**SIN110 Algoritmos e Grafos - Exercício E1**

**Aluna:** Caroline Lopes Resek

**Matrícula:** 2017010113

1. A função abaixo recebe uma lista encadeada com cabeça e um inteiro x, e promete devolver p tal que p->chave==x ou NULL se tal p não existir. Analise a função verificando sua correção e eficiência.

\*Complexidade :

|  |  |  |
| --- | --- | --- |
| Linha | Algoritmo | Número de execuções |
| 1 | struct celula { | 0 |
| 2 | int chave; | 0 |
| 3 | struct celula \*prox; }; | 0 |
| 4 | struct celula \* busca {int x, struct celula \*y){ | - |
| 5 | int achou = 0; | 1 |
| 6 | struct celula \*p = y→prox; | 1 |
| 7 | while( p != NULL && !achou){ | n+1 |
| 8 | if (p→chave == x) achou = 1; | n |
| 9 | p = p→prox; } | n |
| 10 | if (achou) return p; | 1 |
| 11 | else return NULL; } | 1 |

T(n) = 3n + 5.

\*Correção:

1. O algoritmo para o laço (while) executado nas linhas 7-9 tem um número finito de execuções, uma vez que a variável auxiliar ‘p’(ponteiro) receberá o valor NULL ao atingir o final da lista de elementos armazenados.
2. O algoritmo verifica se a variável achou é diferente de 0, caso isso seja verdade, será retornado a variável auxiliar ‘p’, senão o valor retornado será nulo (NULL), indicando que o valor procurado não foi encontrado.
3. Prove a correção e determine o tempo de execução, no pior e no melhor caso, para o algoritmo de classificação abaixo:

\*Correção:

A cada ciclo do laço (para) da linha 2, as variáveis contidas do lado esquerdo para o direito do vetor serão comparadas e, caso necessário (A[j] > A[j+1]), as posições dos dois elementos comparados serão trocadas através do laço da linha 3. E, na linha 6, serão movidos para suas posições corretas os dados contidos do lado direito para o esquerdo do vetor. O número de execuções é finito, uma vez que são controlados pelas variáveis ‘i’ e ‘e’.

* No melhor caso, não será necessária a execução das linhas 5 e 8, uma vez que o vetor A já estaria ordenado.

\*Pior caso:

|  |  |  |
| --- | --- | --- |
| Linha | Algoritmo | Número de execuções |
| 1 | e← 1 | 1 |
| 2 | para i ← n-1 até e faça | (n/2) + 1 |
| 3 | para j ← e até i faça |  |
| 4 | se A[j] > A[j+1] |  |
| 5 | então troca(A[j],A[j+1]) |  |
| 6 | para j ← i até e+1 faça |  |
| 7 | se a[j-1]>A[j] |  |
| 8 | então troca(A[j-1],A[j]) |  |
| 9 | e ← e+1 | n/2 |

T(n) = 1 + [n/2] + 1 + + + + + + + n/2 = 2 + 2 (n/2) + [[(n + 2) + n + n + (n + 3) + (n + 1) + (n + 1)] (n/2)]/2 = 2 + n +[(6n+7)(n/2)]/2 = 2 + n + + 7n/4

F(n) = 2 + + 11n/4

Comportamento Assintótico: O()

\*Melhor caso:

|  |  |  |
| --- | --- | --- |
| Linha | Algoritmo | Número de execuções |
| 1 | e← 1 | 1 |
| 2 | para i ← n-1 até e faça | (n/2) + 1 |
| 3 | para j ← e até i faça |  |
| 4 | se A[j] > A[j+1] |  |
| 5 | então troca(A[j],A[j+1]) | 0 |
| 6 | para j ← i até e+1 faça |  |
| 7 | se a[j-1]>A[j] |  |
| 8 | então troca(A[j-1],A[j]) | 0 |
| 9 | e ← e+1 | n/2 |

T(n) = 1 + [n/2] + 1 + + + + + n/2 = 2 + n + [[(n + 2) + n + (n + 3) + (n + 1)] (n/2)]/2 = 2 + n + [(4n+6)(n/2)]/2 = 2 + n +

F(n) = 2 + + 5n/2

Comportamento Assintótico: O()

1. Projete um algoritmo, prove sua correção e determine a complexidade de tempo e comportamento assintótico, para o problema: *Dado um inteiro N, verifique se N é primo.*

\*Complexidade :

|  |  |  |
| --- | --- | --- |
| Linha | Algoritmo | Número de execuções |
| 1 | int verificaNroPrimo(int n){ | - |
| 2 | int i, cont=0; | 1 |
| 3 | for (i = 1; i <= n; i++){ | n+1 |
| 4 | if (n % i == 0) cont ++; } | n |
| 5 | if (cont == 2) return 1; | 1 |
| 6 | else return 0; } | 0 ou 1 |

Melhor caso(número é primo): T(n) = 1 +(n + 1) + n + 1 = 2n+4

Pior caso(número não é primo): T(n) = 1 +(n + 1) + n + 1 + 1 = 2n+5

Comportamento Assintótico: O(n)

\*Correção:

1. O algoritmo verifica, através do laço nas linhas 3-4, o número de divisores que a variável ‘n’ possui, os quais serão armazenados na variável ‘cont’(inicializada com 0 na linha 2).
2. Ao sair do laço é verificado na linha 5 se a variável ‘cont’ possui o valor 2, uma vez que isso significa que o número é primo, e retorna 1, senão, retorna 0.

* No pior caso o número não será primo e, portanto, as linhas 5 e 6 serão executadas, caso contrário, apenas a linha 5 será executada.