IMD0029 - LUCAS_ANIZIO - 20240065713 - ATIVIDADEBUSCAS_EDB.PDF

Questões teóricas - Responda todas as questões utilizando as suas palavras

Questão 6.1 — Complexidade de Algoritmos de Busca.
Tema: Busca sequencial vs. busca binária

Enunciado: Compare os algoritmos de busca sequencial e busca binária em termos de complexidade, aplicabilidade e limitações. Discuta:

● **Em quais contextos cada um é mais apropriado?**
Busca binária é mais apropriado quando temos grandes conjuntos ordenados. A busca sequencial é mais apropriada quando temos um pequeno conjunto de dados e eles não são estruturados.
● **Quais são os pré-requisitos para aplicar a busca binária corretamente?**
Os pré-requisitos são: estar com todos os dados ordenados, pois se não tiver ordenação, não existe garantia de que os dados vão ser corretos. A estrutura deve permitir o acesso rápido aos índices e todos os elementos precisam ser comparáveis.
● **Como a ordenação influencia a escolha do algoritmo de busca? Ilustre seu argumento com um exemplo concreto de uso para cada tipo de busca.**

A busca sequencial percorre sempre em todos os elementos do vetor de forma sequencial, ou seja, ele percorre de um por um até encontrar o dado alvo. Logo, o vetor pode estar ou não estar com os dados ordenados. No entanto, a busca binária é diferente da busca sequencial, a binária precisa que o vetor obrigatoriamente esteja ordenado para encontrar o dado alvo.

Busca binária:Se o conjunto é grande e está ordenado.
Exemplo:
[10, 17, 36, 70, 100, 140, 170, 200]

Digamos que queremos encontrar o 140, a busca binária começaria pelo meio e descartaria a esquerda ou direita até encontrar o 140.

Busca sequencial: Se o conjunto for pequeno e desordenado
Exemplo:
[35, 28, 10, 7, 90, 100, 60, 71, 69]

Digamos que queremos encontrar o número 71, a busca sequencial iria percorrer todo o conjunto até encontrar o número.

Questão 6.2 — Impacto da Ordenação nos Algoritmos de Busca.
Tema: Pré-processamento e desempenho de algoritmos.

Enunciado: Considere uma situação onde é necessário fazer várias buscas sobre um mesmo conjunto de dados.

● **Vale a pena ordenar os dados previamente?**

Depende do tamanho do conjunto e das quantidades de buscas. Se fizer poucas buscas, o melhor é usar a busca sequencial, pois não haveria um gasto de custo e tempo para ele ser ordenado. Se fizer várias buscas e o conjunto de dados for grande, a melhor opção é usar a busca binária.

● **Qual o impacto dessa escolha no desempenho do sistema a longo prazo?**

Ordenar os dados previamente tem o custo de O(n log n), porém o custo será reduzido quando o número de buscas for muito grande. Então, a longo prazo, o desempenho do sistema tende a melhorar porque cada busca vai ser muito mais rápida. Caso contrário, se o sistema fizer poucas buscas, o custo da ordenação não compensaria.

● **Como isso se relaciona com o tempo de execução dos algoritmos envolvidos?**

A ordenação custa O(n log n), a busca binária custa O(log n), já a busca sequencial custa O(n). Logo, se fizer muitas buscas, o tempo de execução será menor usando a ordenação e a busca binária juntas, porque o custo da ordenação inicial será reduzido pelo número de buscas. Mas, se houver poucas buscas, a busca sequencial será mais eficiente, pois ela vai evitar o gasto excessivo de ordenação.

Questão 6.3 — Recursão x Iteração.
Tema: Estratégias de resolução de problemas

Enunciado: Explique as principais diferenças entre recursão e iteração na resolução de problemas algorítmicos. Discuta:

● **Quais são os prós e contras de cada abordagem?**

Prós da recursão: Expressa as soluções de uma forma mais natural e o código na maioria das vezes é menor e mais legível

Contras da recursão: Ela consome mais memória e pode ser mais lentas devido ao overhead das chamadas de função.

Prós da iteração: Na maioria das vezes é mais eficiente em tempo e memória e mais segura para grandes entradas.

Contra da interação:Pode ser menos clara que a recursão em problemas hierárquicos.

● **Em quais tipos de problemas a recursão pode ser mais vantajosa?**

A recursão pode ser mais vantajosa em estruturas de dados hierárquicos, algoritmos de divisão e busca(dividir e conquistar) como mergesort e quicksort e problemas matemáticos.

● **Existe alguma desvantagem de desempenho ou consumo de memória?**

Recursão: A recursão gasta memória proporcional à profundidade da chamada e pode ser mais lenta, justamente por causa das diversas chamadas de função

Iteração: A iteração perde a legibilidade em problemas recursivos.

Inclua um exemplo simples em pseudocódigo ou linguagem C/C++ para ilustrar sua resposta

Questão 6.4 — Análise de um Cenário Real.
Tema: Escolha de algoritmos na prática

Enunciado: Imagine que você trabalha no setor de tecnologia de uma empresa de logística. O sistema precisa localizar rapidamente pacotes com base em códigos de rastreio que chegam em tempo real. Os dados chegam desordenados.

● **Qual algoritmo de busca você recomendaria para este sistema?**

Entre os dois tipos de busca que vimos, mesmo com os dados desordenados, a busca binária ainda seria a melhor opção.

● **Justifique tecnicamente sua escolha considerando tempo de resposta, necessidade de ordenação e custo computacional.**

A busca binária mesmo com os dados desordenados, ainda teria o seu nível de complexidade sendo O(log n), mas como visto anteriormente, os resultados provavelmente sairiam errados, pois um dos requisitos da busca binária é que todos os seus dados precisam estar ordenados para funcionar corretamente sendo a sua complexidade O(n * log n) e sendo mais eficiente em grandes conjuntos de dados do que a busca sequencial. Logo, para esse cenário, a busca binária ainda sim seria melhor que a busca sequencial.

● **Caso os dados pudessem ser pré-processados, sua escolha mudaria?**

Minha escolha não mudaria. Ela se tornaria ainda mais viável e eficiente, ao ordenar os dados, a busca binária poderia ser aplicada corretamente, garantindo a sua complexidade O(log n) em grande conjunto de dados, o que seria ainda mais eficiente do que a busca sequencial e sua complexidade de O(n). O custo adicional para o pré-processamento seria de O(n log n), mas isso seria compensado quando há diversas buscas sobre os mesmos dados.