Universidade Federal do Piauí – UFPI
Campus Senador Helvídio Nunes de Barros – CSHNB
Sistemas de Informação – Estruturas de dados II – Prof. Juliana Oliveira

Relatório do segundo trabalho de Estruturas de dados II

Aluno: Marcos Vinicius Batista Sampaio

Resumo:

Este relatório trata da descrição detalhada do funcionamento dos códigos implementados para responder as questões passadas como trabalho avaliativo que conta com cinco questões.

Introdução:

Este código implementa uma árvore 2-3, uma estrutura de dados que permite armazenar e recuperar informações de forma eficiente. A árvore 2-3 é uma árvore de busca que é capaz de armazenar um número variável de chaves em cada nó, o que a torna uma estrutura bastante flexível. A implementação inclui funções para criar e inserir nós na árvore, bem como outras funções auxiliares. O código está escrito em linguagem C e utiliza ponteiros para manipular a estrutura da árvore. O trabalho passado para ser resolvido consistia em criar um gerenciador de memoria onde se tem os status de livre e ocupado e assim fazer que seja alternado pelo sistema entre os mesmos automaticamente, além do usuário poder desalocar os espaços, porém nesse trabalho contem apenas a alocação.

Seções Específicas:

Este código implementa uma Árvore 2-3, uma estrutura de dados de árvore que é uma generalização de uma Árvore B para acomodar chaves duplicadas. A árvore tem nós que podem ter uma ou duas chaves e entre um e três filhos. O objetivo é manter a árvore equilibrada e garantir que as chaves estejam ordenadas.

- A estrutura “Informations! armazena informações sobre o intervalo de chaves que um nó contém. A estrutura Informations define um tipo de dado composto com os seguintes campos:

status: um caractere que indica o status de um registro de dados sendo L ou O.
inicioInicio: um número inteiro que representa o início do registro no arquivo.
finalFinal: um número inteiro que representa o fim do registro no arquivo.
inicio: um número inteiro que representa o início de um intervalo de dados.
final: um número inteiro que representa o fim de um intervalo de dados.

- A estrutura “Arv23” é a estrutura do nó da Árvore 2-3 e contém ponteiros para os filhos e informações sobre as chaves armazenadas no nó.
A estrutura é definida com o nome Arv23 e possui seis campos:

info1 e info2: armazenam as informações contidas nos nós da árvore. Na implementação apresentada, a árvore é utilizada para armazenar intervalos, por isso as informações armazenadas na árvore são do tipo Informations.
nkeys: armazena a quantidade de chaves contidas no nó. Para uma árvore 2-3, nkeys pode ter valor 1 ou 2.
esquerda, centro e direita: ponteiros para os filhos do nó. Dependendo do valor de nkeys, o nó pode ter 1 ou 2 filhos. Quando nkeys é 1, apenas esquerda e centro são utilizados. Quando nkeys é 2, esquerda, centro e direita são utilizados.

pai: ponteiro para o pai do nó.
Essa estrutura é utilizada para implementar a Árvore 2-3 e armazenar os intervalos e seus limites superior e inferior.

- A função “criaNo” cria um novo nó com as informações passadas e retorna um ponteiro para o novo nó. Essa função cria um nó de uma árvore 2-3, que é uma estrutura de dados em que cada nó pode ter até 3 filhos e armazena uma ou duas chaves.

A função recebe como parâmetros as informações que serão armazenadas no nó, bem como ponteiros para os seus possíveis filhos esquerdo (FE), centro (FC) e direito (FD).

Em seguida, é alocado um espaço na memória para o novo nó e as informações, filhos e número de chaves são atribuídos ao nó. Por fim, a função retorna o ponteiro para o novo nó criado.

- A função “ehfolha” retorna verdadeiro se o nó passado é uma folha (não tem filhos). A função "ehfolha" recebe como entrada um ponteiro para um nó da árvore 2-3 e verifica se esse nó é uma folha ou não. Ela faz isso verificando se os ponteiros para os nós esquerdo, central e direito do nó de entrada são nulos. Se todos os ponteiros forem nulos, o nó é uma folha e a função retorna 1 (verdadeiro). Caso contrário, a função retorna 0 (falso), indicando que o nó não é uma folha.

- A função “quebraNo” quebra um nó em dois quando o nó já possui duas chaves e uma nova chave deve ser inserida. A chave intermediária é retornada em sobe e o novo nó criado é retornado pela função. A função recebe como entrada um ponteiro para o nó que precisa ser dividido, a chave a ser inserida, um ponteiro para armazenar a chave que será "promovida" e um ponteiro para o filho que será adicionado ao novo nó.

A função começa verificando em qual posição a nova chave deve ser inserida no nó atual. Se a nova chave é menor que a primeira chave do nó atual, ela se torna a nova primeira chave e a segunda chave é movida para um novo nó. Se a nova chave está entre as duas chaves do nó atual, ela se torna a segunda chave do nó atual e a segunda chave é movida para um novo nó. Se a nova chave é maior que a segunda chave do nó atual, ela se torna a nova segunda chave e é movida para um novo nó.

Depois de separar as chaves em dois novos nós, a função cria um novo nó e retorna um ponteiro para ele. Esse novo nó terá como chaves a chave que foi "promovida" e dois novos filhos, que são o filho passado como parâmetro e o antigo filho direito do nó atual.

Por fim, a função atualiza o nó atual para que ele tenha apenas uma chave e seja responsável por apontar para os dois novos nós criados.

- A função “adicionaNo” adiciona uma nova chave a um nó que já possui uma chave e um filho. A nova chave e o filho são adicionados ao nó e a árvore é atualizada. Ela recebe um ponteiro para um nó 2-3 (representado por **No), uma informação info e um ponteiro para um nó filho filho. A função começa comparando o início da informação info com o final da informação armazenada em (*No)->info1. Se info.inicio for maior que (*No)->info1.final, então a informação info deve ser armazenada em (*No)->info2 e o nó filho deve ser armazenado em (*No)->direita. Caso contrário, a informação (*No)->info1 deve ser armazenada em (*No)->info2, o nó filho de (*No)->centro deve ser armazenado em (*No)->direita, a informação info deve ser armazenada em (*No)->info1 e o nó filho filho deve ser armazenado em (*No)->centro.

Depois disso, o número de chaves armazenadas no nó é atualizado para 2. A função não faz nenhuma verificação de se o nó está cheio ou não, É chamada apenas após a verificação de que o nó tem espaço para armazenar mais informações.

- A função “insere” insere uma nova chave na árvore. Se a raiz estiver vazia, a chave é inserida na raiz. Se a raiz for uma folha e já tiver duas chaves, o nó é dividido em dois. Se a raiz não for uma folha, a função é chamada recursivamente no filho apropriado. Se a função retornar um ponteiro, o nó é dividido novamente. Quando a função retorna, a árvore é atualizada para acomodar a nova chave e manter o equilíbrio. A função insere é responsável por inserir um novo nó em uma árvore 2-3. Ela recebe como parâmetros um ponteiro para a raiz da árvore (Raiz), uma estrutura Informations que representa a informação a ser inserida na árvore (info) e um ponteiro para o nó pai (pai).

Inicialmente, a função verifica se a árvore está vazia ou se o nó atual é uma folha. Caso a árvore esteja vazia, ela cria um novo nó com a informação a ser inserida e retorna esse nó. Se o nó atual é uma folha, ela verifica se o nó tem uma chave ou duas chaves. Se o nó tiver apenas uma chave, a informação é adicionada ao nó. Caso contrário, a função chama a função quebraNo para dividir o nó em dois, e retorna o maior nó resultante da quebra.

Se o nó atual não for uma folha, a função percorre a árvore recursivamente, verificando se a informação a ser inserida deve ser adicionada ao nó esquerdo, centro ou direito, dependendo do valor da chave.

Após a recursão, a função verifica se houve necessidade de dividir algum nó durante o processo de inserção. Se não houve, a função simplesmente retorna NULL. Caso contrário, a função verifica se o nó pai é nulo (o que significa que a divisão ocorreu na raiz da árvore) e cria um novo nó com a chave sobressalente e os dois nós resultantes da divisão. Em seguida, retorna NULL.

Conclusão:

Conforme descrito anteriormente, a aplicação da árvore 2-3 apresenta muitos benefícios no que diz respeito ao gerenciamento de memória. Embora não seja uma estrutura de árvore tão simples, ela possui uma vasta ramificação de tamanhos. Com a aplicação adequada, foi possível armazenar uma grande quantidade de memória com acesso mais fácil e eficiente.