Atividade B3-2 – Comparando Eficiência de Algoritmos

Enunciado:

Instruções

Vimos que o tempo de execução de um algoritmo é a quantidade de passos básicos para tratar uma entrada de n elementos ***t(n).***

Com base nesta premissa, elabore uma tabela contagem de tempo, comparando os três algoritmos 1)Busca Linear, 2) Busca Linear em ordem e 3) Busca Binária, que sequem anexo.

Demonstre a análise de cada algorimo para justificar cada linha produzida na tabela

A tabela deverá ter a seguinte configuração:

	BuscaLinear	BuscaLinearEmOrdem	BuscaBinaria
$x \in A$			
$x = A[1]$			
$x = A[n]$			
$x \notin A$			

Resolução:

Vamos analisar cada um dos três algoritmos fornecidos: Busca Linear, Busca Linear em Ordem e Busca Binária. A análise será feita considerando o número de passos básicos necessários para cada um dos casos:

1. Busca Linear (A, n, x): Percorre todos os elementos da lista A até encontrar o elemento “x” ou até percorrer toda a lista.

2. Busca Linear em Ordem (A, n, x): Percorre a lista de forma linear, mas interrompe a busca assim que encontra um elemento maior que “x”, assumindo que a lista está ordenada.

3. Busca Binária (A, n, x): Divide a lista pela metade a cada iteração, buscando “x” em uma lista ordenada.

Vamos analisar cada cenário (coluna da tabela) para cada algoritmo:

Cenário 1: X ∈ A

- Busca Linear:

  - A busca linear vai iterar até encontrar “x” e, em média, espera-se que o elemento seja encontrado na metade da lista. Assim, o número de passos é “n/2” no caso médio.

- Busca Linear em Ordem:

  - Mesmo que a lista esteja ordenada, a busca linear em ordem tem o mesmo comportamento que a busca linear, pois não há ganho se “x” for um elemento pequeno ou grande dentro da lista, resultando em um número médio de passos de “n/2”.

- Busca Binária:

  - A busca binária divide o problema pela metade a cada iteração. O tempo de execução é log2(n), pois é o número de vezes que podemos dividir “n” por 2 até chegar a 1.

Cenário 2: X = A[1]

- Busca Linear:

  - O melhor caso ocorre quando “x” é o primeiro elemento da lista. Apenas 1 passo é necessário.

- Busca Linear em Ordem:

  - Assim como na busca linear, apenas 1 passo é necessário para encontrar “x” no primeiro elemento.

- Busca Binária:

  - A busca binária, mesmo no melhor caso, ainda precisa calcular o índice médio e fazer comparações, resultando em um tempo de execução constante, “O(1)”.

Cenário 3: X = A[n]

- Busca Linear:

  - O pior caso para a busca linear ocorre quando “x” é o último elemento da lista. Neste caso, a busca linear precisa percorrer todos os “n” elementos.

- Busca Linear em Ordem:

- Mesmo que a lista esteja ordenada, a busca linear em ordem ainda precisa percorrer todos os “n” elementos até encontrar “x”.

- Busca Binária:

- A busca binária é mais eficiente, mas ainda levará “log2(n)” passos, independentemente de onde “x” está na lista.

Cenário 4: X ∉ A

- Busca Linear:

- O pior caso ocorre quando “x” não está presente na lista. A busca linear precisa percorrer todos os “n” elementos.

- Busca Linear em Ordem:

- A busca linear em ordem pode sair antes, se encontrar um elemento maior que “x”. No pior caso, precisará percorrer “n” elementos.

- Busca Binária:

- A busca binária precisará dividir a lista até que o intervalo se esgote, resultando em “log2(n)” passos.

Com base nas análises acima, aqui está a tabela de contagem de tempo para os três algoritmos:

Situação	Busca Linear	Busca Linear em Ordem	Busca Binária
X ∈ A	n/2	n/2	Log2(n)
X = A[1]	1	1	O(1)
X = A[n]	n	n	Log2(n)
X ∉ A	n	n	Log2(n)

Resumo da Análise:

1. Busca Linear:

- Melhor caso: 1 passo.

- Pior caso: “n” passos.

- Caso médio: “n/2” passos.

2. Busca Linear em Ordem:

- Melhora marginal se o elemento não estiver na lista e a lista estiver ordenada, mas na prática o comportamento é semelhante à busca linear.

3. Busca Binária:

- Muito mais eficiente em listas ordenadas, com complexidade" log2(n)" para todos os casos, exceto o melhor caso trivial.

Exemplos para cada situação nos três tipos de busca (linear, linear em ordem, binária)

Exemplos para cada uma das situações nas três buscas: Busca Linear, Busca Linear em Ordem, e Busca Binária. Para isso, consideremos uma lista A = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19] com "n = 10".

1. $x \in A$

Aqui, o elemento "x" está na lista, mas sua posição não é especificada.

Exemplo: "x = 7" (presente na lista).

- Busca Linear:

- O algoritmo começa do início da lista e compara cada elemento até encontrar 7.

- Comparações: 1, 3, 5, **7 (encontrado na 4ª posição).**

- Total de passos: 4.

- Busca Linear em Ordem:

- Similar à busca linear, começa do início e compara cada elemento até encontrar 7.

- Comparações: 1, 3, 5, **7 (encontrado na 4ª posição).**

- Total de passos: 4.

- Busca Binária:

- O algoritmo calcula o meio da lista: meio = 1 + 10/2 = 5 . A[5] = 9

- Como “x = 7” é menor que 9, ele descarta a segunda metade e continua na primeira metade.

- Agora, o novo meio é {meio} = 1 + 4/2 = 2 . A[2] = 3.

- Como “x = 7” é maior que 3, ele continua na segunda metade dessa lista.

- O próximo meio é {meio} = 3 + 4/2 = 3. A[3] = 5.

- Como “x = 7” é maior que 5, ele finalmente encontra A[4] = 7.

- Total de passos: 4.

2. x = A[1]

Aqui, o elemento “x” é o primeiro da lista.

Exemplo: “x = 1” (primeiro elemento).

- Busca Linear:

- O algoritmo compara o primeiro elemento da lista.

- Comparação: 1 (encontrado na 1ª posição).

- Total de passos: 1.

- Busca Linear em Ordem:

- Similar à busca linear, o primeiro elemento é comparado imediatamente.

- Comparação: 1 (encontrado na 1ª posição).

- Total de passos: 1.

- Busca Binária:

- Mesmo que 1 esteja na primeira posição, a busca binária ainda precisa calcular o meio.

- {meio} = 5 \). A[5] = 9. Como 1 é menor que 9, ele reduz a busca à primeira metade.

- Novo meio: {meio} = 2. A[2] = 3. Como 1 é menor que 3, ele continua a busca.

- Novo meio: {meio} = 1. A[1] = 1.

- Comparação: 1 (encontrado na 1ª posição).

- Total de passos: 3.

3. x = A[n]

Aqui, o elemento “x” é o último da lista.

Exemplo: **“x = 19” (último elemento).**

- Busca Linear:

  - O algoritmo compara cada elemento até o final da lista.

  - Comparações: 1, 3, 5, 7, 9, 11, 13, 15, 17, **19 (encontrado na 10ª posição).**

  - Total de passos: 10.

- Busca Linear em Ordem:

  - Mesmo com a lista ordenada, o algoritmo percorre até o final.

  - Comparações: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 (encontrado na 10ª posição).

  - Total de passos: 10.

- Busca Binária:

  - Calcula o meio: {meio} = 5. A[5] = 9. Como 19 é maior que 9, a busca continua na segunda metade.

  - Novo meio: {meio} = 8. A[8] = 15. Como 19 é maior que 15, continua a busca.

  - Novo meio: {meio} = 9. A[9] = 17. Como 19 é maior que 17, continua a busca.

  - Comparação: {meio} = 10. A[10] = 19.

  - Total de passos: 4.

4. x ∉ A

Aqui, o elemento “x” não está presente na lista.

Exemplo: “x = 8” (não está na lista).

- Busca Linear:

  - O algoritmo percorre todos os elementos até o final e não encontra “x”.

- Comparações: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19.

- Total de passos: 10 (devolve -1).

- Busca Linear em Ordem:

- O algoritmo percorre a lista até que um elemento maior que 8 seja encontrado.

- Comparações: 1, 3, 5, 7, **9 (interrompe a busca, devolve -1).**

- Total de passos: 5 (devolve -1).

- Busca Binária:

- Calcula o meio: {meio} = 5. A[5] = 9. Como 8 é menor que 9, a busca continua na primeira metade.

- Novo meio: {meio} = 3. A[3] = 5. Como 8 é maior que 5, continua a busca.

- Novo meio: {meio} = 4. A[4] = 7. Como 8 é maior que 7, a busca continua.

- {meio} = 5, mas já foi verificado. Busca esgotada (devolve -1).

- Total de passos: 4 (devolve -1).

Em Resumo:

- Busca Linear: tem tempo linear, fazendo comparações até encontrar x ou esgotar a lista.

- Busca Linear em Ordem: se comporta de forma similar à Busca Linear, mas pode parar mais cedo em listas ordenadas se um elemento maior que x for encontrado.

- Busca Binária: é mais eficiente em listas ordenadas, mas sempre precisa dividir a lista até que o intervalo de busca se esgote.