 2^n , 1 , $\sqrt{log\, n}$, $\log_3 n$, $n \log^2 n$, $2^{(n+3)}$, $(\sqrt{2})^{\log n}$, $n\, 2^n$

8) Uma sequência de n operações é executada sobre uma estrutura de dados. A i-ésima operação custa i se i é uma potência exata de 2 e 1 em caso contrário. Utilize a análise agregada para determinar o custo amortizado por operação.

9) Considere o seguinte problema: Cada um entre n clientes entrega um chapéu ao funcionário da chapelaria em um restaurante. O funcionário devolve os chapéus aos clientes em ordem aleatória. Qual é o número esperado de clientes que recebem de volta seus próprios chapéus? Para resolver este problema, considere o uso de variáveis indicadoras X_i = o cliente i recebe de volta o seu chapéu.

Lista de Exercícios 1 - PAA

Thales Henrique Silva

April 6, 2025

1

a) O algoritmo compara dois a dois, da direita para a esquerda, os elementos vizinhos do vetor. Se o elemento mais à esquerda for maior que o seu vizinho à direita, eles trocam de lugar. Isso se repete de forma que, após a primeira iteração do loop externo, o menor elemento vai estar na primeira posição do vetor, após a segunda iteração, o segundo menor elemento vai ocupar a segunda posição, e assim, por diante, até o vetor estiver completamente ordenado.

b) A invariante do loop externo é que, no final da i-ésima iteração, os i menores elementos ocuparão suas devidas posições em ordem não-decrescente. Isso é verdade no caso base de um vetor com N=2 elementos. Uma única iteração será suficiente para ordenar, pois o menor elemento é deslocado para a esquerda. Suponha que seja verdade para um certo k. Vamos provar que vai continuar verdadeiro para k+1. Por construção, o algoritmo percorre o vetor da direita para a esquerda e desloca o próximo menor elemento até a posição k+1, sem perturbar os k elementos já ordenados. Logo, a invariante é sempre mantida e o algoritmo está correto. Uma observação é que precisamos de fazer 1 iteração a menos que o tamanho do vetor, pois após o último passo, o maior elemento vai estar no final.

c) Seja N o tamanho do vetor. Dado que uma operação constante c é sempre realizada no if, a função de complexidade das comparações é:

$$f(N) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c$$

$$\text{Movendo a constante } c \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} 1$$

$$\text{Resolvendo o somatório interno } c \sum_{i=1}^{N-1} N - i$$

$$\text{Desmembrando o somatório } c \sum_{i=1}^{N-1} N - c \sum_{i=1}^{N-1} i$$

$$\text{Resolvendo o somatório da esquerda } c(N)(N-1) - c \sum_{i=1}^{N-1} i$$

$$\text{Resolvendo o somatório da direta } c(N^2 - N) - c(\frac{N^2}{2} - \frac{N}{2})$$

$$c(\frac{N^2}{2} - \frac{N}{2})$$

No caso da função de complexidade das trocas, ela depende da distribuição de elementos. No melhor caso, se o vetor estiver perfeitamente ordenado, nenhuma troca ocorre, pois a comparação no if sempre retorna falso, então a função seria nula. Por outro lado, no pior caso, o vetor estaria ordenado de forma reversa. Toda iteração geraria uma troca porque o elemento da esquerda seria maior que o vizinho da direita. Assim, a função seria praticamente a mesma que a das comparações, porém com outras constantes.

2

a) O algoritmo está correto no caso base em que $n = 1$. Suponha pela hipótese indutiva que ele está correto para k elementos. Ele também está correto para $k + 1$ porque, se o elemento $A[k + 1]$ for maior que max, por construção, a variável max será atualizada. O raciocínio é o mesmo para min. O algoritmo termina devido ao fato de que o loop executa um número finito de iterações. Portanto, o algoritmo está correto.

b) No melhor caso, $A[1] = max = min$ e não é necessário fazer mais atualizações. Apenas a comparação com o max seria realizada em cada iteração, logo, $f(n) = (n-2)+1 = n-1$. No pior caso, o max está na primeira posição, de forma que a primeira comparação sempre retornaria falso e forçaria a segunda comparação. $Logo, f(n) = 2(n-1)$.

c) A primeira comparação sempre ocorre. A segunda só ocorre se $A[i] \leq max$. Se os elementos são distintos e uniformemente distribuídos, a probabilidade de $A[i]$ ser maior que todos os anteriores é $\frac{1}{i}$ (semelhante ao problema da contratação visto nas aulas). Logo, $P(A[i] \leq max) = 1-\frac{1}{i}$. Assim, o número esperado de comparações na i-ésima iteração é $E[C] = 1+(1-\frac{1}{i}) = 2+\frac{1}{i}$. Somando de $i = 2$ até n: $\sum_{i=2}^{n} 2-\frac{1}{i} = 2(n-1)-\sum_{i=2}^{n} 1/i = 2n-H_n-1 \approx 2n-ln\ n-C_1$, conforme o número harmônico se aproxima do logaritmo natural.

3

a) A condição de instabilidade permanece praticamente a mesma: algum hospital prefere outro estudante, e este estudante também tem preferência por tal hospital. Este novo par não pode estar na lista de pares proibidos.

b) Basta ajustar a lista de preferências dos hospitais excluindo-se os estudantes que o hospital jamais aceitaria, assim como os estudantes que não aceitariam o respectivo hospital. Dessa forma, o hospital nunca poderá propor um par proibido e o restante do algoritmo permanece igual.

c) A prova de término é a mesma para o algoritmo tradicional: a execução acaba quando todos os hospitais tiverem preenchido as vagas ou esgotado suas listas de preferências.

Para a estabilidade, considere um par estudante-hospital (e, h) qualquer que não está no casamento gerado. Se h nunca propôs para e, então ou e está mais no fundo da lista de preferência de h, ou e foi excluído da lista do hospital h pelo fato deste par ser proibido. Se h propôs para e, e este não aceitou, então e prefere seu hospital atual do que h. De qualquer forma, este par (e, h) não é instável.

4

a)

$$
\begin{aligned}
T(n) &= T(n/2) + 1 \\
T(n) &= [T(n/4) + 1] + 1 \\
T(n) &= [T(n/8) + 1] + 2 \\
T(n) &= [T(n/16) + 1] + 3 \\
&\ldots \\
T(n) &= T(n/2^i) + i \\
T(1) &= c \\
n/2^i = 1 &\implies i = log\ n \\
T(n) = T(1) + log\ n &= c + O(log\ n) = O(log\ n)
\end{aligned}
$$

b)

$$\begin{aligned} T(n) &= T(n-1) + n^2 \\ T(n-1) &= T(n-2) + (n-1)^2 \\ T(n-2) &= T(n-3) + (n-2)^2 \\ &... \end{aligned}$$

$T(n)$ é uma soma de quadrados, cuja fórmula fechada é $O(n^3)$. Para confirmar iremos usar indução.
O caso base é verdadeiro: $T(1) = 1^2 \leq c \cdot 1^3 = c$
Supondo que é verdade para $T(n-1) \leq c \cdot (n-1)^3$
Vamos provar que $T(n) \leq c \cdot n^3$

$$\begin{aligned} T(n) &= T(n-1) + n^2 \\ &\leq c(n-1)^3 + n^2 \leq c \cdot n^3 \\ c(n^2 - 2n + 1)(n-1) + n^2 &\leq c \cdot n^3 \\ c(n^3 - 3n^2 + 3n - 1) + n^2 &\leq c \cdot n^3 \\ cn^3 - (3c-1)n^2 + 3cn - c &\leq cn^3 \\ -(3c-1)n^2 + 3cn - c &\leq 0 \\ &\text{Supondo c=1:} \\ -2n^2 + 3n - 1 &\leq 0 \end{aligned}$$

O coeficiente do termo dominante $\mathbf{n^2}$ é negativo. Então a desigualdade é verdadeira para um n suficientemente grande.

c)

$$\begin{aligned} T(n) &= 2T(n/2) + n^3 \\ a &= 2, b = 2, f(n) = n^3 \\ log_2 2 = 1 &\implies f(n) = \Omega(n^{1+\epsilon}) \\ &\text{Caso 3:} \\ af(n/b) &\leq cf(n) \\ 2f(n/2) &\leq cf(n) \\ 2 \cdot \frac{n^3}{8} &\leq cn^3 \\ \frac{n^3}{4} &\leq \frac{1}{4}n^3 \end{aligned}$$

O teorema se aplica. T(n) é $\theta(n^3)$.

d)

$$T(n) = 4T(n/2) + n^{2.5}$$
$$a = 4, b = 2, f(n) = n^{2.5}$$
$$log_2 4 = 2 \implies f(n) = \Omega(n^{2+\epsilon})$$
$$\text{Caso 3:}$$
$$af(n/b) \leq cf(n)$$
$$4f(n/2) \leq cf(n)$$
$$4\frac{n^{2.5}}{2^{2.5}} \leq cn^{2.5}$$
$$4\frac{n^{2.5}}{4 \cdot 2^{0.5}} \leq cn^{2.5}$$
$$\frac{n^{2.5}}{2^{0.5}} \leq cn^{2.5}$$

O teorema se aplica. T(n) é $\theta(n^{2.5})$.

5

a) $O(n^4)$, polinômio de maior grau domina todos os menores, independente das constantes

b) $O(2^n)$, exponencial domina logaritmos e polinômios

c) $O(log\ n)$, pois $log_3 10$ é uma constante

d) $O(n\ log\ n)$, pois

$$C_1 n\ log\ n \geq n$$
$$C_1\ log\ n \geq 1$$

e

$$C_2 n\ log\ n \geq log\ n$$
$$C_2 n\ \geq 1$$

e) $O(4^n)$, pois uma exponencial domina uma função linear e uma exponencial de base maior domina uma exponencial de base menor:

$$\begin{aligned} 4^n &\geq 2^n \\ (2^2)^n &\geq 2^n \\ 2^{2n} &\geq 2^n \\ 2n &\geq n \\ n &\geq 0 \end{aligned}$$

6

a) Verdadeiro, fazendo

$$\begin{aligned} C_1 &= 6 : \\ C_1 n^3 &\geq 3n^3 + 2n^2 + n + 1 \\ 6n^3 &\geq 3n^3 + 2n^2 + n + 1 \\ 3n^3 + 2n^3 + n^3 &\geq 3n^3 + 2n^2 + n + 1 \\ 2n^3 + n^3 &\geq 2n^2 + n + 1 \\ 2n^3 - 2n^2 + n^3 - n &\geq 1 \\ 2n^2(n-1) + n^2(n-1) &\geq 1 (\text{Sempre verdadeiro para } n > 1) \end{aligned}$$

b) Verdadeiro,

$$C_1 n^2 \leq 6n^3 + 5n^2 \leq C_2 n^2$$

dividindo por n^2, que é sempre positivo e diferente de 0:

$$C_1 \leq 6n + 5 \leq C_2$$

Não é possível satisfazer o lado direito, pois C_2 é constante e $6n$ cresce ao infinito.

c) Verdadeiro, pois independente de C_1, o crescimento do lado esquerdo vai superar o direito em algum momento. A função n domina $log\ n$.

$$\begin{aligned} n^2 &\geq C_1 n\ log\ n \\ n &\geq C_1\ log\ n \end{aligned}$$

d) Verdadeiro, fazendo

$$\begin{aligned} C_1 = 9: \\ 9n^3 + 3logn \geq C_1 n^3 \\ 9n^3 + 3logn \geq 9n^3 \\ 3log\ n \geq 0 \end{aligned}$$

e) Verdadeiro, calculando o limite:

$$\begin{aligned} \lim_{n\to\infty} \frac{n(\log n)^2}{n^2} \\ \lim_{n\to\infty} \frac{(\log n)^2}{n} \\ \lim_{n\to\infty} \frac{2\log n}{n} \\ \lim_{n\to\infty} \frac{2}{n} = 0 \end{aligned}$$

7

$1, \sqrt{log\ n}\ , \log_3 n\ , n\ (log\ n)^2,\ n^2\ , \sqrt{2}^{log\ n}, 2^n\ e\ 2^{n+3}, n2^n$

Um logaritmo com expoente menor perde para um logaritmo com expoente maior; n^2 ganha de $n\ (log\ n)^2$ porque ao cancelar o fator n, o logaritmo vai perder para o polinômio, independente dos respectivos expoentes (positivos); $\sqrt{2}^{log\ n}$ vai perder para as outras exponenciais porque tanto a base $\sqrt{2}$ quanto o seu fator exponencial $log\ n$ são menores; $2^n\ e\ 2^{n+3}$ são equivalentes pois diferem apenas por um fator constante igual a 8. Por fim, $n2^n$ ganha das outras exponenciais porque após cancelar o fator 2^n, ainda resta um crescimento n.

8

O custo real das n operações, indo de 1 até n, é dado a seguir:

$$\sum_{i=1}^{n} f(i)$$

, em que $f(i)$ é a função que retorna o devido valor, 1 ou i, dependendo de i.

Primeiramente, é necessário estimar a soma das potências exatas de 2 indo até n:

$$\sum_{j=0}^{x} 2^j = 1 + 2 + 4 + ... + 2^x$$

Considere as desigualdades a seguir:

$$\begin{aligned} 2^x &\le n \\ log\ 2^x &\le log\ n \\ x\ log\ 2 &\le log\ n \\ x\ &\le log\ n \end{aligned}$$

Ou seja, o expoente x da maior potência de 2 até n é $x = \lfloor log\ n \rfloor$. A função chão é aplicada pois queremos um número inteiro. Agora podemos simplificar o somatório anterior:

$$\sum_{j=0}^{x} 2^j = \sum_{j=0}^{\lfloor log\ n \rfloor} 2^j$$

Isso é uma progressão geométrica de razão $q = 2$ e $1 + \lfloor log\ n \rfloor$ termos, cuja soma é:

$$S_n = a \cdot \frac{1 - q^n}{1 - q} = \frac{1 - 2^{1+\lfloor log\ n \rfloor}}{1 - 2} = \frac{1 - 2^{1+\lfloor log\ n \rfloor}}{-1} = 2^{1+\lfloor log\ n \rfloor} - 1$$

Para fins de análise de complexidade, podemos ignorar a subtração por 1 conforme n cresce:

$$2^{1+\lfloor log\ n \rfloor} - 1 \approx 2^{1+\lfloor log\ n \rfloor} = 2^1 \cdot 2^{\lfloor log\ n \rfloor} \le 2^1 \cdot 2^{log\ n} = O(2n) = O(n)$$

Veja que o chão $\lfloor log\ n \rfloor$ é sempre limitado por $log\ n$. Isso permite cancelar o logaritmo com a exponencial e concluir que o somatório tem crescimento $O(n)$.

Agora, observe que

$$\sum_{i=1}^{n} f(i) \le \sum_{i=1}^{n} 1 \ + \sum_{j=0}^{\lfloor log\ n \rfloor} 2^j$$

Ou seja, estamos somando 1 em todos os termos independente de i ser uma potência de 2. Logo, o custo real é garantidamente menor que ou igual a esse custo amortizado.

Veja que essa nova soma é simplesmente n:

$$\sum_{i=1}^{n} 1 = n = O(n)$$

Já provamos que a soma das potências de 2 tem complexidade $O(n)$. Visto que o custo real é limitado pela soma de dois somatórios $O(n)$, o custo médio de cada operação pela análise agregada será $(O(n) + O(n))/n = O(1)$.

9

O número de clientes que recebeu seu chapéu é:

$$X = \sum_{i=1}^{n} X_i$$

$$E[X] = E[\sum_{i=1}^{n} X_i]$$

Pela linearidade da esperança:

$$E[X] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{1}{n} = 1$$

Como a distribuição é uniforme, a chance de cada cliente receber seu chapéu é $E[X_i] = \frac{1}{n}$. Logo, o valor esperado total é 1.