

MAC0338 - Análise de Algoritmos
Departamento de Ciência da Computação
Segundo semestre de 2023

Lista 4

Entregar: Exercícios 8, 14 e 18(a).

Instruções: Leia as instruções postadas no e-disciplinas. Além disso, para os exercícios que requerem um algoritmo você deve: escrever um pseudocódigo, descrever seu funcionamento, provar corretude e fazer uma análise formal do consumo de tempo.

1. **(CLRS 8.2-4)** Descreva um algoritmo que, dados n inteiros no intervalo de 1 a k , preprocesse sua entrada e então responda em $O(1)$ qualquer consulta sobre quantos dos n inteiros dados caem em um intervalo $[a..b]$. O préprocessamento efetuado pelo seu algoritmo deve consumir tempo $O(n + k)$.
2. **(CLRS 8.3-4)** Mostre como ordenar n inteiros no intervalo de 0 até $n^2 - 1$ em tempo $O(n)$.
3. Desenhe a árvore de decisão para o SELECTIONSORT aplicado a $A[1..3]$ com todos os elementos distintos.
4. **(CLRS 8.1-1)** Qual a menor profundidade (= menor nível) que uma folha pode ter em uma árvore de decisão que descreve um algoritmo de ordenação baseado em comparações?
5. Mostre que $\lg(n!) \geq (n/4) \lg n$ para $n \geq 4$ sem usar a fórmula de Stirling.
6. **(CLRS 8.1-3)** Mostre que não há algoritmo de ordenação baseado em comparações cujo consumo de tempo é linear para pelo menos metade das $n!$ permutações de 1 a n . O que acontece se trocarmos “metade” por uma fração de $1/n$? O que acontece se trocarmos “metade” por uma fração de $1/2^n$?
7. **(CLRS 8.2-1)** Simule a execução do COUNTINGSORT usando como entrada o vetor

$$A[1..11] = \langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2 \rangle.$$

8. **(CLRS 8.2-2)** Prove que o COUNTINGSORT é estável.
9. **(CLRS 8.2-3)** Suponha que o **para** da linha 7 do COUNTINGSORT é substituído por

para $j \leftarrow 1$ até n faça

Prove que o COUNTINGSORT ainda funciona. O algoritmo resultante continua estável?

10. **(CLRS 8.3-2)** Quais dos seguintes algoritmos de ordenação são estáveis: insertionsort, mergesort, heapsort, e quicksort. Descreva uma maneira simples de deixar qualquer algoritmo de ordenação estável. Quanto tempo e/ou espaço adicional a sua estratégia usa?
11. **(CLRS 8.4-1)** Simule a execução do BUCKETSORT com o vetor

$$A[1..10] = \langle 0.79, 0.13, 0.16, 0.64, 0.39, 0.20, 0.89, 0.53, 0.71, 0.42 \rangle.$$

12. **(CLRS 8.4-2)** Qual é o consumo de tempo de pior caso para o BUCKETSORT? Que simples ajuste do algoritmo melhora o seu pior caso para $O(n \lg n)$ e mantém o seu consumo esperado de tempo linear.
13. **(CLRS 28.2-5)** Quão rápido você consegue multiplicar uma matriz $kn \times n$ por uma $n \times kn$ usando o algoritmo de Strassen como uma subrotina? Responda a mesma questão com a ordem das matrizes de entrada invertida.
14. **(CLRS 28.2-6)** Escreva um algoritmo que multiplica dois números complexos $a + bi$ e $c + di$ usando apenas três multiplicações reais. O seu algoritmo deve receber como entrada os números a , b , c e d e devolver os números $ac - bd$ (componente real do produto) e $ad + bc$ (componente imaginária do produto).
15. No SELECT-BFPRT, os elementos do vetor são divididos em grupos de 5. O algoritmo continua linear se dividirmos os elementos em grupos de 7? E em grupos de 3? Prove sua resposta.
16. Considere a seguinte variante do PARTICIONE-BFPRT, que chamaremos de PARTICIONE-D. Em vez de acionar o SELECT-BFPRT para calcular a mediana das medianas, ela aciona recursivamente o próprio PARTICIONE-D, para calcular uma “mediana aproximada” do vetor das medianas. Suponha que o PARTICIONE-D rearranja o vetor $A[p..d]$ e devolve um índice q tal que $A[p..q-1] \leq A[q] < A[q+1..d]$ e $\max\{k, n-k\} \leq 9n/10$, onde $n = d - p + 1$ e $k = q - p + 1$. Analise o consumo de tempo da variante do SELECT-BFPRT que chama o PARTICIONE-D em vez do PARTICIONE-BFPRT.
17. Tente provar por indução a suposição feita sobre o PARTICIONE-D no exercício acima. (Você pode assumir que para vetores pequenos, por exemplo, com até 5 elementos, o PARTICIONE-D devolve uma mediana do vetor.) Apresente a prova da suposição ou explique porque você não consegue prová-la. O que acontece com a sua prova/argumento caso você substitua a fração $9/10$ por uma outra fração α mais próxima de 1? Caso você não consiga provar a suposição, você é capaz de descrever um contra-exemplo para ela?
18. A *silhueta de um prédio* é uma tripla (l, h, r) de números positivos com $l < r$, onde h representa a altura do prédio, l representa a posição inicial da sua base e r a posição final.

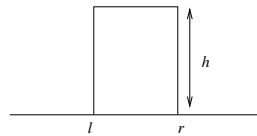


Figura 1: Silhueta (l, h, r) de um prédio.

Considere uma coleção $\mathcal{S} = \{(l_1, h_1, r_1), \dots, (l_n, h_n, r_n)\}$ com a silhueta de n prédios. Para cada número positivo x , denote por \mathcal{S}_x o conjunto $\{i : 1 \leq i \leq n \text{ e } l_i \leq x \leq r_i\}$. Denote ainda por $h(\mathcal{S}_x)$ o número $\max\{h_i : i \in \mathcal{S}_x\}$.

O *skyline* de \mathcal{S} é uma sequência $(x_0, t_1, x_1, \dots, t_k, x_k)$ tal que

1. $x_0 = 0$ e $x_k = \max\{r_i : 1 \leq i \leq n\}$;
2. $x_{j-1} < x_j$ para $j = 1, \dots, k$;
3. para $j = 1, \dots, k$, $t_j = h(\mathcal{S}_{x_j})$ para todo x tal que $x_{j-1} < x < x_j$;

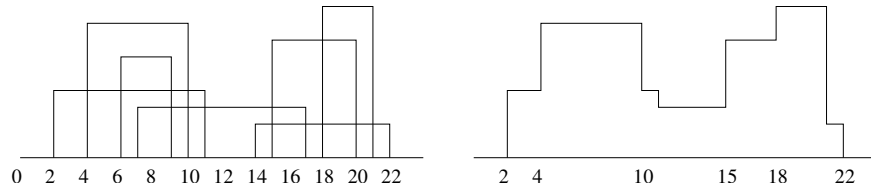


Figura 2: Coleção de silhuetas $\mathcal{S} = \{(15, 7, 20), (4, 8, 10), (18, 9, 21), (2, 4, 11), (7, 3, 17), (6, 6, 9), (14, 2, 22)\}$ e seu skyline $(0, 0, 2, 4, 4, 8, 10, 4, 11, 3, 15, 7, 18, 9, 21, 2, 22)$.

4. $t_j \neq t_{j+1}$ para $j = 1, \dots, k - 1$.

- (a) Escreva um algoritmo que recebe o skyline de uma coleção \mathcal{S}_1 de silhuetas de prédios e o skyline de uma coleção \mathcal{S}_2 de silhuetas de prédios e devolve o skyline de $\mathcal{S}_1 \cup \mathcal{S}_2$. Seu algoritmo deve consumir tempo $O(n)$, onde $n = |\mathcal{S}_1 \cup \mathcal{S}_2|$. Explique por que seu algoritmo faz o que promete e por que consome o tempo pedido.
- (b) Escreva um algoritmo que recebe um inteiro n e uma coleção \mathcal{S} de n silhuetas de prédios e devolve o skyline de \mathcal{S} . Seu algoritmo deve consumir tempo $O(n \lg n)$. Explique por que seu algoritmo faz o que promete e por que consome o tempo pedido.