Algoritmos e Estrutura de Dados I

Kayky Sena
Miller Anacleto
Jorge Salhani

I. Descrição do Problema

A representação e realização de operações sobre conjuntos faz parte do ramo da matemática denominado Teoria dos Conjuntos. Como visão geral, a importância de seu estudo deve-se à possibilidade de generalizar conceitos matemáticos a quaisquer conjunto de objetos, abstratos ou não.

Um conjunto pode ser determinado por um conjunto finito ou infinito de elementos sobre os quais certas propriedades e operações são definidas. Algumas das mais básicas que temos são: pertencimento, união e intersecção.

Sejam, por exemplo, dois conjuntos $A = \{x; x \in \mathcal{N}\}$ e $B = \{x; x = 2n, n, x \in \mathcal{Z}\}$. Ou seja, A é o conjunto de todos os números naturais $A = \{0, 1, 2, 3, ...\}$ e B, o conjunto de todos os números inteiros pares $B = \{..., -4, -2, 0, 2, 4, 6, ...\}$.

Dadas as operações básicas citadas acima, temos que $A \cup B$ representa a união dos conjuntos A e B, e $A \cap B$, a intersecção entre eles de modo que

$$\begin{aligned} A \cup B &= \{x; (x \in A) \cup (x \in B)\} \\ &= \{..., -4, -2, 0, 1, 2, 3, 4, 5, 6, ...\} \\ A \cap B &= \{x; (x \in A) \cap (x \in B)\} \\ &= \{0, 2, 4, 6, ...\} \end{aligned}$$

E também, quando dispomos de um elemento particular, podemos verificar se o mesmo está presente em um determinado conjunto pelo conceito de pertencimento. Aqui vale lembrar que esta operação vale tanto para elementos e quanto para conjuntos. Logo, sendo $a = 3$, $a \in A$ e $a \notin B$, assim como para $C = \{x; x \in \mathcal{Z}\}$, temos que $B \in C$.

Neste trabalho vamos adotar apenas estas operações definidas acima, focando na estrutura de dados que, em complexidade de algorítimo e tempo de execução para as operações, melhor representam conjuntos finitos.

II. Modelagem da Solução

Nos termos mencionados na seção anterior, escolhemos o tipo abstrato de dados (TAD) AVL para representar os conjuntos. Árvores AVL (Adelson-Velsky Landis, seus criadores) são árvores binárias de busca com reestruturação dinâmica de nós a cada inserção e/ou remoção de elementos de modo a mantê-la sempre balanceada.

Como uma vez representados os conjuntos suas operações decorrem imediatamente por meio de buscas, aqui optamos por detalhar a estrutura geral de uma árvore AVL. Em primeiro lugar precisamos definir profundidade de uma sub-árvore como o maior caminho de sua raiz até os nós-folha, ou seja, aqueles que não apresentam sub-árvores. Sendo n o número de nós da sub-árvore AVL, sua profundidade D pode ser obtida pela função

$$D = \lceil log_2(n + 1) \rceil$$

Agora podemos definir as duas características centrais de uma AVL. Esse TAD é uma árvore binária de busca (ABB), pois cada nó (elemento da árvore) apresenta no máximo duas ramificações ou sub-árvores, sendo o valor relacionado à raiz da sub-árvore esquerda é menor que o valor relacionado à raiz da sub-árvore direita. E uma AVL é uma árvore balanceada, visto que, dado um nó qualquer nela contido, a diferença da profundidade de suas sub-árvores esquerda e direita é no máximo 1, em valor absoluto. Na Figura 1 apresentamos exemplos de árvores para melhor entender as características de uma AVL.

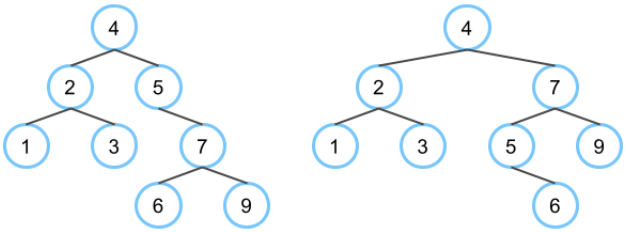

Figura 1: Representação de uma árvore binária de busca (ABB) não balanceada (à esquerda) e desta mesma árvore após balanceamento, sendo agora, portanto, uma AVL (à direita).

Na seção seguinte discutiremos os motivos pelos quais escolhemos este TAD AVL para a representação dos elementos de um conjunto. Agora, no entanto, vamos analisar os TADs utilizados e o funcionamento das operações de união, intersecção e pertencimento.

Neste trabalho utilizaremos três TADs de nome ITEM, SET e AVL. O TAD ITEM representa o conteúdo de cada nó da árvore AVL, o qual se relaciona aos elementos de cada conjunto. Como os conjuntos serão restritos a conter números inteiros, o TAD ITEM apenas contém um inteiro, além das operações essenciais de interface, tais como a criação de um item e o resgate de seu conteúdo.

O TAD SET contém a interface que permite as operações obre os conjuntos dados, sendo que sua estrutura contém apenas o TAD AVL. Na próxima seção também explicaremos o motivo da existência de um TAD SET intermediário.

Por fim, o TAD AVL contém a estrutura vinculada à árvore AVL, assim como as operações padrão de inserção, remoção e busca. Vale lembrar também que cada item do tipo ITEM está contido na árvore AVL por meio de um TAD NO, responsável por armazenar o item e por conectar a raiz (NO) de cada uma de suas sub-árvores esquerda e direita. Na figura 2 esquematizamos as dependências de cada TAD para melhor compreensão.

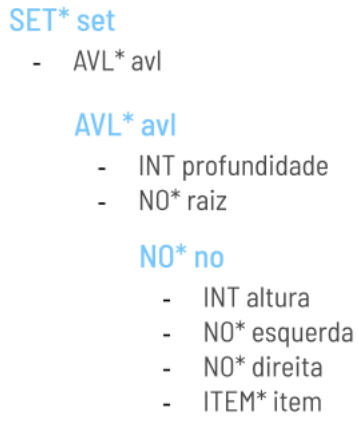

Figura 2: Representação esquemática das dependências do TAD SET, assim como seus tipos. SET é composto pelo TAD AVL, que por sua vez contém o TAD NO, que contém o TAD ITEM.

Para o modelo aqui proposto, é esperado que as informações de entrada para execução do código esteja no formato a seguir. Vale destacar a notação $|C|$ representando a cardinalidade de C, ou seja, o número de elementos que estão contidos no conjunto C.

5	$\leftarrow \|A\|$
3	$\leftarrow \|B\|$
1 2 3 4 5	$\leftarrow A$
7 2 8	$\leftarrow B$
1	$\leftarrow$ operação
3	$\leftarrow$ elemento

O campo 'operação' pode ser 1: pertence; 2: união; 3: intersecção, 4: remoção. Caso 2 ou 3, o campo 'elemento' é nulo, pois ambas dependem apenas dos conjuntos A e B operados. Já para 1 e 4, é necessário informar qual elemento será analisado. Para a operação 1, seja $a = 3$ (caso exemplo acima), verificamos se é válido que $a \in B$. Para a operação 4 (extra), seja $a = 3$, executamos a remoção de modo que $A - \{a\}$.

Conforme os elementos de cada conjunto A e B são lidos, suas respectivas árvores AVL são construídas para representá-los. As figuras 3 e 4 mostram a evolução da construção para A e B exemplificados acima, respectivamente.

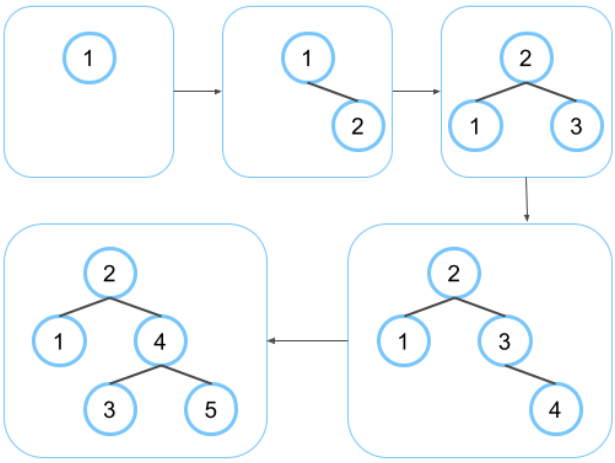

Figura 3: Representação da evolução da árvore AVL conforme os elementos de $A = \{1, 2, 3, 4, 5\}$ são inseridos.

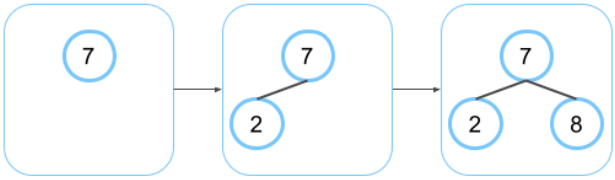

Figura 4: Representação da evolução da árvore AVL conforme os elementos de $B = \{7, 2, 8\}$ são inseridos.

Uma vez construídas as árvores AVL para os conjuntos A e B, as operações 1 e 4 executam a busca binária do elemento desejado, retornando se o elemento pertence ou não (caso 1) ou retornando um novo conjunto, removido o elemento informado (caso 4).

Já as operações 2 e 3, respectivamente, copia a árvore AVL relativa ao conjunto A e nela inserem todo elemento presente no conjunto B que já não esteja contido em A (caso 2), ou para cada elemento da árvore AVL relativa ao conjunto A, verifica se o mesmo pertence à árvore AVL do conjunto B, inserindo em uma nova árvore criada.

Fica evidente que todas as 4 operações realizam buscas binárias ou inserções com possível balanceamento, padrão para TAD AVL. Na seção seguinte vamos analisar a complexidade de cada uma das operações e justificar as escolhas dos TADs utilizados ao longo deste trabalho.

III. Discussão

Voltemos ao início da seção anterior, onde definimos os TADs utilizados. A escolha da criação de um TAD SET composto apenas por um TAD AVL é justificada pela possibilidade de desacoplamento da interface de uso das operações entre dois conjuntos A e B da sua forma de representação computacional. Caso uma nova funcionalidade seja implementada tal que faça preferível uma outra representação de conjuntos distinta de uma árvore AVL, as funções de interface continuariam idênticas, alterando apenas o TAD AVL.

Por outro lado a escolha do TAD árvore AVL como representação dos conjuntos se mostra interessante quando avançamos ao fim da seção anterior, onde descrevemos brevemente como as operações de união, intersecção, pertencimento e subtração são executadas. Para todas as operações que estudamos neste trabalho são executadas buscas, cuja maior eficiência (com exceção às ABBs perfeitamente balanceadas) ocorre em árvores binárias de busca balanceadas, ou seja, em AVL.

Vale destacar também que, por definição, conjuntos apresentam elementos únicos e que, quando finitos, são enumeráveis. Como árvores AVL também devem garantir a unicidade de seus elementos e sabendo que o conjunto dos números inteiros, sem repetição, representa adequadamente quaisquer conjuntos, isto representa um fator que direciona à escolha de árvores AVL para a solução eficiente do problema deste trabalho.

Ao utilizarmos o TAD AVL, o maior custo que teremos ao realizar uma das 4 operações terá complexidade $\mathcal{O}(n) \sim n \log_2 m$, pois para cada $x \in A$, com $|A| = n$, devemos performar uma busca no conjunto B tal que $|B| = m$, cuja complexidade é $\mathcal{O} \sim \log_2 m$, que ocorre para a operações de união e intersecção. De fato a complexidade é logarítmica para árvores AVL, pois a ordenação adequada de itens (em relação à raiz da sub-árvore considerada, o nó-filho à sua esquerda contém um item com valor numérico menor e o nó-filho à sua direita, maior) permite que a cada iteração da busca aproximadamente metade do conjunto seja descartada. Já avaliar a afirmação $x \in A$ quando representamos A como um TAD AVL é equivalente à operação de busca simples em uma árvore AVL, refletido na complexidade assintótica de $\mathcal{O}(n) \sim \log_2 n$.

Nas figuras 6 e 5 fica evidente que o tempo de execução de cada uma das operações segue as complexidades assintóticas descritas acima, onde, por simplicidade, assumimos que conjuntos com grande cardinalidade (n grande), $|A| \sim |B| = n$.

Na figura 5, seja $C = A, B$, notamos que as operações de inserção ($C - \{x\}$), remoção ($C - \{x\}$) e impressão apresentam apenas uma pequena distinção, aproximadamente na ordem de 10^{-4}, entre seus respectivos tempos de execução.

Por fim é interessante destacarmos a diferença observada entre o tempo de execução entre as operações $2 : A \cup B$ e

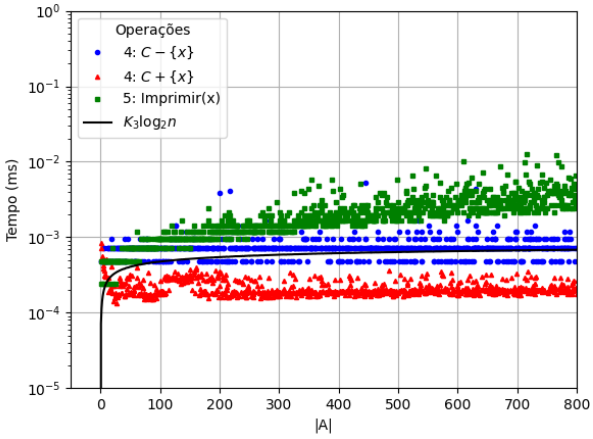

Figura 5: Tempo de execução (ms) em relação ao número de elementos do conjunto A (i.e., $|A|$) para as operações de remoção ($C - \{x\}$), inserção ($C + \{x\}$) e impressão (Imprimir(x), com $C = A$. A curva $K_3 \log_2 n$ em linha preta contínua representa os pontos para o caso da remoção, cuja constante multiplicativa é $K_3 \approx 7.14(10^{-5})$.

$3 : A \cap B$, visto na figura 6. Notamos que $T(2) > T(3)$, ou seja, o tempo de execução da rotina de união é maior que o tempo de execução para a intersecção. Para isso, vamos separar ambos os algoritmos da seguinte forma:

Seja C a árvore AVL resultado. Caso $A \cup B$:

1) Para cada $x \in A$, $C + \{x\}$.
2) Para cada $y \in B$, se $y \notin B$, então $C + \{y\}$.

E caso $A \cap B$:

1) Para cada $x \in A$, se $x \in B$, então $C + \{x\}$.

Fica evidente que o número de inserções (operação $C + \{k\}, k = x, y$) é maior para o caso $A \cup B$. Com os resultados que obtivemos na figura 6, a razão entre o tempo de execução de cada operação é de 0.1913. Em outras palavras, performar operação de intersecção é aproximadamente 5 vezes mais rápido quando comparado à operação de união.

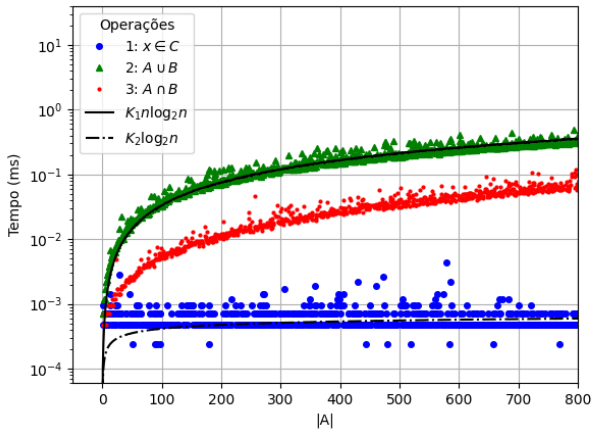

Figura 6: Tempo de execução (ms) em relação ao número de elementos do conjunto A (i.e., $|A|$) para as operações de pertencimento $(x \in C)$, com $C = A$, união $(A \cup B)$ e intersecção $(A \cap B)$. As curvas $K_1 n \log_2 n$ que melhor representam os pontos para as respectivas operações apresentam constante multiplicativa $\approx 4.545(10^{-5})$ (para $A \cup B$, explícita no gráfico, em linha preta contínua) e $\approx 8.696(10^{-6})$ (para $A \cap B$), de modo que sua razão é ≈ 0.1913. Para a curva $K_2 \log_2 n$, temos a constante $\approx 6.25(10^{-5})$ (para $x \in C$, explícita no gráfico em linha preta tracejada).

IV. Conclusão

Concluímos, com este trabalho, a viabilidade da representação de conjuntos utilizando um TAD árvore AVL. Com ela, é possível otimizar a performance das operações centrais de conjuntos, tais como intersecção, união e pertencimento, além de otimizar operações de modificação dos conjuntos, como inserção e remoção de elementos. Embora conjuntos sejam mais imediatamente pensados como uma sequência de elementos, facilmente representado e implementado como listas, sua representação em AVL apresenta ganhos de performance bastante expressivos conforme discutimos ao longo deste trabalho.