Relatório do Projeto II

Integrantes:

Jesus Sena Fernandes - 12697470
Tiago Chaves Bezerra - 14609637

Justificativas para escolha de Árvores AVL para o projeto:

a. **Balanceamento rigoroso**: as Árvores AVL mantêm um equilíbrio mais estrito em comparação com as Árvores Rubro-Negras. Por exemplo, para uma árvore AVL com 1000000 de elementos, a altura máxima seria em torno de 20 (1,44 log(1000000)), enquanto uma Árvore Rubro-Negra poderia ter uma altura de até 40 (2 log(1000000)). Este balanceamento mais rigoroso é crucial para garantir tempos de busca e acesso eficientes. Em contraste, a flexibilidade maior no balanceamento das Árvores Rubro-Negras pode levar a uma altura maior, resultando em tempos de busca e acesso mais lentos.
b. **Altura reduzida**: a altura menor das Árvores AVL torna operações como busca, inserção e remoção mais rápidas. Imaginando uma operação de busca em uma Árvore AVL com altura 20 contra uma Árvore Rubro-Negra com altura 40; a busca na AVL seria, em média, mais rápida devido ao menor número de níveis a serem percorridos. As Árvores Rubro-Negras, com sua altura potencialmente maior, podem ter um desempenho mais lento nessas operações.
c. **Eficiência nas operações de conjunto**: nas Árvores AVL, a eficiência na inserção é crucial para operações de conjunto como união e interseção. Ao realizar a união de dois conjuntos, a inserção de elementos do segundo conjunto no primeiro é mais rápida devido à menor altura da árvore AVL. Em contraste, as Árvores Rubro-Negras, com sua altura maior, podem ter uma eficiência menor nessas operações devido ao maior número de níveis a serem percorridos durante a inserção.
d. **Otimização da complexidade de operações**: operações como união, interseção, inserção e remoção têm sua complexidade otimizada em Árvores AVL devido à altura logarítmica. Isso significa que o tempo para realizar estas operações cresce de forma logarítmica com o tamanho do conjunto, o que é eficiente para conjuntos grandes. As Árvores Rubro-Negras, embora ainda tenham uma complexidade logarítmica, podem ser menos eficientes devido à sua altura potencialmente maior.
e. **Melhor desempenho em leitura intensiva**: Em aplicações focadas em leituras frequentes, como verificações de pertinência ou impressão de elementos, a AVL se destaca. Em contraste, as Árvores Rubro-Negras podem ter um desempenho inferior em tais operações devido à sua altura maior, o que requer mais tempo para percorrer a árvore.
f. **Rebalanceamento eficiente**: embora as Árvores AVL necessitem de mais rotações para manter o balanceamento após inserções e remoções, comparadas às Árvores Rubro-Negras, o impacto negativo é mitigado pela altura mais baixa da árvore, o que mantém a eficiência geral das operações. As Árvores Rubro-Negras, embora necessitem de menos rotações, podem ter um desempenho geral inferior devido à sua maior altura.

g. **Adequação ao requisito do projeto**: o projeto exige uma estrutura de dados que otimize a complexidade computacional de operações específicas de conjuntos. A AVL, com sua altura logarítmica e balanceamento rigoroso, fornece um bom equilíbrio entre eficiência operacional e complexidade computacional, tornando-a uma escolha adequada para o projeto.

Análise da complexidade Big O das funções:

avl_cria, set_criar, item_criar, no_criar (Complexidade: O(1)):

- Justificativa: essas funções são simples alocações de memória (malloc) e configurações iniciais, como atribuir valores iniciais a ponteiros e variáveis. Como estas operações têm um número fixo de passos, independentemente do tamanho da árvore ou conjunto, a complexidade é constante, O(1).

avl_inserir, set_inserir (Complexidade: O(log n)):

- Justificativa: a inserção em uma AVL começa com a busca pela posição correta, o que requer percorrer a altura da árvore (O(log n)). Após a inserção, podem ser necessárias rotações para manter o balanceamento da árvore. Estas rotações também são dependentes da altura da árvore, mantendo a complexidade em O(log n).

avl_remover, set_remover (Complexidade: O(log n)):

- Justificativa: semelhante à inserção, a remoção começa com uma busca pelo nó a ser removido (O(log n)). Ajustes podem ser necessários para manter o balanceamento da árvore após a remoção, o que também é baseado na altura da árvore, mantendo a complexidade em O(log n).

avl_busca, set_pertence (Complexidade: O(log n)):

- Justificativa: a busca em uma árvore AVL é realizada percorrendo a árvore da raiz até o nó desejado. Devido ao balanceamento da árvore, a altura máxima é logarítmica em relação ao número de nós, resultando em uma complexidade de busca de O(log n).

avl_apagar, set_apagar, no_apagar, item_apagar (Complexidade: O(n)):

- Justificativa: para apagar a árvore, é necessário percorrer todos os nós e liberar a memória associada a cada um. Esta operação requer uma visita a cada nó uma vez, resultando em uma complexidade linear, O(n).

avl_imprimir (Complexidade: O(n)):

- *Justificativa: imprimir todos os elementos da árvore AVL requer percorrer cada nó* uma vez (através de uma travessia em ordem, pré-ordem ou pós-ordem), o que tem uma complexidade linear, O(n), pois todos os nós são visitados.

set_uniao, set_interseccao (Complexidade: O(n log n)):

- Justificativa: para unir ou intersectar dois conjuntos, é preciso percorrer todos os elementos de ambos os conjuntos (em uma mesma interação) e realizar inserções

na nova árvore AVL. Cada inserção tem uma complexidade de O(log n), e como isso é feito para cada elemento dos conjuntos, a complexidade total é O(n log n).

As análises de complexidade mostram como as operações de inserção, remoção e busca são afetadas pela altura logarítmica da árvore, enquanto as operações que requerem percorrer toda a árvore têm uma complexidade linear. A união e interseção de conjuntos têm uma complexidade maior devido à necessidade de realizar múltiplas inserções.

Conclusão

Portanto, a escolha da Árvore AVL para este projeto é justificada pela sua capacidade de garantir uma altura menor, o que se traduz em eficiência nas operações de conjunto. Além disso, a AVL atende aos requisitos do projeto de otimizar a complexidade computacional, sendo uma escolha mais adequada em comparação com a Árvore Rubro-Negra para este caso específico.