**Primeiro teste de Algoritmos** e Estruturas de Dados

21 de Outubro de 2019

**14h10**m - 15h00m

**Responda a todas as perguntas no enunciado do teste. Justifique todas as suas respostas.**

Fórmulas:

Nome: N. Mec.:

**1=**

***n***

3.0

1: No seguinte código,

***n(n* + 1)**

#include <stdio.h>

**k=**

**1**

*non* + 1)*(2n* + 1)

int f(int x) { return x % 3; } int g(int x) { return x % 2; }

• Ž \* = (n(n+)

• logn

int main(void)

iel-1000, 1000)

TOUD Valores + int c = 0;

LUV*D va*lores -

Zero for(int i = -1000; i <= 1000; i++)

if(f(i) Il g(i))

• n! zn"e-n 2an

printf("i = %d\n", i); (i > 0*) &&* c++; C só *é* incrementado quando iso (porqu*e &*& -and)

**1*2***

printf("c = %d\n", c);

1.0 1.0 1.0

a) para que valores da variável i é avaliada a função g(x*)?* b) que valores de i são impressos*?* c) que valor de c é impresso?

Respostas:

(al A função g so é avaliach s a condição til for verdadeira. Ora, til

s*ó é v*er*dadei*ra *p*ara valores de i *que s*ão múltiplus de 3*, logo g(x*) é avaliada para os múltiples de 3 (-999,-996,...,0,..., 996 999).

(6) São impressos todos os valores de i que são múltiplus de 3 e pares

(-996 -990 - 984 ,...,0,..., 984, 990 996). *(C*) No intervalo d*e* i *l*it [-1*000*, 10*0*0)), existem 100*D v*alores positivos

(1... 1000). Desses, apnas 333 são múltiplos de 3 13,6,9, ..., 99a) e, destes, apanas 166 sau pares 16, 12, 18,... 996). Assim sendo, o valor impresso de c é 166.

Página 1 de 4 —

**Responda a todas as perguntas no enunciado do teste. Justifique todas as suas respostas.**

***3.0***

*rele*mbrar: growth rate: 1, log n*, vn, n, n* log n*, n?, n*3, 2", n! 2: Ordene as seguintes funções por ordem crescente de ritmo de crescimento. Responda nas duas colunas da direita da tabela. Na coluna da ordem, coloque o número 1 na função com o ritmo de crescimento menor (e, obviamente, coloque o número 5 na com o ritmo de crescimento maior).

Número da função

termo dominante

ordem

função n99 +1.1"

kh, K=1.1

1

**3**

***n***

2

***n!* 42n**

4on

1.2" + n2

n’ log n°9 + nI*n*

n23h

nk k = 2/3 L 1.20 4 K", K=1.2

24\_*, (k*? + k) I EM (K2) a nt k = 3 (X) n?.loginga) + n°3, 2 logo n 73 cresc*e m*ais rápido

4.0

3: Um programador inexperiente escreveu a seguinte função para copiar uma zona de memória com size bytes que começa no endereço src para uma outra zona de memória que começa no endereço dest.

void mem\_copy (char \*src, char \*dest, size\_t size)

**raciocínio (b)**

for(size\_t i = 0; i < size; i++)

*Zes*tado inicial) 0 1 2 3

0 *1 2* 3 4 5 *6* 7 8 9 lestado inicial) dest[i] = src[i];

1 2 3

5 4

LEO

6 3 4

1 2 4 5,9 Listy

5 6

DI 34 34 3.11=21

(1:

2 0 1 2 5 6 7 6 7

**raciocinio (a)**

un strat

DOWS 10 00

Responda às seguintes perguntas, considerando que para cada uma das duas primeiras o conteúdo inicial do array c é char c[10] = { 0,1,2,3,4,5,6*,7*,8,9 };

1.3

a) qual o conteúdo do array c depois de mem\_copy(&c [3] ,&c[5], 4); ter sido executado?

Resposta: 0

2 34 3*4* 34 «[1] [2] [3] [4] c[5] [6] [7] [8] [9]

[0]

1.3

b) qual o conteúdo do array c depois de mem\_copy(&c [5] , &c [3], 4); ter sido executado?

Resposta: 0

25678789 c[0] c[1] [2] [3] [4] [5] [6] [7] [8] [9]

1.4

c) num dos casos anteriores a cópia do conteúdo de parte do array não foi feita corretamente;

sugira uma maneira de corrigir este problema (não é obrigatório escrever código).

Resposta:

O caso que está errado é o da alínea (a). Uma maneira de corrigir o problema seria tornar a função recursiva em vez de iterativa, e guardar os valores originais de c numa variável temporária...

void mem\_copy(char\*src, char \*dest, size\_t size, size\_t counter)

if (size==counter) { return; }

*// v*ariável temp *//* guardar valor original na variável temp

int temp; temp = dest(counter]; dest[counter] = src[counter] dest[counter] = temp:

*// r*ecuperar valor original

-

Página 2 de 4 —

counter++; /*/* incrementar counter mem\_copy(dest(counter), src/counter), size, counter):

*/*/ chamada recursiva

**Responda a todas as perguntas no enunciado do teste. Justifique todas as suas respostas.**

3.0

4: A notação "big Oh” é usualmente usada para descrever a complexidade computacional do pior caso de um algoritmo. Porque?

Resposta (tente não exceder as 100 palavas): *A* notucao "big on" é ilm tipo de not*ac*cio matemáti*c*a q*ue ex*amina a taxa de crescimento de uma função lalgoritmo através do *l*imite s*up*rio*r deste* mesmo *crescim*ento. *Ora, e*m geral, *qu*anto maior for a taxa de crescimento de um algoritmo, maior é o seu tempo de execução (e pior é o caso em questão). Assim sendo, o limite superior dado pela notação big on" é usualmente representativo do pior caso de um algoritmo.

**2.0**

: Escreva o código de uma função que tenha uma complexidade computacional de CV Como alternativa, pode optar por escrever o código de uma função de tenha uma complexidade computacional de (log *n*). (Pode usar pseudo-código, se bem que uma função em C será mais valorizada.)

Resposta:

(sqrt(n))

**Explicação:**

void main(int n)

Se tivermos um k tal que s = 1+2+3+...+k = n, então o número de iterações é dado por k. Como 1+2+3+...+k = (k\*(k+1}}/2, então a equação (k\*(k+1)/2 = n dá-nos k.

int i = 0; int s = 0; while (s<=n)

(k\*(k+1)/2 = n <=> k\*(k+1) = 2\*n

<=> k^2 + k = 2\*n <=> k^2 + k-2 n = 0 <=> k = (-1 + sqrt(1+8n))/2, pela fórmula resolvente

S += 1; i += 1;

logo temos (sqrt(n))

(log(n))

**Explicação:**

void main(int n)

Se tivermos um k tal que s = e^k = n, então o número de iterações é dado por k. Então, a equação e^k = n dá-nos k.

int i = 0; int s = 0; while (s<=n)

enk = n <=> k = log(n)

logo temos (log(n))

S = ei i += 1;

Página 3 de 4 —

**Responda a todas as perguntas no enunciado do teste. Justifique todas as suas respostas.**

5.0

6: Para a seguinte função,

int f(int n)

int r = -1;

for(int i = -2; i <= n;i++) -

for(int j = i;j >= -3; j--) G

r += i - j; return r;

> n +3 ite*racoes*

n +4 itera*c*oes

2.0 2.0 1.0

a) quantas vezes é executada a linha r += i - j;? b) que valor é devolvido pela função? c) qual é a complexidade computacional da função?

Respostas: (al (n+3) (n +41 vezes (b) valor devol*vid*lo:

13-1 + Ź *(3 (*i-j))

JE-3

i=-*2*

c

somar de ia- 3 Du -3 ui é igua*l*

(C) Oln?)

Página 4 de 4 —