Questões Teóricas - Arthur Luiz Rosado Alves

6.1 - Complexidade de algoritmo de Busca
Tema: Busca sequencial vs. busca binária

A)

Inicialmente, o contexto de cada um é diferente, a busca sequencial se trata do algoritmo mais simples de busca, onde percorre elemento por elemento até encontrar o valor desejado ou todos os elementos serem verificados. Assim **não exige** que os dados estejam **ordenados**, ideal para conjuntos pequenos, em seu pior caso teremos a complexidade O(N). A busca binária se trata de um algoritmo um pouco mais sofisticado onde divide repetidamente o conjunto ao meio, descartando a outra metade onde o elemento não pode estar. Desse modo, se torna um algoritmo altamente eficiente para grandes volumes, em seu pior caso teremos O(log n), porém é necessário que o vetor ou a lista esteja necessariamente **ordenada**.

B)

os pré requisitos como dito anteriormente, para início de conversa o vetor necessariamente deve estar ordenado, seja em ordem crescente ou decrescente. Também é necessário ter o início e o fim do conjunto bem definido, se tratando dos elementos,eles devem ter como comparar uns aos outros para assim definir o que seria maior ou menor do que o meio.

C)

A ordenação tem forte influência na decisão do algoritmo de busca, primeiramente sem o vetor está ordenado nem temos a opção de escolher a busca binária. de outro modo, podemos otimizar o algoritmo de busca sequencial se o vetor estiver ordenado para o caso em que a chave não se encontra no vetor indicado, se vetor[i] > chave podemos dar um break no algoritmo assim reduzindo o número de operações, a complexidade do algoritmo continua O(n) mas o tempo médio melhora um pouco, quando está ordenado.

Exemplos:

Em um sistema de estoque gigante com milhões de produtos, onde os produtos estão indicados por uma primary key ordenada, a **busca binária** será muito eficiente para encontrar o produto em poucos passos, mesmo contendo milhões de produtos.

Digamos que você está numa rede de wifi local onde temos menos de 15 dispositivos conectados e quer verificar se aquele dispositivo específico está conectado, esse processo de conferir seria muito mais viável pela busca sequencial, pois o número de dispositivos conectados é pequeno, a

ordenação também seria inviável pois essa rede sofre alterações a todo momento por um dispositivo cair da rede ou algo do tipo. Nesses casos, a busca sequencial garante simplicidade, baixo custo computacional e um bom tempo também. (vale lembrar que é um caso de baixa escala, se fosse uma rede gigante de um datacenter iremos repensar esse caso)

Algoritmo de busca	melhor caso	pior caso	caso medio
Sequencial	O(1)	O(n)	O(n)
Binaria	O(1)	O(log n)	O(log n)

6.2 Impacto da Ordenação nos Algoritmos de Busca

A) Sim, nesse caso de “várias buscas” a ordenação só seria feita uma vez, ou seja o custo de fazer essa ordenação seria diluído por cada busca. Então o custo de ordenar se paga ao longo das buscas, pela melhora de eficiência.
B) no longo prazo, você estaria transformando buscas que seriam feitas na complexidade O(n) busca sequencial para complexidade O(log n) busca binária, no longo prazo isso indicaria um sistema mais eficiente, na escala de tempo x memória, assim seria um sistema muito mais escalável em dados que crescem com o tempo.
C) Como dito anteriormente, sem ordenação cada busca iria solicitar o custo de O(n), com a ordenação o custo iria para O(log n). a complexidade O(log n) torna o tempo de busca muito mais rápido a medida que o sistema escala, saindo de um crescimento linear para um crescimento logaritmo (muito mais eficiente). Dessa forma, o tempo por consulta fica bem menor à medida que o sistema escala.

6.3 Recursão e Iteração

Inicialmente, Recursão se trata quando uma função chama ela mesma para resolver partes menores desse problema, tendendo esse problema a um caso base, onde está a condição de parada. Iteração, é quando usa-se laços repetindo aquele código até a condição de parada ser satisfeita.

A) Recursão: prós: tem como principal vantagem, dividir um problema complexo em subproblemas com complexidade menor, trazendo assim maior legibilidade e clareza ao código, em termos de eficiência a recursão se destaca nos algoritmos de dividir para conquistar, de ordenação como o quick

sort garantindo assim complexidade O(n log n), muito mais eficientes do que métodos baseados em iteração.
Contra: cada chamada consome espaço na memória, podendo gerar chamadas redundantes e aumentando o tempo de execução, tendo assim um alto custo de memória e de processamento.
Iteração:

Prós: eficiência de memória, dando uma condição de parada clara que seria uma restrição para o algoritmo, evitando assim chamadas desnecessárias. No geral, garante um desempenho previsível.

Contra: a interação normalmente demanda estruturas como vetores extras, aumentando o custo de memória e em alguns casos, também o número de operações necessárias, tornando a solução mais trabalhosa e menos eficiente em termos de memória e tempo.

B) Em problemas em que vai executar um algoritmo inerentemente recursivo, em que a solução pode ser expressa em termos do próprio problema. o uso da abordagem recursiva é mais simples em geral quando é possível dividir o problema pela metade (como no quicksort, busca binária)

C) Sim, a recursão traz uma desvantagem em alguns casos, pelo fato de cada chamada da função, é criado um registro independente na pilha de execução do programa, até que o caso base seja atingido. Assim, quando existem muitas chamadas recursivas o algoritmo se torna menos eficiente em termos de memória e processamento em comparação às soluções interativas.

Iteração, normalmente vai ser mais eficiente em termos de tempo e memória, menos nos problemas naturalmente recursivos que a solução interativa vai exigir estruturas adicionais, o que tornaria o algoritmo mais trabalhoso de implementar e menos legível para entender em comparação a versão recursiva.

6.4 Cenário Real

A/B) Inicialmente, com a informação que os dados chegam desordenados, pensei em duas opções: ordenar para usar a busca binária, ou já partir para busca sequencial. A primeira opção seria inválida pois, a cada momento chegariam novos pacotes, o que tornaria essa ordenação inviável em custo computacional. Portando usaria a busca sequencial para percorrer a lista até o item desejado que teria o custo de O(n) no pior caso e caso médio, mas sem depender de uma ordenação.

C) Sim, a busca binária passaria a ser minha escolha, pois, apos a ordenação dos dados, cada busca poderia ser realizada em O(log n), o que

representa uma redução significativa no tempo de resposta em comparação com O(n), tornando o sistema mais eficiente e escalável.