Introdução

O problema abordado para o trabalho é a criação de uma treap, que é uma árvore binária de pesquisa com uma heap, onde se deve implementar funções de remoção e inserção usando as funções de corte e fusão, e uma função de pesquisa.

Para a solução do problema foi utilizado na função de inserção uma manipulação de ponteiros, e na remoção, o programa percorre a árvore e vai dividindo a mesmo em outras duas árvores, depois apaga-se a árvore principal e faz a fusão das duas novas árvores criadas.

Solução do Problema

Para que esse trabalho prático fosse concretizado foi necessário a utilização da "treap", e a partir disso, funções que manipulam ponteiros foram criadas a fim de construir uma arvore binária com balanceamento.

A partir de prioridades, essa arvore utiliza de um método, que insere as chaves função definida por "insere", ela vai comparar a chave a ser inserida a partir da sua prioridade com o nó atual, se o nó atual for a "cabeça", a partir da estrutura condicional "if", a "cabeça" terá seus ponteiros alterados de acordo com o resultado da comparação entre eles, pois a chave atual se for menor; ela fica à esquerda e se for maior ela vai para direita. Dessa forma será alocado um espaço na memória onde essa nova chave será direcionada e assim modificando os ponteiros de seu antecessor. Se houver outras chaves, serão feitas comparações dentro da função inserir entre as chaves até ser definido local exato onde esta será inserida, neste caso os ponteiros de seus sucessores e antecessores serão modificados da mesma forma como dita anteriormente. Se não existir nenhuma chave, o programa chama a função inicializar e aloca um espaço na memória e cria os seus ponteiros.

Em outra função, contraria a anterior, a "Remove" recebe como parâmetro a chave a ser removida da arvore, inicialmente irá ocorrer um teste para saber se a chave atual existe, se sim, entra a parte em que o problema é resolvido, pois será criado uma copia das chaves para duas novas arvores, estas duas novas arvores serão definidas de forma que as chaves menores vão para arvore A, e as maiores vão para arvore B, está novas arvores terão novos ponteiros e serão independentes da arvore antiga, a função corte é o ponto crucial do trabalho, pois é ela que faz a modificação de ponteiros alocando novos espaços e redefinindo as novas arvores criadas a partir da remoção de uma determinada chave, ela vai usar também a função insere para que as chaves já existem sejam mantidas da mesma forma, mas agora em duas novas arvores. Após todo esse processo de criação e manipulação dos novos

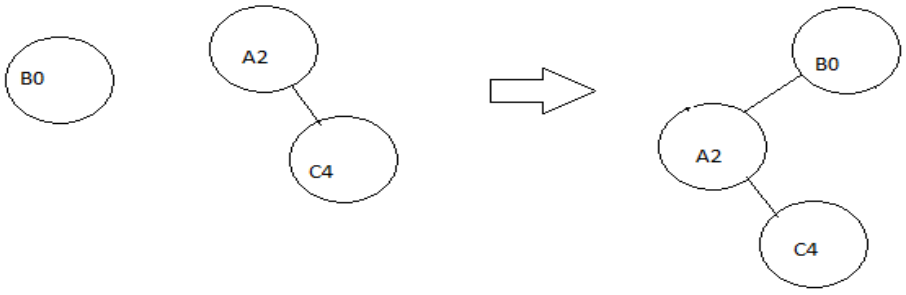
ponteiros, a arvore antiga deixa de existir na memória, exceto a sua cabeça pois ela será reutilizada para o uso da fusão entre as duas novas arvores.

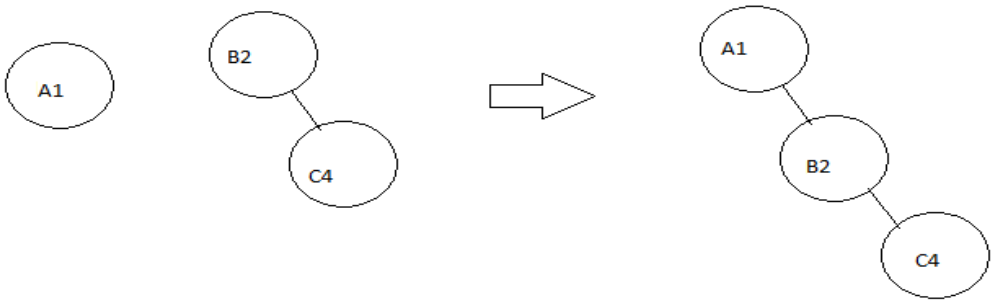
A resolução final do problema chega na função "fusão" que é inicializada, e nesta será feita uma comparação entre as duas novas arvores, para definir quem será a nova cabeça da arvore "p", usando as prioridades, se a prioridade da arvore da menor for menor que a da arvore maior, o ponteiro irá percorrer a direita da árvore menor até encontrar o nulo, quando for verdadeiro, esse nó vai ser modificado, esse nó vai apontar para a arvore maior. Caso contrário, verifica-se a esquerda da arvore maior é nulo, e assim como no caso anterior, o ponteiro vai percorrer até achar o nulo, quando encontrado, a esquerda do nó atual será manipulada para apontar para o menor e p apontar para a cabeça da arvore maior. Assim a nova arvore é criada, sendo que esta já estará balanceada, e respeitando as prioridades.

O usuário terá a opção de verificar se a chave existe na arvore ou não, com a função "pesquisar", ela utiliza a mesma estrutura condicional if para comparar as chaves com a atual e localizar a chave, ao fim dessa função uma nova função para ajudar o usuário é chamada, "imprime", a função mostra o caminho usada para encontrar a chave, mostrando na tela as letras R ou L dizendo se a partir da cabeça a chave fica a Rigth (direita) ou Lefth (esquerda), e assim sucessivamente.

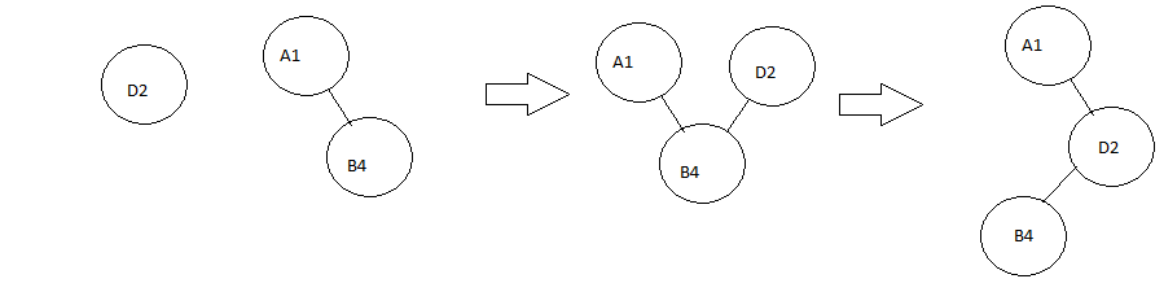
Exemplos:

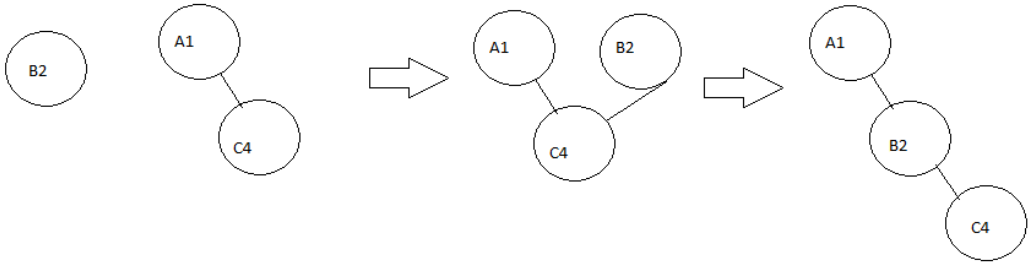
- Inserção cabeça:



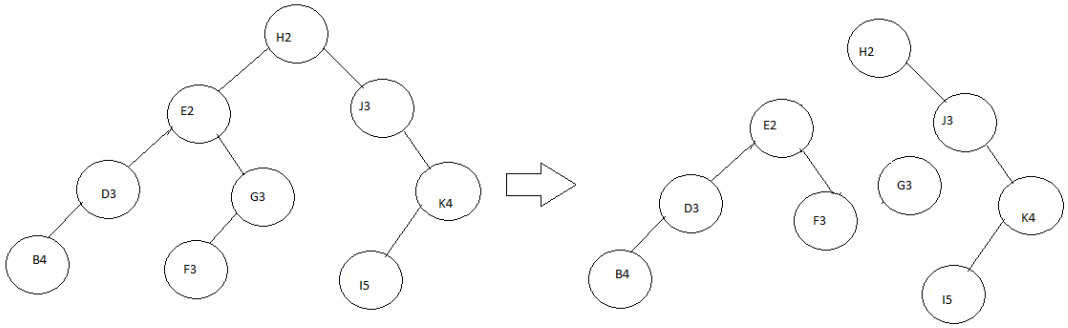


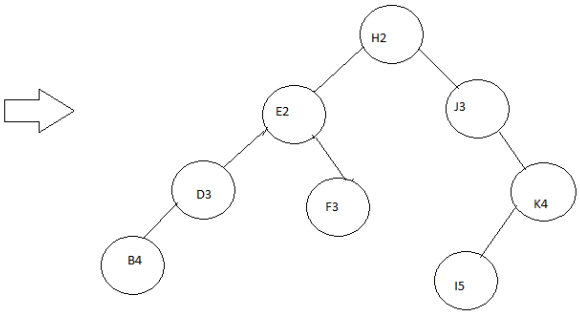
- Inserção meio:





- Corte Fusão





Considerações Finais

O problema do trabalho realizado dificultou muito a implementação do código, principalmente a parte de corte e fusão da árvore binária, onde, por mexer com muitos ponteiros, a aplicação fica difícil por conta da grande chance de se perder os ponteiros ou a cabeça da árvore, e também por ter que implementar uma heap, as inserções tem uma complexidade maior por conta da prioridade.