Complexidade de algoritmos

Árvore Rubro-Negra

A árvore rubro-negra é uma estrutura de dados de árvore binária balanceada, caracterizada por garantir que o caminho mais longo da raiz até uma folha não seja mais do que o dobro do caminho mais curto. Isso é conseguido através de regras que garantem a propriedade de balanceamento, onde cada nó é colorido de vermelho ou preto e devem seguir restrições específicas quanto à coloração. Essa estrutura permite que operações de inserção, exclusão e busca sejam realizadas em tempo O(log n), tornando-a altamente eficiente para grandes volumes de dados.

A principal vantagem da árvore rubro-negra em relação à árvore binária de busca (ABB) tradicional é a manutenção do balanceamento. Enquanto a ABB pode degenerar para uma lista ligada em casos de entradas ordenadas ou quase ordenadas, resultando em operações de tempo O(n), a árvore rubro-negra assegura operações eficientes em O(log n) mesmo no pior caso. Isso é particularmente importante ao lidar com bases de dados grandes e dinâmicas, onde a consistência na performance é crucial.

Estrutura de Dado	Complexidade do Tempo: Caso Médio: Indexing	Complexidade do Tempo: Caso Médio: Busca	Complexidade do Tempo: Caso Médio: Inserção	Complexidade do Tempo: Caso Médio: Remoção	Complexidade do Tempo: Pior Caso: Indexing	Complexidade do Tempo: Pior Caso: Busca	Complexidade do Tempo: Pior Caso: Inserção	Complexidade do Tempo: Pior Caso: Remoção	Espaço de Complexidade: Pior Caso
Árvore com Busca Binária	-	O((log n))	O((log n))	O((log n))	-	O(n)	O(n)	O(n)	O(n)
Árvore-B	-	O((log n))	O((log n))	O((log n))	-	O((log n))	O((log n))	O((log n))	O(n)
Árvore Rubro-Negra	-	O((log n))	O((log n))	O((log n))	-	O((log n))	O((log n))	O((log n))	O(n)
Árvore AVL	-	O((log n))	O((log n))	O((log n))	-	O((log n))	O((log n))	O((log n))	O(n)

Smooth Sort

O Smooth Sort é um algoritmo de ordenação desenvolvido por Edsger Dijkstra que combina as vantagens do Heap Sort com melhorias para eficiência em casos específicos. Ele é um algoritmo adaptativo com complexidade de tempo O(n log n) no pior caso, mas pode alcançar desempenho próximo a O(n) em casos de dados quase ordenados. Essa característica adaptativa torna o Smooth Sort particularmente eficiente para conjuntos de dados onde a ordenação está parcialmente presente.

Uma das vantagens do Smooth Sort sobre o Quick Sort é a estabilidade e a melhor gestão do pior caso. Enquanto o Quick Sort tem complexidade O(n^2) no pior cenário, o Smooth Sort garante O(n log n) independentemente da distribuição dos dados. Isso é crucial em aplicações onde o pior caso precisa ser evitado a todo custo, como em sistemas de tempo real ou bancos de dados com grandes volumes de dados.

Algoritmo	Estrutura De Dados	Complexidade do Tempo: Melhor Caso	Complexidade do Tempo: Caso Médio	Complexidade do Tempo: Pior Caso	Espaço de Complexidade: Pior Caso
Quick Sort	Array	O(n log(n))	O(n log(n))	$O(n^2)$	O(log(n))
Merge sort	Array	O(n log(n))	O(n log(n))	O(n log(n))	O(n)
Heap sort	Array	O(n log(n))	O(n log(n))	O(n log(n))	O(1)
Smooth sort	Array	O(n)	O(n log(n))	O(n log(n))	O(1)

Resultados dos Testes

A escolha da árvore rubro-negra e do Smooth Sort foi fundamentada em suas características de desempenho consistentes e robustas, especialmente ao lidar com grandes volumes de dados. Durante os testes realizados com a classe CustomerOperatorTest, os tempos de execução para operações de busca e ordenação foram mensurados e comparados. A árvore rubro-negra demonstrou uma velocidade de busca superior à da ABB tradicional, mantendo um tempo de resposta consistente mesmo com inserções e exclusões frequentes.

Os testes com o Smooth Sort mostraram que ele supera o Quick Sort em cenários onde os dados são quase ordenados, graças à sua adaptabilidade. Embora o Quick Sort seja conhecido por sua rapidez em muitos casos, o Smooth Sort mostrou-se mais confiável, evitando os piores casos que podem ocorrer com o Quick Sort. Porém nesse caso em específico, o quick sort se mostrou um pouco mais rápido, porém ele só supera o smooth sort em casos médios, sendo pior no melhor e no pior caso

Algoritmo	Tempo de Execução
ABB	384 ms
Árvore Rubro-Negra	356 ms
Quick Sort	320 ms
Smooth Sort	431 ms