BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Professora:
ADRIANA PADUA LOVATTE
Grupo:
GUSTAVO SARAIVA MARIANO
JOÃO VICTOR NASCIMENTO RIBEIRO

TRABALHO
MATEMÁTICA DISCRETA

SERRA - ES
2024

INTRODUÇÃO

Ordenar dados é uma tarefa essencial em computação e matemática, e diferentes algoritmos fazem isso de maneiras variadas. Neste trabalho, vamos explorar dois algoritmos de ordenação: o Bubble Sort e o Merge Sort.

O Bubble Sort é um método simples que compara e troca elementos adjacentes até que tudo esteja na ordem correta. É fácil de entender, mas pode ser lento com listas grandes. Por outro lado, o Merge Sort é mais eficiente. Ele divide a lista em partes menores, ordena essas partes e depois as junta novamente. Esse método é mais rápido para listas grandes, embora seja um pouco mais complexo.

Vamos comparar esses dois métodos para ver como cada um se sai e quando é melhor usar cada um deles.

DESENVOLVIMENTO

BUBBLE SORT

Para encontrarmos a sequência de recorrência e a fórmula fechada do Bubble Sort, primeiro precisamos saber como funciona o algoritmo:

1) **Lógica do algoritmo:**
    - O Bubble Sort realiza várias passagens pela lista.
    - Em cada passagem pela lista, de forma sequencial, ele compara os elementos de forma adjacente, sempre de dois em dois, e os troca se necessário.
    - No final de cada passagem, o maior elemento terá sido deslocado para o final da lista.

2) **Número de comparações do algoritmo a cada iteração:**
   - Na primeira passagem, o algoritmo realiza ***n - 1*** comparações.
   - Na segunda passagem, o algoritmo realiza ***n - 2*** comparações, pois o maior elemento da primeira passagem já estará na posição correta.
   - Na terceira passagem, realizará ***n - 3*** comparações, e assim por diante.

3) **Definindo a sequência de recorrência:**
   O tempo total $C_{(n)}$ para ordenar uma lista de ***n*** elementos é o tempo para ordenar os ***n − 1*** elementos restantes, somado o número de comparações feitas na ***n-ésima*** iteração anterior.

   Caso base:
   - Para ***n = 1,*** não haverá comparações, pois só possui um elemento na lista, portanto, o número de comparações é igual a $C_{(1)} = 0$.

   Iteração até n elementos:
   - Para ***n = 2,*** de acordo com a definição, será as comparações restantes dos ***n - 1*** elementos, somado com o número de comparações feitas no elemento anterior, ou seja,

$$C_{(2)} = C(2 - 1) + (2 - 1) \Rightarrow$$
$$C_{(2)} = C(1) + 1 \Rightarrow$$
$$C_{(2)} = 0 + 1 \Rightarrow$$
$$C_{(2)} = 1$$

- Para ***n = 3***, de acordo com a definição, será as comparações restantes dos ***n - 1*** elementos, somado com o número de comparações feitas no elemento anterior, ou seja,

$$C_{(3)} = C(3 - 1) + (3 - 1) \Rightarrow$$
$$C_{(3)} = C(2) + 2 \Rightarrow$$
$$C_{(3)} = 1 + 2 \Rightarrow$$
$$C_{(3)} = 3$$

- Então, para ***n*** termos, temos:

$$C_{(n)} = C(n - 1) + (n - 1)$$

Sequência de recorrência do Bubble Sort

4) **Definindo a fórmula fechada:**

Para construir uma fórmula fechada, muitas vezes começamos observando os valores calculados da sequência. No caso do **Bubble Sort**, é possível calcular alguns valores iniciais e observar um padrão.

Para os primeiros valores calculados anteriormente:

- $C_{(1)} = 0$
- $C_{(2)} = 1$
- $C_{(3)} = 2$
- $C_{(4)} = 6$
- $C_{(5)} = 10$
- E assim por diante, até ***n*** iterações.

- Podemos observar que os valores são resultantes da soma das comparações anteriores com o número de comparações que serão realizadas na iteração atual, observando assim, que se trata de uma progressão aritmética, vejamos a fórmula de uma P.A.:

$$S_m = \frac{m * (a_1 + a_m)}{2}$$

a_1 = valor do primeiro termo

a_m = valor do último termo

m = número de termos

Fórmula da Progressão Aritmética

- Utilizando a fórmula auxiliar demonstrada em sala de aula para encontrar a fórmula fechada do **Bubble Sort**, devemos realizar os seguintes passos:

$$S_{(n)} = C^{n-1} * S(1) + (\sum^{n} i = 2 \quad C^{n-i}) * g(i)$$

Fórmula auxiliar

Temos pela sequência de recorrência:

S(1) = 0

C = 1

g(i) = (n - 1)

Realizando as trocas na fórmula, teremos:

$$S_{(n)} = 1^{n-1} * 0 + (\sum^{n} i = 2 \quad 1^{n-i}) * (n - 1) \Rightarrow$$

$$S_{(n)} = 0 + (\sum^{n} i = 2 \quad n - 1)$$

- A somatória dos ***n - 1 ; n > 1*** termos é:

(2 - 1) + (3 - 1) + (4 - 1) + … + (n -1)

- Portanto, a progressão aritmética e a definição da fórmula fechada dessa somatória seria:

$$a_1 = 1$$
$$a_m = (n - 1)$$
$$m = (n - 1)$$
$$S_{(n-1)} = ((n - 1) * (1 + (n - 1))) / 2 \Rightarrow$$
$$S_{(n-1)} = ((n - 1) * n) / 2$$

Fórmula fechada do Bubble Sort

- Agora, para demonstrar a validade da fórmula fechada encontrada devemos utilizar indução matemática:

Caso base: $C_{(1)} = 0$

$$n = 1$$
$$P_{(1)} = ((1 - 1) * 1) / 2 \Rightarrow$$
$$P_{(1)} = 0 \quad OK$$

Passo Indutivo:

Para $P_{(k)}$ verdade, $P_{(k+1)}$ verdade.

$C_{(k)} = C(k - 1) + (k - 1)$ **Sequência recorrente (S.R.)** VERDADE

$P_{(k)}$: $C_{(k)} = (k * (k - 1)) / 2$ **Fórmula fechada** VERDADE

$P_{(k+1)}$: $C(k) + k = ((k + 1) * k) / 2$ **PROVAR!**

Usando $C(k)$ em $P(k + 1)$

$P_{(k+1)}$: $((k * (k - 1)) / 2) + k = (k * (k + 1)) / 2 \Rightarrow$

$P_{(k+1)}$: $(k^2 - k + 2k) / 2 = (k * (k + 1)) / 2 \Rightarrow$

Simplificando:

$P_{(k+1)}$: $(k^2 + k) / 2 = (k * (k + 1)) / 2 \Rightarrow$

Colocando *k* em evidência do lado esquerdo da igualdade

$P_{(k+1)}$: $(k * (k + 1)) / 2 = (k * (k + 1)) / 2$ VERDADE

- Com isso, podemos demonstrar a validade da fórmula fechada da sequência de recorrência do Bubble Sort.

IMPLEMENTAÇÃO DO ALGORITMO BUBBLE SORT EM LINGUAGEM DE PROGRAMAÇÃO:

Imagem: CodigoBUBBLE.png

Código: **BUBBLESORT.py**

MERGE SORT

Para encontrarmos a sequência de recorrência e a fórmula fechada do Merge Sort, primeiro precisamos saber como funciona o algoritmo:

1) **Divisão do problema:**
   - O Merge Sort é um algoritmo que utiliza a abordagem de "Dividir para Conquistar".
   - Ou seja, para uma lista de tamanho n, o algoritmo divide a lista em duas sublistas de tamanho $n / 2$.
   - Cada uma dessas sublistas são ordenadas recursivamente pelo Merge Sort.
   - Após ordenar as sublistas, o algoritmo as combina (daí então que vem o nome de Merge, que é combinação em inglês) em uma única lista ordenada por completo.

2) **Construindo a Sequência de Recorrência:**
   - O número de comparações que o Merge Sort irá realizar para cada sublista será de *duas vezes* o número de comparações para uma das sublistas de tamanho $n / 2$, podemos então, expressar como:

$$C_{(n)} = 2C(n / 2)$$

   - Ademais, após comparar as sublistas, ele precisa fazer a comparação dessas duas sublistas já ordenadas para retornar elas em uma só lista ordenada, fazendo o "merge" dessas duas sublistas de acordo com o número de elementos, ou seja, n vezes, podendo então acrescentar a expressão anterior:

$$C_{(n)} = 2C(n/2) + n$$

- Portanto, podemos expressar a relação do Merge Sort, para ***n*** elementos, como:

$$C_{(n)} = 2C(n/2) + n$$

Sequência de recorrência do Merge Sort

- Expandindo a relação de recorrência, começamos a observar um padrão:

$$C_{(n)} = 2C(n/2) + n \Rightarrow$$
$$C_{(2)} = 2[2C(n/4) + (n/2)] + n \Rightarrow$$
$$C_{(2)} = 4C(n/4) + 2(n/2) + n \Rightarrow$$
$$C_{(3)} = 4[2C(n/8) + (n/4)] + 2(n/2) + n \Rightarrow$$
$$C_{(3)} = 8C(n/8) + 4(n/4) + 2(n/2) + n$$

- Quando $2C(n/2^{k})$ for igual a *1*, ***k*** assume o valor de $\log_2 n$. Então, teremos:

$$C_{(n)} = n * C(1) + n * \log_2 n$$

- Sabendo que $C_{(1)}$ é *0*, pois para ordenar uma lista de tamanho 1 não requer nenhuma comparação, temos que:

$$C_{(n)} = n * \log n \text{ (log na base 2)}$$

Fórmula fechada do Merge Sort

- Agora, para demonstrar a validade da fórmula fechada encontrada devemos utilizar indução matemática:

Caso base:

$$C_{(1)} = 0$$
$$n = 1$$
$$P_{(1)} = 1 * \log_2 1$$
$$P_{(1)} = 1 * 0$$
$$P_{(1)} = 0 \text{ OK}$$

Passo Indutivo:

Para $P_{(k)}$ verdade, $P_{(k+1)}$ verdade.

$C_{(k)} = 2C(k/2) + k$ **Sequência recorrente VERDADE**

$P_{(k)}$: $k * \log_2 k$ **Fórmula fechada VERDADE**

$P_{(k+1)}$: $2C((k+1)/2) + (k+1) = (k+1) * \log_2 (k+1)$ **PROVAR**

Como a fórmula fechada é derivada da sequência de recorrência, podemos escrever C((k + 1) / 2), como:

$P_{(k+1)}$: $2 * ((k+1)/2) \log_2 ((k+1)/2) + (k+1) \Rightarrow$

Simplificando, temos:

$P_{(k+1)}$: $(k+1) \log_2 ((k+1)/2) + (k+1) \Rightarrow$

Utilizando a propriedade da divisão de logaritmo em $\log_2 ((k + 1) / 2)$, iremos obter:

$\log_2 ((k + 1) / 2) = \log_2 (k + 1) - \log_2 2$ ($\log_2 2 = 1$)

Substituindo na expressão:

$P_{(k+1)}$: $(k+1) * (\log_2 (k+1) - 1) + (k+1) \Rightarrow$

Fazendo a distributiva em (k + 1):

$P_{(k+1)}$: $((k+1) * \log_2 (k+1)) - (k+1) + (k+1) \Rightarrow$

Os últimos dois termos se anulam, - (k + 1) + (k + 1), restando:

$P_{(k+1)}$: $(k+1) \log_2 (k+1) = (k+1) \log_2 (k+1)$ **VERDADE**

- Com isso, podemos demonstrar a validade da fórmula fechada da sequência de recorrência do Merge Sort.

IMPLEMENTAÇÃO DO ALGORITMO MERGE SORT EM LINGUAGEM DE PROGRAMAÇÃO:

Imagem: CodigoMERGE.png

Código: MERGESORT.py

TABELA DE COMPARAÇÃO

Bubble Sort

Tipo de lista	Comparações	Segundos
Lista Crescente (10 elementos)	9	0.01s
Lista Decrescente (10 elementos)	45	0.01s
Lista aleatória com 50 mil elementos	1.249.960.804	136.36s

Imagem com resultados: resultadoBUBBLE.png

Merge Sort

Tipo de lista	Comparações	Segundos
Lista Crescente (10 elementos)	15	0.00s
Lista Decrescente (10 elementos)	19	0.00s
Lista aleatória com 50 mil elementos	718.013	0.13s

Imagem com resultados: resultadoMERGE.png

CONCLUSÃO

Neste trabalho, exploramos o funcionamento e as diferenças entre o Bubble Sort e o Merge Sort. Vimos que, enquanto o Bubble Sort é simples e fácil de implementar, ele não é eficiente para listas grandes. Já o Merge Sort, apesar de ser mais complexo, se destacou pela sua eficiência em ordenar grandes quantidades de dados.

Além disso, analisamos as sequências de recorrência associadas aos algoritmos e derivamos suas fórmulas fechadas. Esse tipo de análise é crucial para entender o comportamento dos algoritmos e nos permite fazer escolhas informadas sobre qual deles usar em diferentes situações.

PERGUNTAS RESPONDIDAS PELO GRUPO

1	**O que o grupo achou de fazer este trabalho? Resposta discursiva** Foi um trabalho bem desafiador que colocou o grupo atrás de resolução e exercícios para resolver e entender melhor a matéria e a proposta apresentada na documentação do trabalho.
2	**O grupo identificou algum conteúdo da ementa da disciplina com o assunto do trabalho? Resposta discursiva** Sim.
3	**Se a resposta do item 2 foi positiva, qual ? Resposta discursiva** Indução matemática e relações de recorrência.
4	**O grupo acha interessante que trabalhos como este sejam colocados como parte da avaliação do aluno? Resposta (Sim ou Não) Porque ? (Resposta discursiva)** Sim, porque estimula o aluno a ir atrás de resoluções para os problemas apresentados no trabalho, fazendo com que seu desenvolvimento e conhecimento na matéria se abrangem.
5	**O grupo já conhecia (ou ouviu falar) sobre o tema que realizou o trabalho? Resposta (Sim ou Não).** Sim, em outra matéria do curso.
6	**Quais dificuldades foram encontradas na realização do trabalho?** Entender a lógica por trás de cada parte das sequências de recorrências dos métodos escolhidos e fontes na internet que ajudem a entender melhor.
7	**O conteúdo da disciplina que foi utilizado no trabalho ficou melhor entendido?** Uma parte sim, porém algumas dúvidas surgiram no processo de desenvolvimento do trabalho, que depois foram esclarecidas em sala de aula.
8	**O grupo gostaria de elaborar uma nova pergunta para futuros trabalhos? Qual ?** Não temos perguntas novas a acrescentar.

REFERÊNCIAS BIBLIOGRÁFICAS

TREINAWEB. **Conheça os principais algoritmos de ordenação**. TreinaWeb, 17 mar. 2021. Disponível em: https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao. Acesso em: 13 ago. 2024.

BRIANCO, Gabriel. **Algoritmos de Ordenação - Como funcionam?** YouTube, 27 fev. 2020. Disponível em: https://youtu.be/UulVP9VayJA. Acesso em: 13 ago. 2024.

GEEKSFORGEEKS. **Bubble Sort Algorithm**. GeeksforGeeks, s.d. Disponível em: https://www.geeksforgeeks.org/bubble-sort-algorithm/. Acesso em: 13 ago. 2024.

BETRYBE. **Bubble Sort: tudo sobre esse algoritmo de ordenação**. Blog da Trybe, 22 dez. 2022. Disponível em: https://blog.betrybe.com/tecnologia/bubble-sort-tudo-sobre/. Acesso em: 13 ago. 2024.

W3SCHOOLS. **Bubble Sort Algorithm**. W3Schools, s.d. Disponível em: https://www.w3schools.com/dsa/dsa_algo_bubblesort.php. Acesso em: 13 ago. 2024.

GEEKSFORGEEKS. **Merge Sort**. GeeksforGeeks, s.d. Disponível em: https://www.geeksforgeeks.org/merge-sort/. Acesso em: 13 ago. 2024.

MOURA, João Arthur B. **Merge Sort**. Estruturas de Dados, 23 abr. 2021. Disponível em: https://joaoarthurbm.github.io/eda/posts/merge-sort/. Acesso em: 13 ago. 2024.

MOURA, João Arthur B. **Análise de Algoritmos Recursivos**. Estruturas de Dados, 27 nov. 2021. Disponível em: https://joaoarthurbm.github.io/eda/posts/analise-algoritmos-recursivos/. Acesso em: 13 ago. 2024.

BRIANCO, Gabriel. **Merge Sort - Como funciona?** YouTube, 12 mar. 2020. Disponível em: https://www.youtube.com/watch?v=3j0SWDX4AtU. Acesso em: 13 ago. 2024.

MENDES, Guilherme R. **Algoritmos: Merge Sort**. Medium, 26 out. 2021. Disponível em: https://guilherme-rmendes95.medium.com/algoritmos-merge-sort-ef12dadeba2a. Acesso em: 13 ago. 2024.

PROGRAD. **MATHED#7 - Análise de Algoritmos Recursivos: Exemplos e Recorrências**. YouTube, 9 dez. 2021. Disponível em: https://youtu.be/7ssHXAzAyuU. Acesso em: 13 ago. 2024.

FERREIRA, Roger. **Teorema Mestre e a Análise de Algoritmos Recursivos**. YouTube, 26 nov. 2020. Disponível em: https://www.youtube.com/watch?v=XO2YNACqD6w. Acesso em: 13 ago. 2024.

Merge Sort Algorithm Explained. YouTube, 18 jun. 2021. Disponível em: https://www.youtube.com/watch?v=JuXZz-B8dJM. Acesso em: 14 ago. 2024.

Merge Sort Explained. YouTube, 5 nov. 2021. Disponível em: https://www.youtube.com/watch?v=gXWGMr6lZUg. Acesso em: 14 ago. 2024.

Merge Sort Algorithm - Detailed Explanation. YouTube, 25 fev. 2022. Disponível em: https://www.youtube.com/watch?v=rG08wJgfydA&t=2s. Acesso em: 14 ago. 2024.