Lista de exercícios 2

Aluno: Pedro Lucas Silva Haga Torres
Matrícula: 16/0141575 Matéria: Estrutura de Dados Turma: B

1. Apresente uma definição de algoritmo.

"Informalmente, um algoritmo é qualquer processo computacional bem definido que recebe um conjunto de valores como *entrada* e produz algum valor, ou conjunto de valores, como *saída*. Um algoritmo é, portanto, uma sequência de passos computacionais que transforma a *entrada* na *saída*". (CORMEN, et al. 2009, tradução minha)

2. Cite quatro características de um algoritmo.

1. Corretude;
2. Não ambíguo;
3. Possui fim;
4. Independência de qualquer linguagem computacional.

3. Quais as diferenças entre ordenação em bolha, *merge sort* e *quick sort*?

O algoritmo *bubble sort* varre todo o vetor comparando pares de elementos adjacentes (elemento 0 e 1, 1 e 2, 2 e 3, etc.) e troca os elementos de acordo com a regra de ordenação. Como o último elemento sempre será posicionado corretamente, a cada varredura o algoritmo percorre n-1 elementos, até a sua condição de parada.

O *merge sort* divide a entrada em n listas menores, até que cada uma seja uma lista unitária. A partir dessas listas unitárias, o algoritmo funde-as (*merge*) ordenando seus elementos, em listas de 2, 4, 8, … , elementos, até chegar a lista original, que estará ordenada.

No algoritmo *quick sort* escolhe-se um pivô e posiciona-se ele no centro do vetor, os elementos maiores que o pivô devem ficar a sua direita, enquanto os menores ficam à esquerda. Pode-se repetir essa operação quantas vezes for conveniente, para, então, ordenar-se os subconjuntos divididos previamente.

4. Dado um vetor de tamanho n, elabore um pseudocódigo para encontrar um elemento utilizando busca binária. Utiliza uma chamada de rotina para garantir que a entrada esteja ordenada. Qual a complexidade do seu algoritmo? Compare o resultado com um algoritmo de busca linear. Se fosse necessário ordenar sempre a entrada, qual seria mais vantajoso?

```
Função buscBin
  Entrada: vetor[], n, procurado;
  Variaveis inteiras: i, j, comp;
  Saída: i ou -1 em caso de erro.

  merge_sort(vetor)

  i = 0;
  j = n-1;

  Enquanto (i <= j)
      comp = (i+j)/2;
      Se (vetor[comp] == procurado)
          Retorne (comp)
      Caso contrário, se (procurado < vetor[comp])
           j = m - 1
```

```
      Caso contrário
          i = m+1
  Retorne(-1)
Fim do pseudocódigo
```

A complexidade do meu algoritmo é $O(n * \log(n) + \log(n)) = O(n * \log(n))$. A complexidade do algoritmo de busca linear é $O(n) < O(n * \log(n))$, logo, usar apenas um algoritmo de busca linear seria mais vantajoso. Porém, supondo a necessidade de sempre ordenar a entrada, o algoritmo de busca binária é mais vantajoso, pois a comparação ficaria entre $O(\log(n)$ e $O(n)$; e $O(\log(n) < O(n)$.

5. Suponha que você tenha um sistema que receba tarefas via clientes de vários locais do mundo e que um processo as realize em momento agendados, na ordem de chegada das mesmas. Qual seria a estrutura de dados mais adequada para esse sistema controlar as tarefas? Qual a complexidade dela para inserir e remover tarefas? Justifique.

A estrutura de dados mais adequada seria uma fila, pois ela segue o parâmetro FIFO (primeiro a entrar, primeiro a sair). A complexidade seria $O(1)$ porque tanto a cabeça, quanto a cauda da lista, ficam guardadas em parâmetros controlados pelo programa.

6. Elabore um pseudocódigo para inserir um elemento no final de uma lista.

```
Função inserir
  Entrada: ponteiro para o primeiro elemento da lista, elemento.
  Variáveis utilizadas: *p

  p = ponteiro;
  Enquanto (p != null)
      Se (p->ponteiro == NULL)
          Saia do laço;
      Caso contrário
          p = p->ponteiro;

  Aloque espaço na memória para a estrutura do novo elemento;

  p->ponteiro = &novo;
  p = p->ponteiro;

  p->elem = elemento;
  p->ponteiro = NULL;
Fim do pseudocódigo
```

7. Considerando que uma lista possui apenas o ponteiro para o primeiro elemento, qual a complexidade, no pior caso, para inserir um novo elemento? Justifique.

A complexidade, nesse caso, seria $O(n)$, pois seria necessário varrer todos os elementos da lista para chegar ao último e encadeá-lo.

8. Qual a diferença entre pilha e fila?

O critério de saída dos dados, pilha segue o parâmetro LIFO (último a entrar, primeiro a sair) e fila o critério FIFO (primeiro a entrar, primeiro a sair). A fila também exige uma variável a mais, pois é necessário guardar o primeiro elemento da fila, o que não é necessário no caso de uma pilha.