21

24

{P2 = Q1}

23

i: N <= i < 2\*N: r[i] = C[i-N])}

(

∀

∧

22

i: 0 <= i < N: r[i] = B[i])

(

∀

{Cota2: 2\*N-1 - x}

{Q1:

20

i = i+1

19

fi

18

r[i] = C[i-N]

17

[] i >= N ->

16

r[i] = B[i]

29

1

-

-

33

r[x] = r[x+1];

32

h = r[x];

31

;if r[x] > r[x+1]

30

;do x != 2\*N-1 ->

15

{Order pairwise}

28

0 <= x <= N}

∧

27

k: 0 <= k < 2\*x-1: r[k] <= r[k+1])

(

∀

26

{Inv2:

25

|[

6

var x,h,i = 0,0,0;

5

var r array [0,...,2N) of int;

4

const C array [0,...,N) of int;

3

const B array [0,...,N) of int;

2

const N: int;

1

{P1: N >= 0}

r

.

ordena de forma creciente en un tercer arreglo

B,C

y los

El siguiente programa toma dos arreglos enteros unidimensionales

1. Iteración en arreglos e invariantes

EJERCICIOS DE LA GUIA

Jhonny Lanzuisi

hecho por

*la mayoría tomados de la guía o los laboratorios*

i: i <= k < 2\*i: r[k] = C[k-i])

if i < N ->

14

;do i != 2\*N ->

13

{Catenate B and C}

12

{Cota t1: 2\*N - i}

11

0 <= i <= N}

∧

10

Algunos ejercicios en CGL

(

∀

∧

9

k: 0 <= k < i: r[k] = B[k])

(

∀

8

{Inv1:

7

17

22

x := x-1

21

b[x] == b[x-1] ->

while x >= 0

∧

20

K := 1

19

;if b[x] == l ->

18

{Cota t2: N+x-y}

K = K+1

od

16

]|

15

fi

14

[] b[h] > l -> y := h

13

;if b[h] <= l -> x := h

12

h := (x+y) div 2

Para

2

-

-

la

recibirá una matriz y regresará una variable booleana que indicará si

is\_premagic

que

función

una

construiremos

esto,

lograr

11

igual.

si tiene la menos una fila y columna tales que la suma de sus elementos sea

Queremos determinar si una matriz es pre-mágica. Una matriz es pre-mágica

3. Iteración en matrices. Matriz Pre-mágica

26

>>K

25

skip

24

[] b[x] != l ->

23

38

2. Iteración en arreglos

1.1. Ahora en python

42

]|

41

x: 0 <= x < 2\*N-1: r[x] <= r[x+1])}

(

∀

40

{Q2:

39

x = x+1

func bin\_search(l: int; b: array [0,..,N) of int) -> int

fi

37

skip

36

[] r[x] <= r[x+1]

35

x = -1

34

r[x+1] = h;

ALGUNOS EJERCICIOS EN CGL

5

|[var h: int;

10

;do x+1 != y ->

9

{Cota t1: y-x-1}

8

K = 0

7

x,y = 0,N

6

var K: int;

Jhonny Lanzuisi

var x,y: int;

4

K >= 0}

∧

3

b[k]=l : 1)

{Q: K = (+k : 0 <= k < N

∧

2

{P:}

1

28

34

i := i+1

33

colsum := colsum + A[i,c];

32

;do i != N ->

31

0 <= i <= N}

∧

30

{Inv: colsum = (+k: 0 <= k < i: A[k,c])

29

{Cota: N-i}

od

i,colsum := 0,0;

27

var colsum: int;

26

var i: int;

25

{Q: rowsum = (+i: 0 <= i < N: A[i,c])}

24

0 <= c < M}

M >= 0

{P: N >= 0

∧

∧

∧

3

-

-

43

ismagic := false

42

i,j := 0,0

41

ismagic: bool;

40

var i,j: int;

39

sum\_row(i,A,rw) = sum\_col(j,A,cl))}

23

38

0 <= j < M

(i,j): 0 <= i < N

{Q: ismagic = (

∧

∃

37

M >= 0}

{P: N >= 0

∧

36

func is\_magic(A: array [0,..,N)\*[0,..,M) of int) -> bool

35

Salida rowsum: int;

9

var j: int;

8

{Q: rowsum = (+j: 0 <= j < M: A[r,j])}

7

0 <= r < N}

N >= 0

{P: M >= 0

∧

∧

6

)

5

var rowsum: int;

4

Entrada A: array [0,..,N)\*[0,..,M) of int;

3

Entrada r: int;

2

proc sum\_row(

1

|[

de una fila específica por un lado y las entradas de una columna por el otro.

Harán falta dos funciones más, muy parecidas, que sumen las entradas

matriz es, o no, pre-mágica.

ALGUNOS EJERCICIOS EN CGL

16

)

22

Salida colsum: int;

21

Entrada A: array [0,..,N)\*[0,..,M) of int;

20

Entrada c: int;

19

proc sum\_col(

18

od

17

j := j+1

Jhonny Lanzuisi

rowsum := rowsum + A[r,j];

15

;do j != M ->

14

0 <= j <= M}

∧

13

{Inv: rowsum = (+k: 0 <= k < j: A[r,k])

12

{Cota: M-j}

11

j,rowsum := 0,0;

10

# Inv

13

j = j+1

12

rowsum = rowsum + A[r][j]

11

while j != M:

10

and 0 <= j <= M)

9

assert (rowsum == sum(A[r][k] for k in range(0,j))

8

# Inv

7

t = M - j # Cota

6

j,rowsum = 0,0

5

assert 0 <= r < N

4

assert N >= 0 and M >= 0

3

M = len(A[0])

2

19

4

-

-

23

# Test

22

return rowsum

21

assert rowsum == sum(A[r][j] for j in range(0,M))

20

assert t >= 0

N = len(A)

t = M-j

18

assert t > M-j

17

# Cota

16

and 0 <= j <= M)

15

assert (rowsum == sum(A[r][k] for k in range(0,j))

14

∧

51

ismagic = true

50

;if rw = cl ->

49

sum\_col(j,A,cl)

48

sum\_row(i,A,rw);

47

ismagic = false ->

;do j != M

[] rw != cl ->

46

{Cota 2: M-j}

45

ismagic = false ->

;do i != N

∧

44

{Cota 1: N-i}

ALGUNOS EJERCICIOS EN CGL

57

1

def sum\_row(r: int, A: list) -> int:

que todo funciona como debería. a

Ahora el programa anterior, pero traducido a python para poder verificar

3.1. Implementación de python

60

]|

59

>>ismagic

58

od

Jhonny Lanzuisi

i := i+1

56

od

55

j := j+1

54

fi

53

skip

52

return colsum

]

26

[0,3,-4,7]

25

[1,2,3,4],

24

A = [

23

# Test

22

27

21

assert colsum == sum(A[i][c] for i in range(0,N))

20

assert t >= 0

19

t = N-i

18

assert t > N-i

17

# Cota

i,j = 0,0

5

-

-

8

t2 = M-j # Cota 2

7

t1 = N-i # Cota 1

6

ismagic = False

5

16

4

assert N >= 0 and M >= 0

3

M = len(A[0])

2

N = len(A)

1

def is\_magic(A: list) -> bool:

28

sum\_col(2,A)

sum\_row(1,A)

assert 0 <= c < M

4

assert N >= 0 and M >= 0

3

M = len(A[0])

2

N = len(A)

1

def sum\_col(c: int, A: list) -> int:

28

5

27

]

26

[0,3,-4,7]

25

[1,2,3,4],

24

A = [

ALGUNOS EJERCICIOS EN CGL

while i != N:

and 0 <= i <= N)

15

assert (colsum == sum(A[k][c] for k in range(0,i))

14

# Inv

13

i = i+1

12

colsum = colsum + A[i][c]

11

Jhonny Lanzuisi

10

and 0 <= i <= N)

9

assert (colsum == sum(A[k][c] for k in range(0,i))

8

# Inv

7

t = N - i # Cota

6

i,colsum = 0,0

M

j := 0

11

isp := true

10

var j: int;

9

var isp: bool;

8

var B: array [0,..,M) of int;

7

N}

r

0

0

12

0

{P: N

≤

≤

∧

≥

∧

≥

6

)

5

Salida isp: bool;

4

19

6

-

-

24

od

23

j := j+1

22

skip

21

[] B[j] = A[r,j] ->

20

isp := false

Entrada A: array [0,..,N)\*[0,..,M) of int;

if B[j] != A[r,j] ->

18

;do j != M ->

17

j := 0

16

od

15

j := j+1

14

B[M-1 - j] := A[r,j]

13

;do j != M ->

14

print(is\_magic(A))

20

]

19

[0,3,-4,7]

18

[1,2,3,4],

17

A = [

16

# Test

15

return ismagic

21

i = i+1

13

j = j+1

12

ismagic = True

11

if sum\_row(i,A) == sum\_col(j,A):

10

while j != M and ismagic == False:

9

while i != N and ismagic == False:

ALGUNOS EJERCICIOS EN CGL

A

3

Entrada r: int;

2

proc is\_palin(

1

|[

anterior en las dos funciones anteriores.

sum\_rows

non\_palin\_rows

ejercicio

del

. Hará falta la función

función

Jhonny Lanzuisi

usaremos la

Para alterar las entradas no palíndromas de la matrix

tar las filas obtenidas anteriormente.

palin\_rows

será la función que se encargará de con-

Luego la función

tido contrario, y despues revisa si las dos filas (original y copia) son iguales.

una matriz es un palíndromo. Para esto hace una copia de la fila, pero en sen-

is\_palin

que chequea si una fila de

Primero definiremos un procedimiento

Iteración en matrices. Filas Capicúa

4.

49

55

B[i,j] = sumr

54

do j != M ->

53

sum\_rows(i,A,sumr)

52

if isp = false ->

51

is\_palin(i,A,isp)

50

;do i != N

od

isp := true

48

i,j,sumr := 0,0,0

47

var B: array [0,..,N)\*[0,..,M) of int;

46

var isp: bool;

45

var i,j,sumr: int;

44

0}

M

fi

7

-

-

66

]|

65

>>B

64

od

63

i := i+1

62

0

61

od

60

B[i,j] = A[i,j]

59

do j != M ->

58

j := 0

57

[] isp = true ->

56

var i: int;

is\_palin(i,A,isp)

32

;do i != N ->

31

isp := true

30

i,palinrows := 0,0

29

isp: bool;

28

var palinrows: int;

27

33

26

0}

M

0

{P: N

≥

∧

≥

25

func palin\_rows(A: array [0,..,N)\*[0,..,M) of int) -> int

ALGUNOS EJERCICIOS EN CGL

39

{P: N

≥

∧

≥

43

-> array [0,..,N)\*[0,..,M) of int

42

func non\_palin\_rows(A: array [0,..,N)\*[0,..,M) of int)

41

>>palinrows

40

od

Jhonny Lanzuisi

i = i+1

38

fi

37

skip

36

[] isp = false ->

35

palinrows := palinrows+1

34

if isp = true ->

modified version of

fdl-1.3.en.html

\*

is the GNU version of the original

program from Unix at Bell Labs.

is a

groff

roff

roff

https://www.gnu.org/licenses/

, which is considered the first text-formatting computer pro-

RUNOFF

gram, and was created for the CTSS. Currently,

is avialable in all major operating sys-

groff

tems and comes preinstalled in most GNU/Linux and BSDs.

-

-

8

main typeface is Palatino set 10/12.5, accompanied by Helvetica for sans-

ALGUNOS EJERCICIOS EN CGL

LICENSE & COLOPHON

This document was typeset using

\* with the mom macros in a Unix-

groff

like operating system. The editor used for editing the text was

The

emacs.

Jhonny Lanzuisi

serif and Courier for monospaced.

Copyright (C) 2021 Jhonny Lanzuisi.

Permission is granted to copy, distribute and/or

modify this document under the terms of the

GNU Free Documentation License, Version 1.3

or any later version published by the Free Soft-

ware Foundation; with no Invariant Sections, no

Front-Cover Texts, and no Back-Cover Texts. A

copy of the license can be found at

6

License & colophon

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

8

- i -

3. Iteración en matrices. Matriz Pre-mágica . . . . . . . . . . . . . . . . . . . . . . . . . .

Ejercicios de la guia

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1

1. Iteración en arreglos e invariantes

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1

1.1. Ahora en python

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2

2. Iteración en arreglos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2

**Contents**

2

3.1. Implementación de python

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4

4.

Iteración en matrices. Filas Capicúa

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .