Anna Beatriz Dias Morais – 27470

Lista 1

1)

Struct celula {

Int chave;

Struct célula \*prox;

};

Struct célula \*busca int x, struct célula \*y) { contagem

Int achou = 0; 1

Struct célula \*p = y->prox; 1

While p!=NULL && !achou){ n+1

If p->chave==x) achou = 1; 2

P = p->prox; 1

}

If achou) return p; 1

Else return NULL; 1

}

Eficiência: 1+1+// n+1)\*2)+1+1 = 2 + 2n + 2 + 2 = 2n + 7

Complexidade: O/n)

Correção: O algoritmo não está correto, pois está faltando um “else” logo após a linha do “if” que encontra a chave desejada. Pois sem o “else” ele vai para a próxima linha do código e retorna a próxima chave depois da encontrada. Quando colocarmos o “else” ele ai retornar a chave encontrada correta.

2-)

Linha Contagem

1 1

2 n-1

3 n-2

4 1

5 1

6 n-2

7 1

8 1

9 1

Eficiência: 1+/n-1)\*[/n-2)/1+1)+/n-2)/1+1)+1] = 1+/n-1)\*[/n-2)\*2+/n-2)/\*2+1]

1+/n-1)\*[/2n-4)+/2n-4)+1] = 1 + 2n² - 4n – 2n + 4 + 2n² - 4n – 2n +4 + n -1

= 4n² - 11n + 8 Comportamento Assintótico: O/n²)

Correção: O algoritmo está correto, porque ele para. Porque no laço de repetição ele tem uma condição de parada que é atendida e um incremento.

Complexidade: A complexidade do algoritmo no melhor e no pior caso é a mesma, O/n2). Pois o algoritmo faz todas as verificações mesmo ele já estando ordenado.

3-)

#include <stdio.h>

#include <stdlib.h>

int main() { Contagem

int i, x; 2

int div = 0; 1

printf("Digite um número inteiro e positivo: "); 1

scanf("%d", &x); 1

for (i = 1; i <= x; i++) { n

if (x % i == 0) { 1

div++; 1

}

}

if (div == 2) 1

printf("O número %d é primo!", x); 1

else 1

printf("O número %d não é primo!", x); 1

return 0; 1

}

Correção: O algoritmo para corretamente e retorna corretamente se o número é primo ou não.

Eficiência: 2+1+1+1+2n+5 = 2n + 10

Comportamento Assintótico: O/n)