Atividade I - 4 pontos

----------IMPORTANTE----------

Caso identifique o uso de IA para as perguntas dissertativas irei zerar a resposta e irei retirar metade dos pontos conseguidos no geral. As perguntas dissertativas devem seguir os conceitos aprendido em sala de aula e não podem ser inventados, apesar disso, não há uma resposta verdadeiramente correta, mas a que faz mais sentido no contexto aprendido.

O e-mail do participante (**izaell.oficial@uni9.edu.br**) foi registrado durante o envio deste formulário.

Nome *

IZAEL ALVES DA SILVA

RA *

922114939

Perguntas sobre algoritmos de ordenação

Algoritmo de **busca de ordenação** refere-se a uma classe de algoritmos usados para reorganizar os elementos de uma lista ou array em uma ordem específica, geralmente crescente ou decrescente. Os algoritmos de ordenação são essenciais na ciência da computação, uma vez que muitos outros algoritmos dependem de dados ordenados para operar eficientemente

Escreva com suas palavras em qual cenário o Bubble Sort tem seu melhor desempenho? (Resposta curta) *

Quando os dados estão poucos desordenados ou minimamente desordenados

Imagine que você está fazendo um novo método de ordenação para uma linguagem criada na disciplina de compiladores. Essa linguagem foi feita para ser de uso de propósito geral, por exemplo, Java, C# ou Python. Como a linguagem é nova e pode ser usada para computar "qualquer coisa" você, a priori, não sabe qual contexto ela será mais utilizada. Por exemplo, você não tem como saber se os dados a serem ordenados nessa linguagem será considerado grande, pequeno, pré-ordenado ou totalmente desordenado entre outros. Pensando como um cientista da computação, escreva **com suas palavras** qual seria a sua abordagem / estratégia para criar uma função sort() padrão para a nova linguagem de modo que otimize o processo de ordenação? *

Já li algo sobre isso, e eu usaria aquela estratégia de combinar vários algoritmos, dependendo dos dados e quantidade. Acho que usar um algoritmo de acordo com as características do vetor ou lista seria mais eficiente e rápido. Por exemplo, para listas ou vetores pequenos, a função SORT() poderia usar um algoritmo simples como o Insertion Sort. Já para listas ou vetores grandes e desordenados, a funçaõ poderia optar por algoritmos mais complexos como o QuikSort ou MergeSort. Durante a ordenação, a funcção SORT() poderia até mudar de estratégia, adaptando-se às características da lista para obter o melhor desempenho possível.

Como o Bubble Sort se compara ao Quick Sort em termos de complexidade de tempo? *

- ○ Ambos têm o mesmo tempo de execução no médio caso.
- ○ O Bubble Sort é mais eficiente que o Quick Sort em listas grandes e desordenadas.
- ◉ O Quick Sort tem um pior caso O(n^2), mas é geralmente mais rápido que o Bubble Sort devido à sua complexidade média.
- ○ Ambos os algoritmos têm o mesmo desempenho no melhor médio.
- ○ O Bubble Sort usa mais memória do que o Quick Sort.

Qual é a diferença entre o pior caso do Insertion Sort e seu melhor caso? *

- (●) O pior caso ocorre quando a lista está em ordem decrescente, e o melhor caso ocorre quando a lista está ordenada.
- ( ) O pior caso ocorre quando a lista contém números negativos, e o melhor caso quando contém números positivos.
- ( ) O pior caso ocorre quando a lista contém números iguais, e o melhor caso quando contém números diferentes.
- ( ) O pior caso e o melhor caso são sempre o mesmo, independentemente da lista.
- ( ) O pior caso ocorre quando a lista contém strings e o melhor caso quando contém inteiros.

Por que o Selection Sort é considerado ineficiente em listas de grandes volumes de dados? *

- ( ) Porque ele exige uma grande quantidade de memória adicional.
- ( ) Porque ele faz mais comparações e trocas do que o Merge Sort.
- (●) Porque ele sempre realiza O(n^2) operações, mesmo quando a lista já está ordenada.
- ( ) Porque ele usa a técnica de divisão e conquista, que é ineficiente em grandes listas.
- ( ) Porque ele depende de chamadas recursivas profundas.

Por que o Insertion Sort é frequentemente usado em combinação com algoritmos como o Merge Sort ou Quick Sort? *

- ( ) Porque o Insertion Sort tem uma melhor complexidade espacial.
- ( ) Porque o Insertion Sort é mais eficiente para grandes listas.
- (•) Porque o Insertion Sort é mais eficiente para listas pequenas ou quase ordenadas.
- ( ) Porque o Insertion Sort é mais fácil de implementar recursivamente.
- ( ) Porque o Insertion Sort requer menos memória auxiliar.

Qual é a principal técnica utilizada no algoritmo Merge Sort para ordenar uma lista? *

- ( ) Busca binária
- ( ) Divisão e conquista
- ( ) Ordenação por troca
- ( ) Inserção direta
- (•) Ordenação por comparação de adjacentes

O que torna o Merge Sort uma boa escolha em sistemas com grandes quantidades de dados que não podem ser armazenados inteiramente na memória? *

- ( ) Usa pouca memória adicional.
- ( ) Realiza poucas comparações entre os elementos.
- (•) Divide a lista em partes menores e processa essas partes separadamente, o que é eficiente com sistemas de armazenamento externo.
- ( ) Pode ser implementado de forma não-recursiva, o que economiza espaço.
- ( ) Evita completamente o uso de memória auxiliar.

Qual é uma vantagem do Merge Sort sobre o Quick Sort em termos de complexidade de tempo? *

- ( ) O Merge Sort tem uma complexidade média menor.
- (•) O Merge Sort tem uma complexidade no pior caso melhor que O(n^2)
- ( ) O Merge Sort é um algoritmo in-place, enquanto o Quick Sort não é.
- ( ) O Merge Sort é mais eficiente para listas pequenas.
- ( ) O Merge Sort tem melhor desempenho para listas quase ordenadas.

Quando o Quick Sort pode se comportar pior que outros algoritmos de ordenação, como o Merge Sort? *

- ( ) Quando a lista é muito grande.
- (•) Quando a lista já está ordenada ou quase ordenada.
- ( ) Quando a lista contém muitos elementos duplicados.
- ( ) Quando a lista é composta por números flutuantes.
- ( ) Quando a lista é muito pequena.

O que acontece com o desempenho do Quick Sort se a lista de entrada for **completamente ordenada** e o pivô escolhido for sempre o primeiro elemento? *

- ( ) O desempenho será otimizado para O(n log n) devido à lista já estar ordenada
- (•) O desempenho será O(n^2) porque a lista será dividida de forma desigual em cada iteração
- ( ) O algoritmo irá falhar, pois não consegue lidar com listas já ordenadas
- ( ) O desempenho será O(n) pois o Quick Sort identificará que a lista está ordenada e irá parar
- ( ) O algoritmo se tornará instável, trocando elementos de forma desnecessária, mas a complexidade continuará sento O(n log n)

Perguntas sobre busca

A busca linear percorre cada elemento de uma lista sequencialmente até encontrar o valor desejado, sendo eficiente para pequenas listas ou dados desordenados. A busca binária divide repetidamente uma lista ordenada ao meio, descartando metade a cada passo, sendo mais rápida, mas requer ordem prévia.

Qual é a complexidade de tempo da busca linear no **pior caso** e em que cenário ela ocorre? Explique o porquê *

No pior caso seria (O de N), quando o elemento não é encontrado ou está no final. O algoritmo verifica todos os valores da lista antes de encontrar o valor desejado, se ele estiver no final, ou antes de saber que ele não está na lista, quando o valor não é encontrado.

Imagine que você foi selecionado para realizar uma pequena pesquisa científica acadêmica na área de estruturas de dados, especificamente no subcampo de pesquisa e ordenação em inteligência artificial. Sem se preocupar com os detalhes, o seu objetivo é buscar as palavras que melhor se encaixem na formação de uma sentença. No seu cenário, todas as palavras estão em ordem alfabética e contêm todas as palavras que serão usadas. Desenvolva uma abordagem/estratégia que combine busca linear e busca binária para encontrar a próxima palavra, com o menor número de passos possíveis. *

Exemplo de aplicação: Dado a frase: "A disciplina de estrutura de _____"
Nesse caso o algoritmo saberá que a próxima palavra deverá ser **dados**, porém não sabe qual o seu índice ou se ela existe na lista, você terá que desenvolver um estratégia para selecionar a palavra "dados" na lista ordenada.

Se imaginarmos um dicionário, por exemplo, onde ele esteja em ordem alfabeta, lógico. Para encontrar uma palavra começamos do inicio, certo? Não. A gente abre mais ou menos na metade do dicionário, eu pelo menos faço assim. Olho se a palavra que procuro está antes ou depois daquela pagina, se estiver depois eu descarto a primeira parte. e repito o processo na segunda parte dividindo-a novamente no meio, isso é uma busca binaria. Mas há um problema. E se houver palavras semelhantes a que estou procurando? tipo se eu estiver procurando "dados" e houver "dado"? Eu não preciso recomeçar a busca. Eu posso continuar procurando palavras que começam com "dado", correto? Pois é, essa é a busca linear direcionada. Então minha estratégia seria achar uma posição aproximada usando a busca binaria para descobrir mais ou menos onde a palavras buscada está. E com a busca linear direcionada acharia com mais afinco a palavra certa. Esta estratégia é rápida, precisa e flexível.

Em qual cenário a busca linear pode ser mais eficiente que a busca binária? *

- ○ Quando a lista está ordenada em ordem crescente.
- ○ Quando a lista contém um número muito grande de elementos.
- ◉ Quando a lista não está ordenada e os elementos não têm uma estrutura específica.
- ○ Quando o elemento buscado está no meio da lista.
- ○ Quando a lista contém apenas números inteiros.

Se um algoritmo de busca linear é aplicado a uma lista com **elementos duplicados**, qual é o comportamento esperado ao buscar um elemento que aparece várias vezes? *

- ◉ O algoritmo retorna a primeira ocorrência do elemento.
- ○ O algoritmo retorna a última ocorrência do elemento.
- ○ O algoritmo retorna todas as ocorrências do elemento.
- ○ O algoritmo retorna um erro, pois elementos duplicados não podem ser processados.
- ○ O algoritmo retorna a posição de um elemento aleatório.

Se um algoritmo de busca linear é usado para buscar um elemento em uma lista muito grande, que estratégia pode ser adotada para melhorar a eficiência sem mudar o algoritmo? *

- ( ) Ordenar a lista antes de aplicar a busca.
- ( ) Utilizar uma abordagem de busca binária.
- (•) Executar a busca em paralelo em múltiplas threads.
- ( ) Fazer a busca em uma lista de índices em vez da lista principal.
- ( ) Dividir a lista em sublistas menores e aplicar a busca linear em cada uma.

Em um sistema que utiliza uma lista de usuários identificados por IDs, se um novo usuário for adicionado com um ID que já existe, qual comportamento pode ser esperado de um algoritmo de busca linear que verifica a duplicação antes da adição? *

- ( ) O algoritmo sempre adicionará o novo ID, ignorando duplicatas.
- (•) O algoritmo identificará a duplicata e evitará a adição do novo ID.
- ( ) O algoritmo resultará em um loop infinito se não for tratado corretamente.
- ( ) O algoritmo retornará um erro, pois IDs duplicados não são permitidos.
- ( ) O algoritmo alterará o ID do novo usuário para evitar a duplicação.

Qual é a condição necessária para que a busca binária funcione corretamente em uma lista? *

- ( ) A lista deve estar desordenada.
- ( ) A lista deve ser composta apenas por números inteiros.
- ( ) A lista deve estar ordenada.
- (•) A lista deve ser composta por elementos únicos.
- ( ) A lista deve ser pequena (menos de 10 elementos).

Durante a execução da busca binária, como o algoritmo determina a parte da lista onde deve continuar a busca? *

- ( ) Ele sempre escolhe o elemento mais à esquerda.
- (•) Ele compara o elemento buscado com o elemento do meio da lista e decide a partir daí.
- ( ) Ele utiliza um índice aleatório.
- ( ) Ele reordena a lista a cada iteração.
- ( ) Ele descarta a lista e inicia uma nova busca.

Suponha que você esteja desenvolvendo um sistema de busca para uma plataforma de vídeos, onde os vídeos são armazenados em uma lista ordenada por número de visualizações. Como a busca binária pode ser aplicada aqui? *

- ○ Você deve sempre percorrer todos os vídeos até encontrar o desejado.
- ◉ Você pode utilizar a busca binária para encontrar rapidamente um vídeo com um número específico de visualizações.
- ○ Você deve reordenar os vídeos com base em suas tags antes de procurar.
- ○ Você deve criar uma nova lista apenas com os vídeos mais populares.
- ○ Você pode apenas usar a busca linear, já que a lista é muito grande.

Em um site de compras online, você deseja encontrar um produto específico em uma lista de produtos ordenados por preço. Qual é a maneira mais eficiente de utilizar a busca binária para realizar essa tarefa? *

- ○ Filtrar os produtos por categoria e então usar a busca linear.
- ○ Utilizar a busca binária para encontrar o produto pelo seu preço.
- ◉ Percorrer todos os produtos até encontrar o desejado.
- ○ Reordenar os produtos por nome antes de procurar.
- ○ Fazer uma busca por palavras-chave no site.