Avaliação III

DIM0320 — Algoritmo e Programação de Computadores

2015.1

1 Quadrado mágico (3.5)

Uma matriz *quadrada* de dimensão $n \times n$ é um **quadrado mágico** de ordem n se todos os inteiros $1, 2, 3, 4, ..., n^2$ aparacem na matriz e se, em cada linha, cada coluna, e cada diagonal, a soma dos elementos resulta em um mesmo valor.

A seguinte matriz é um quadrado mágico:

$$\begin{pmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{pmatrix}$$

1. Escrever uma função que recebe um inteiro i e uma matriz e calcula a soma da linha i da matriz m de dimensão $n \times n$, segundo a seguinte declaração.

```
funcao soma_lin(m: vetor [1..100, 1..100] de inteiro; i, n : inteiro) : inteiro
```

   Nas próximas questões, suponha a existência de uma função

```
funcao soma_col(m: vetor [1..100, 1..100] de inteiro; j, n: inteiro) : inteiro
```

   que realiza a soma de uma dada coluna j duma matriz de tamanho $n \times n$.

2. Escrever uma função que verifica se todos os inteiros do intervalo $[1, n^2]$ ocorrem numa dada matriz $n \times n$, $n \leq 100$.

```
funcao verif(m: vetor [1..100, 1..100] de inteiro; n : inteiro): logico
```

3. Utilizar essas funções para escrever uma função para verificar se uma dada matriz é um quadrado mágico de ordem $n \leq 100$, de acordo com a seguinte declaração.

```
funcao quadrado_magico(m: vetor [1..100, 1..100] de inteiro; n : inteiro) : logico
```

```
algoritmo "qm"
var i, j, n : inteiro
    soma : inteiro
    m: vetor[1..100, 1..100] de inteiro

funcao soma_lin(m: vetor[1..100, 1..100] de inteiro; i, n : inteiro) : inteiro
var k, soma : inteiro
inicio
   soma <- 0
   para k de 1 ate n faca
     soma <- soma + m[i, k]
   fimpara
fimfuncao

funcao soma_col(m: vetor[1..100, 1..100] de inteiro; j, n : inteiro) : inteiro
var k, soma : inteiro
inicio
   soma <- 0
   para k de 1 ate n faca
```

```
20      soma <- soma + m[k, j]
21    fimpara
22  fimfuncao
23
24
25  funcao verif(m: vetor[1..100, 1..100] de inteiro; n : inteiro): logico
26  var v : vetor [1..10000] de logico
27      i, j : inteiro
28  inicio
29      para i de 1 ate n * n faca
30        v[i] <- falso
31      fimpara
32
33      para i de 1 ate n faca
34        para j de 1 ate n faca
35          se m[i, j] > n * n ou m[i, j] < 1 entao retorne falso
36          senao
37            v[m[i, j]] <- verdadeiro
38          fimse
39        fimpara
40      fimpara
41
42      i <- 1
43      enquanto i <= n * n faca
44        se nao v[i] entao retorne falso fimse
45        i <- i + 1
46      fimenquanto
47      retorne verdadeiro
48  fimfuncao
49
50  funcao quadrado_magico(m: vetor [1..100, 1..100] de inteiro; n : inteiro) : logico
51  var i, j : inteiro
52      soma, soma_check : inteiro
53  inicio
54    se nao verif(m, n) entao
55      retorne falso
56    fimse
57
58    soma <- soma_lin(m, 1, n)
59    para i de 2 ate n faca
60      se soma_lin(m, i, n) <> soma entao
61        retorne falso
62      fimse
63    fimpara
64
65    para i de 1 ate n faca
66      se soma_col(m, i, n) <> soma entao
67        retorne falso
68      fimse
69    fimpara
70
71    soma_check <- m[1, 1]
72    para i de 2 ate n faca
73       soma_check <- soma_check + m[i, i]
74    fimpara
75    se soma_check <> soma entao
76      retorne falso
77    fimse
```

```
78
79      soma_check <- m[1, n]
80      para i de 2 ate n faca
81          soma_check <- soma_check + m[i, n - i + 1]
82      fimpara
83      se soma_check <> soma entao
84        retorne falso
85      fimse
86
87      retorne verdadeiro
88  fimfuncao
89
90  inicio
91      leia(n)
92      para i de 1 ate n faca
93        para j de 1 ate n faca
94            leia(m[i, j])
95        fimpara
96      fimpara
97
98      se quadrado_magico(m, n) entao
99        escreva("m e um quadrado magico de ordem ", n)
100     senao
101       escreva("m nao e um quadrado magico de ordem ", n)
102     fimse
103 fimalgoritmo
```

2 K-ésimo menor elemento de uma sequência (3)

Escrever uma função que determina o k-ésimo menor elemento de um vetor de inteiros dentro dos índices $[p, u]$ e retorna o valor deste elemento.

A declaração da função é:

```
funcao kmenor(v : vetor [1..100] de inteiro; k, p, u: inteiro): inteiro
```

Os parâmetros da função verificam:

- $u \geq p$
- $1 \leq u - p + 1 \leq 100$
- $1 \leq k \leq u - p + 1$

```
algoritmo "kesimo"
var v: vetor [1..100] de inteiro
    i : inteiro

funcao kmenor(v : vetor [1..100] de inteiro; k, first, last: inteiro): inteiro
var tmp, j, i, km : inteiro
inicio
   j <- (first + last) \ 2

   se first = last e 1 = k entao
     km <- v[first]
   senao
   se first < last entao
     tmp <- v[last]
     v[last] <- v[j]
     v[j] <- tmp
     j <- last - 1
     i <- first
     enquanto i <= j faca
       se v[i] > v[last] entao
         tmp <- v[j]
         v[j] <- v[i]
         v[i] <- tmp
         j <- j - 1
       senao
         i <- i + 1
       fimse
     fimenquanto

      tmp <- v[last]
      v[last] <- v[i]
      v[i] <- tmp

      se i - first + 1 = k entao
        km <- v[i]
      senao
        se i - first + 1 > k entao
          km <- kmenor(v, k, first, i - 1)
        senao
          km <- kmenor(v, k - (i - first + 1), i + 1, last)
        fimse
        fimse
   fimse
   fimse
```

```
45      retorne km
46  fimfuncao
47
48  inicio
49      para i de 1 ate 10 faca
50          v[i] <- i
51      fimpara
52
53      para i de 1 ate 10 faca
54          escreval(i, "-esimo menor : ", kmenor(v, i, 1, 10))
55      fimpara
56
57
58  fimalgoritmo
```

3 Hierosolyma Est Perdita (3.5)

Responder às seguintes questões usando o algoritmo abaixo.

1. Desenha o fluxograma desse algoritmo.

1. Desenha a árvore de chamada do algoritmo, detalhando os valores dos parâmetros. A árvore começa da função principal.

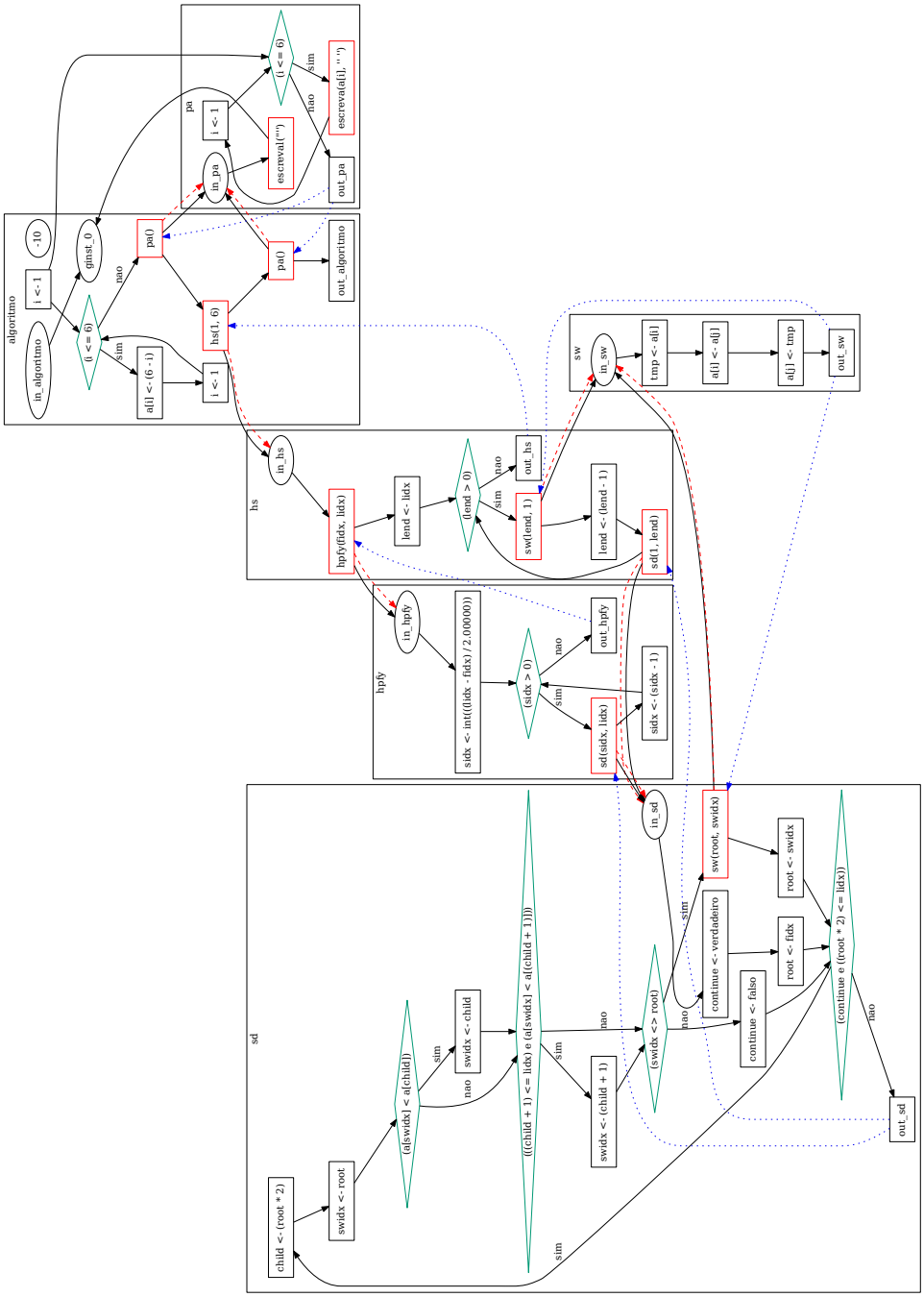
2. O que será escrito na tela ?

3. O que faz esse algoritmo ?

```
algoritmo "h"
var a : vetor [1..6] de inteiro
    i : inteiro

procedimento pa ()
inicio
  escreval("")
  para i de 1 ate 6 faca
    escreva(a[i], " ")
  fimpara
fimprocedimento

procedimento sw(i, j: inteiro)
var tmp: inteiro
inicio
  tmp <- a[i]
  a[i] <- a[j]
  a[j] <- tmp
fimprocedimento

procedimento sd(fidx, lidx:inteiro)
var root, swidx, child: inteiro
    continue: logico
inicio
   continue <- verdadeiro
   root <- fidx
   enquanto continue e (root * 2 <= lidx) faca
     child <- root * 2
     swidx <- root
     se a[swidx] < a[child] entao
        swidx <- child
     fimse
     se (child + 1 <= lidx) e (a[swidx] < a[child + 1]) entao
        swidx <- child + 1
     fimse
     se swidx <> root entao
        sw(root, swidx)
        root <- swidx
     senao
        continue <- falso
     fimse
   fimenquanto
fimprocedimento

procedimento hpfy(fidx, lidx: inteiro)
var sidx: inteiro
inicio
```

```
48    sidx <- int((lidx - fidx) / 2.0)
49    enquanto sidx > 0 faca
50      sd(sidx, lidx)
51      sidx <- sidx - 1
52    fimenquanto
53  fimprocedimento
54
55
56  procedimento hs(fidx, lidx: inteiro)
57  var lend:inteiro
58  inicio
59    hpfy(fidx, lidx)
60    lend <- lidx
61    enquanto lend > 0 faca
62      sw(lend, 1)
63      lend <- lend - 1
64      sd(1, lend)
65    fimenquanto
66  fimprocedimento
67
68
69  inicio
70    para i de 1 ate 6 faca
71          a[i] <- 6 - i
72    fimpara
73    pa()
74    hs(1, 6)
75    pa()
76  fimalgoritmo
```

1. O fluxograma é:



1. O algoritmo realiza as chamadas sucessivas:

```
pa()
hs(1,6)
hpfy(1,6)
sd(2,6)
sd(1,6)
sw(6,1)
sd(1,5)
sw(1,2)
sw(2,4)
sw(5,1)
sd(1,4)
sw(1,3)
sw(4,1)
sd(1,3)
sw(1,2)
sw(3,1)
sd(1,2)
sw(2,1)
sd(1,1)
sw(1,1)
sd(1,0)
pa()
```

2. A saída na tela será:

```
5 4 3 2 1 0
0 1 2 3 4 5
```

3. Este algoritmo é uma implementação do algoritmo de ordenação *heapsort*. Veja `https://pt.wikipedia.org/?title=Heapsort` para mais detalhes.