Documentação da implementação de árvore AVL

tNo: Estrutura dos nodos, guarda uma chave, o nível do nó e o balanceamento(altura de esq - altura da dir), além dos ponteiros para o nó a esquerda, a direita e o nó pai.

criaNo: aloca espaço para um nó, e inicia ele com a chave, os outros elementos do no são vazios.

Busca: Algoritmo padrão de busca binária, usando um nó de início, a raíz da árvore, e a chave a ser buscada.

Altura: calcula as alturas da subarvore esquerda e direita e retorna a maior das duas.

calcBalanc: calcula o fator de balanceamento, que é a altura da esquerda(he) - altura da direita(hd), pode ser positivo (subarvore esquerda > subarvore direita) ou negativo(subarvore direita > subarvore esquerda).

diminui/aumentaNivel: diminui/aumenta o nível de todos os nodos de uma árvore/subárvore, usa o metódo pós-ordem(esq, dir, pai).

rotEsq/Dir : rotação para a esquerda ou direita, seguem os algoritmos normais, mas no final recalcula o fator balanceamento de x e y, e ajusta o nivel dos nodos/subárvores que mudaram de posição.

Balanceamento: calcula o fator balanceamento do no.
-Caso seja > 1(desbalanceado na esquerda), pode ser:
1)zigzig para a esquerda (chave < no->esq->chave) : uma rotação a direita;
2)zigzag para a direita (chave > no->esq->chave): uma rotação a esquerda de z, depois a direita de x;

-Caso seja < -1(desbalanceado na direita) pode ser:
1)zigzig para a direita (chave > no->dir->chave) : uma rotação a esquerda;
2)zigzag para a esquerda (chave < no->dir->chave): uma rotação a direita de z, depois a esquerda de x;

Inclusão: Segue o algoritmo de inclusão, inclui chaves menores na esquerda e maiores na direita, o nível de cada novo nodo é o nível do pai mais um, pois está mais "baixo" na árvore. Antes de retornar/"subir" de nó, analisa o balanceamento do nó atual.

Transplante: segue o algoritmo padrão de transplante, mas no final, arruma o nível do nó transplantado(v) para ser o igual o do anterior(u).

Antecessor: algoritmo recursivo de antecessor, a entrada é no->esq, então acha o nó mais a direita na esquerda.

Tree-delete: Segue o algoritmo padrão tree-delete, caso o nó a ser removido seja a raíz, a nova raíz é o antecessor, retorna a nova raíz.

Exlcusão: Foi implementado o algoritmo tree-delete, utilizando do transplante de elementos e do algoritmo de antecessor(tree-minimum) para exclusão de uma chave. Realiza o balanceamento "subindo" na árvore, até a raíz, já que uma exclusão pode desbalancear em al-

gum lugar. Tem os seguintes casos:

-lado esquerdo desbalanceado:
    1)subarvore esquerda do nó esquerdo maior que a direita: uma rotação a direita de x;
    2)subarvore direita do nó esquerdo maior que a esquerda: uma rotação a esquerda de z e uma a direita de x.

-lado direito desbalanceado:
    1)subárvore esquerda do nó direito maior que a direita: uma rotação a esquerda de x;
    2)subárvore direita do nó direito maior que a esquerda: uma rotação a direita de z e uma a esquerda de x.

Depois "sobe" na árvore, apontado cima para o pai, até a raíz(cima == NULL), retorna a raíz nova, já que pode ter se alterado com as rotações.

Impressão: Algoritmo de impressão em ordem com pequena modificação para imprimir o nível do nó junto com a chave.

destroiArvore: percorre a árvore em pós-ordem(esq, dir, raíz) deletando os nodos.

Leitura: le um char c e um inteiro n até o fim da stream de dados.