Com base nas informações das fontes e na nossa conversa anterior, seguem algumas perguntas que podem cair em uma prova sobre os algoritmos mencionados:

**Merge Sort:**

* **Verdadeiro ou Falso:** O Merge Sort é um exemplo da técnica de Divisão e Conquista. Justifique sua resposta.
* **Múltipla Escolha:** Qual das seguintes opções descreve corretamente a etapa de "conquista" no algoritmo Merge Sort, de acordo com a técnica de Divisão e Conquista?   
    
  a) Dividir o vetor em duas metades.  
  b) Resolver cada subproblema recursivamente.   
  c) Combinar as duas metades ordenadas em um único vetor ordenado.   
  d) Comparar elementos adjacentes e trocá-los se estiverem fora de ordem.
* **Pergunta Curta:** Explique brevemente os três passos principais da técnica de Divisão e Conquista aplicados ao Merge Sort.
* **Pergunta Curta:** Quais são as vantagens do Merge Sort?
* **Verdadeiro ou Falso:** A fase de divisão no Merge Sort visa criar subproblemas que precisam ser resolvidos iterativamente. Justifique sua resposta

**Multiplicação de Inteiros Grandes (Relacionado ao Algoritmo de Karatsuba):**

* **Verdadeiro ou Falso:** A multiplicação clássica de dois números inteiros grandes de *n* dígitos possui uma complexidade de tempo menor do que a abordagem por Divisão e Conquista.Justifique sua resposta.
* **Múltipla Escolha:** Na abordagem por Divisão e Conquista para multiplicação de inteiros grandes descrita, qual é o primeiro passo crucial?

a) Realizar multiplicações dígito a dígito.   
b) Dividir cada número em dois números com a metade da quantidade de dígitos.   
c) Deslocar os números para a direita.   
d) Realizar adições dos números originais.

* **Pergunta Curta:** De acordo com a fonte [#0-divConq.pdf], quais são as três etapas para multiplicar bignums usando a técnica de Divisão e Conquista?
* **Pergunta Curta:** Explique brevemente como a multiplicação por (10^n) é interpretada na abordagem de Divisão e Conquista para multiplicação de inteiros grandes.

**Problema da Mochila (Knapsack Problem):**

* **Verdadeiro ou Falso:** A solução para o Problema da Mochila utilizando Programação Dinâmica envolve testar todas as combinações possíveis de itens. Justifique sua resposta com base nas fontes [#1-ProgDinâmica-parte1.pdf] e [#1-ProgDinâmica-parte2.pdf].
* **Múltipla Escolha:** No algoritmo de Programação Dinâmica para o Problema da Mochila, o que representa a entrada maxTab[i][j]?

a) O peso total dos itens na mochila.   
b) O valor do i-ésimo item.   
c) O valor máximo que pode ser obtido com os primeiros *i* itens e uma capacidade de mochila *j*.   
d) Se o i-ésimo item está ou não na mochila.

* **Pergunta Curta:** Descreva brevemente como a matriz maxTab é preenchida no algoritmo de Programação Dinâmica para o Problema da Mochila, conforme explicado nas fontes [#1-ProgDinâmica-parte1.pdf] e [#1-ProgDinâmica-parte2.pdf].
* **Pergunta Curta:** Quais são as duas opções consideradas ao decidir o valor de maxTab[i][j] se o *i*-ésimo item cabe na mochila de capacidade *j*?

**Distância de Edição (Edit Distance):**

* **Verdadeiro ou Falso:** O algoritmo de Distância de Edição com Programação Dinâmica calcula o custo de transformar uma string em outra resolvendo recursivamente subproblemas sem armazenar resultados intermediários. Justifique sua resposta com base na fonte [#1-ProgDinâmica-parte5.pdf].
* **Múltipla Escolha:** No algoritmo de Distância de Edição utilizando Programação Dinâmica, o que representa a posição matriz[i][j] na tabela? a) O custo da i-ésima operação de edição.   
  b) O custo de transformar a string A inteira na string B inteira.   
  c) O custo de transformar os *i* primeiros caracteres de A nos *j* primeiros caracteres de B.   
  d) O número de operações de edição realizadas até o momento.
* **Pergunta Curta:** Quais são as três operações básicas consideradas no cálculo da Distância de Edição e como seus custos são utilizados no algoritmo apresentado na fonte [#1-ProgDinâmica-parte5.pdf]?
* **Pergunta Curta:** Explique como a primeira linha e a primeira coluna da matriz são inicializadas no algoritmo de Distância de Edição utilizando Programação Dinâmica.

**Maior Subsequência Comum (Longest Common Subsequence - LCS):**

**1. Conceito**

**O que é a LCS entre duas strings?**  
*(Explique com suas palavras.)*

**2. Ordem dos caracteres**

**Na LCS, os caracteres precisam ser consecutivos nas strings?**  
*(Justifique sua resposta.)*

**3. Exemplo**

**Qual é a LCS das strings "ABCDEF" e "AEBDF"?**  
*(Escreva a subsequência encontrada e seu comprimento.)*

**4. Aplicação prática**

**Cite uma aplicação real onde o algoritmo de LCS é utilizado.**  
*(Dica: pode ser na computação ou na biologia.)*

**5. Ideia principal do algoritmo**

**Explique o que acontece no algoritmo de LCS quando os últimos caracteres comparados são iguais.**

**6. Alternativa no algoritmo**

**E se os últimos caracteres das duas strings forem diferentes?  
O que o algoritmo faz nesse caso?**

**7. Programação dinâmica**

**Por que usamos uma tabela (matriz dp) para resolver o problema da LCS usando programação dinâmica?**

**8. Complexidade**

**Qual é a complexidade de tempo e espaço do algoritmo de LCS usando programação dinâmica?  
Explique o que significam m e n nessa análise.**

**9. Código**

**No algoritmo em Java apresentado, o que a linha dp[i][j] = 1 + dp[i-1][j-1]; representa?**

**10. Última célula da matriz**

**Por que o valor dp[m][n] no final da matriz é a resposta do problema?**