Estrutura de Dados

José Gildásio Freitas do Ó - 473901

Maio 2023

1. Apresente o pseudocódigo de uma função Tamanho(v) que recebe como entrada o nó cabeça v de uma lista encadeada e retorna o tamanho da lista (ou seja, a quantidade de nós na lista). Qual a complexidade dessa função?

   - **Resposta:**
     - Algoritmo: Tamanho(V[])
     - Entrada: vetor
     - Saida: tamanho do vetor
     - Complexidade: O(n)

```
1–  contador ← 0
2–  p ← v
3–  enquanto ( p → prox ≠ λ ) faça
4–  |    contador++
5–  |    p ← p → prox
6–  return contador
```

2. Apresente o pseudocódigo de uma função Concatenar(v1 ,v2) que recebe como entrada os nós cabeça v1 e v2 de duas listas encadeadas e concatena a primeira lista com a segunda, retornando a nova lista concatenada. Qual a complexidade dessa função?

   - **Resposta:**
     - Algoritmo: Concatena(L1, L2)
     - Entrada: Lista 1 e Lista 2
     - Saida: Lista 1 concatenada com lista 2
     - Complexidade: O(n1)

```
1–  p ← L1
2–  enquanto (p → prox ≠ λ ) faça
3–  |    p ← p → prox
4–  |    p → prox ← L2 → prox
5–  desaloca(L2)
6–  return L1
```

3. Apresente o pseudocódigo de uma função ListarInverso(v) que recebe como entrada o nó cabeça v de uma lista encadeada e imprime as chaves de todos os nós da lista na ordem inversa à ordem de ocorrência destes nós. Qual a complexidade dessa função?

- **Resposta:**
  - Algoritmo: ListarInverso(L)
  - Entrada: Lista L
  - Saida: Lista L invertida
  - Complexidade: O(n)

    1– **se** L = λ **então** return
    2– ListarInverso(L→prox)
    3– Imprime(L→chave)
    4– ListarInverso(L)
    5– ListarInverso(L→prox)

4.

5.

Q4)

```
Void RemoveTodos(int L[], int& tam, int x)
    int tamAtual = 0
    para int i = 0 até i < tam faça
        se L[i] != x então
            L[tamAtual] = L[i]
            tamAtual++
    i++
    tam = tamAtual
```

O(n)

Q5)

```
Void RemoveTodos(Nó* inicial, int x)
    Nó atual = inicial
    Nó anterior = null
    enquanto (atual != null) faça
        se atual->chave == x então
            se atual != null então
                atual->prox = atual->prox
            senão
                inicial = atual->prox
            Nó temp = atual
            atual = atual->prox
            delete temp
        senão
            anterior = atual
            atual = atual->prox
```

O(n)

Q6-

```
void incluirOrdenado ( Nó& inicial, int x )
    Nó nova = new Node (x)
    se inicial == null ou x < inicial -> chave então
    |   nova -> prox = inicial
    |   inicial = nova
    senão
    |   Nó anterior = inicial
    |   enquanto (anterior -> prox != null && x >= anterior -> prox ->
    |   chave)
    |   |  anterior = anterior -> prox
    |   nova -> prox = anterior -> prox
    |   anterior -> prox = nova
```

$O(n)$