Universidade Federal de Pernambuco
Centro de Informática
Bach. em Engenharia da Computação

IF672ec Algoritmos e Estr. de Dados 2023.1
Prof. Paulo Fonseca

Primeira Prova — 03 de Agosto de 2023

■ QUESTÃO 1 (2,0pt)

Considere o algoritmo a seguir

Algorithm *whatdoido*
Input *head* ptr. para cabeça de uma lista com sentinela
Output ???

```
1  chg ← True
2  while chg do
3      chg ← False
4      cur ← head
5      while cur→next→next ≠ ⊥ do
6          ncur ← cur→next
7          if ncur→val > ncur→next→val
           then
8              cur→next ← ncur→next
9              ncur→next ← ncur→next→
               next
10             cur→next→next ← ncur
11             chg ← True
12         cur ← cur→next
13 return head
```

a) Ilustre a execução sobre a entrada

$$\backslash \to 4 \to 2 \to 3 \to 0 \to 1 \to \bot$$

exibindo a lista ao final de cada iteração do loop externo (linhas 2–15).

b) Descreva brevemente ***o que*** o algoritmo faz, ou seja, que resultado produz a partir da entrada.

c) Qual a complexidade assintótica do ***pior caso***? Justifique sucintamente.

■ QUESTÃO 2 (2,0pt)

Dados um vetor de inteiros $V = (v_0, \ldots, v_{n-1})$ e um inteiro S, queremos encontrar dois elementos v_i e v_j de V tais que $v_i + v_j = S$. Esse problema pode ser resolvido com auxílio de uma tabela de dispersão (***Hashtable***) da seguinte maneira. Inicie com uma hashtable H vazia. Para cada $i = 0, \ldots, n-1$, procure $d = S - v_i$ em H. Caso encontre, retorne (d, v_i). Caso não encontre, insira v_i em H e continue. Se, após haver inserido todos os elementos de V, ainda não tiver encontrado um par de soma S, é porque tal par não existe, e nesse caso retorne $\perp$.

Considere uma hashtable ***aberta*** de tamanho inicial $m = 3$ com função de dispersão $h(k) = k \mod m$. Nesta tabela, imediatamente antes de cada inserção, se o ***fator de carga*** for maior que o limite $\alpha_{max} = 0.5$, executa-se o ***rehashing***, atualizando o tamanho da tabela para $2m + 1$.

Ilustre a execução do algoritmo sobre a entrada

$$V = (28, 54, 25, 60, 14, 44, 45) \quad S = 70,$$

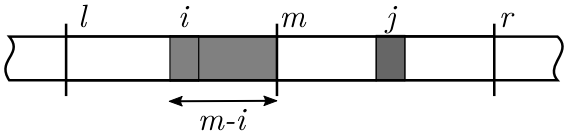
exibindo a tabela após cada inserção realizada. Indique o que o algoritmo retorna neste caso.

■ QUESTÃO 3 (2,0pt)

Num vetor de inteiros $V = (v_0, \ldots, v_{n-1})$, temos uma ***inversão*** quando um valor maior está à esquerda de um valor menor, isto é quando um par (i, j) é tal que $i < j$ e $v_i > v_j$. Podemos contar a quantidade de inversões de um array com uma variação do ***Mergesort***, como explicado a seguir.

A função ***merge*** combina dois segmentos adjacentes ordenados $V[l : m] = (v_l, \ldots, v_{m-1})$ e $V[m : r] = (v_m, \ldots, v_{r-1})$. Se, ao comparar um elemento v_i do primeiro

segmento com um elemento v_j do segundo segmento, tivermos $v_j < v_i$, então v_j está invertido com relação a v_i e todos os demais elementos do primeiro segmento à direita da posição i, ou seja, devemos contabilizar mais $m - i$ inversões.



Ilustre a execução deste ***Mergesort*** modificado sobre o vetor de entrada

$$V = (4, 0, 6, 2, 7, 3, 1, 5)$$

exibindo o vetor V imediatamente após cada execução da função ***merge***, juntamente com o número de inversões contabilizadas até então.

■ QUESTÃO 4 (2,0pt)

a) Desenhe uma AVL com 8 nós de valores $10, 20, 30, \ldots, 80$, cuja soma dos ***fatores de balanço*** dos nós é ***máxima***.

b) Ilustre a inserção da chave 45 nessa árvore, representando as rotações necessárias.

■ QUESTÃO 5 (2,0pt)

Suponha que queiramos amarrar progressivamente n cordas de comprimentos $L = (l_0, \ldots, l_{n-1})$ de maneira a formar uma única corda. Cada vez que unimos as extremidades de duas cordas com um nó, transformamos essas duas cordas numa só, e portanto a quantidade de cordas diminui em uma unidade. Desta forma, após $n - 1$ nós, teremos uma única corda. Suponha adicionalmente que queiramos realizar a sequência de nós com menor custo total, sendo que o custo de um nó é igual à soma dos comprimentos das duas cordas unidas.

Por exemplo, se tivermos $n = 3$ cordas de comprimentos $L = (1, 2, 3)$, podemos unir as cordas de comprimentos 1 e 2, com um nó de custo $c_0 = 1 + 2 = 3$, e então teremos duas cordas de comprimentos $(3, 3)$. Para uní-las, precisaremos de um nó de custo $c_1 = 3 + 3 = 6$, e portanto o custo total da operação será $C = c_0 + c_1 = 3 + 6 = 9$. De fato, este é o custo mínimo. Se, por exemplo, houvéssemos unido primeiro as cordas de comprimentos 2 e 3, teríamos $c_0 = 5$ e $c_1 = 6$, e portanto o custo total seria $C = 11$.

Esse problema pode ser resolvido com o auxílio de uma ***min-heap*** da seguinte forma:

1. Transforme L numa min-heap
2. Inicialize $C \leftarrow 0$
3. Enquanto $heap\_size(L) > 1$:
   3.1 $x \leftarrow heap\_extract(L)$
   3.2 $y \leftarrow heap\_extract(L)$
   3.3 $s \leftarrow x + y$
   3.4 $heap\_insert(L, s)$
   3.5 $C \leftarrow C + s$

Considerando a execução do algoritmo acima sobre a entrada

$$L = (4, 3, 2, 6, 1)$$

a) Ilustre a construção *offline* da minh-heap do passo 1.

b) Ilustre a execução do laço do passo 3, exibindo a min-heap na forma de árvore após cada passo 3.1, 3.2 e 3.4.

■ ■ ■