Instituto Infnet
Disciplina: Algoritmos Avançados
Aluno(a): Nizshime Samara Cotta Mathildes Scorzello

QUESTÃO 4. # Defina problemas P, NP e NP-completos.

Classificação:
Na programação, a classificação de problemas em **P, NP e NP-Completos** é usada para avaliar os recursos necessários para a resolução de problemas computacionais:

- ***Problemas em P*** são aqueles que podemos resolver eficazmente
- ***Problemas NP*** são desafiadores, mas podemos verificar soluções rapidamente.
- ***Problemas NP-completos*** são os mais difíceis e desafiadores de todos.

A classificação é feita com base em evidências teóricas, e a questão de se P é igual a NP (ou seja, se todos os problemas em NP podem ser resolvidos em tempo polinomial) é uma das questões mais importantes e não resolvidas na ciência da computação.

Descrição de cada uma das classificações:

- **P (Problemas Polinomiais):**
  Problemas em P são aqueles que podem ser resolvidos em tempo polinomial, o que significa que o tempo de execução do algoritmo para resolver o problema é limitado por uma função polinomial do tamanho da entrada.
  Exemplo: Algoritmo de Dijkstra para Caminhos Mais Curtos, este algoritmo encontra o caminho mais curto entre dois vértices em um grafo com arestas de peso não negativo em tempo polinomial.

- **Problemas NP (Não-Determinísticos Polinomiais):**
  Problemas em NP são aqueles para os quais, dado um candidato a solução, você pode verificar a validade da solução em tempo polinomial.
  No entanto, encontrar a solução em si pode não ser tão eficiente.
  Em programação, problemas em NP geralmente envolvem encontrar soluções que podem ser verificadas rapidamente.
  Exemplo: O problema de encontrar um conjunto de números que some a um valor específico (Subset Sum). O **Subset-Sum** é quando dado um conjunto de números e um valor, verificar se existe um subconjunto cuja soma seja igual ao valor é rápido, mas encontrar esse subconjunto é mais desafiador.

- **Problemas NP-Completos (Não-Determinísticos Polinomiais Completos):**
  Problemas NP-completos são uma classe especial de problemas em NP que são considerados entre os mais difíceis. Não se sabe se é possível encontrar soluções eficientes (tempo polinomial) para eles.
  Em programação, esses problemas são notoriamente difíceis de resolver e muitas vezes requerem abordagens heurísticas.
  Exemplo: O **Problema do Caixeiro Viajante**, onde você deve encontrar a rota mais curta que visite um conjunto de cidades exatamente uma vez e retorne ao ponto de partida.

Não existe um algoritmo conhecido que resolva todos os casos em tempo polinomial.

QUESTÃO 5. # Explique o funcionamento dos algoritmos abaixo.

- **Bubblesort:**
  O Bubblesort é um algoritmo de ordenação, o funcionamento deste algorítimo consiste em que o início da lista é avaliado e comparando o valor da primeira posição com o próximo. Se o primeiro for maior que o próximo, o algoritmo os troca de lugar.
  Isto ocorrerá para todos os pares de números na lista até que nenhuma troca seja necessária durante uma passagem avaliativa completa (repete-se esse processo até que a lista esteja completamente ordenada).
  Embora o Bubblesort seja intuitivo e fácil de implementar, não é o método mais eficiente de ordenação, especialmente para listas grandes.

- **Insertion Sort:**
  O Insertion Sort é um algoritmo de ordenação, ele trabalha inserindo cada elemento no local apropriado em relação aos elementos já ordenados.
  O método Insertion Sort é eficiente para listas pequenas ou que já estejam parcialmente ordenadas, devido à sua abordagem simples e ao fato de não exigir muitas operações em tais condições.

- **Mergesort:**
  O Merge Sort é um algoritmo de ordenação eficiente e geralmente indicado para listas grandes onde a estabilidade é necessária e o uso adicional de memória não é uma preocupação por conta de sua recursividade e necessidade de armazenamento temporário devido a sua abordagem de “dividir para conquistar”.
  Ele trabalha dividindo a lista em sub-listas menores (abordagem "dividir para conquistar"), ordenando-as recursivamente e em seguida, combinando-as para produzir a lista ordenada final.
  Este algorítimo é mais eficiente do que o Bubblesort e o Insertion Sort para listas grandes.

Em resumo, a respeito das três técnicas de Algorítimos:
Cada um desses algoritmos facilita a ordenação de dados de acordo com regras definidas, possuindo particularidades e níveis de eficiência distintos. A escolha do algoritmo mais adequado dependerá das características dos dados a serem ordenados e dos requisitos de eficiência para a tarefa em questão.