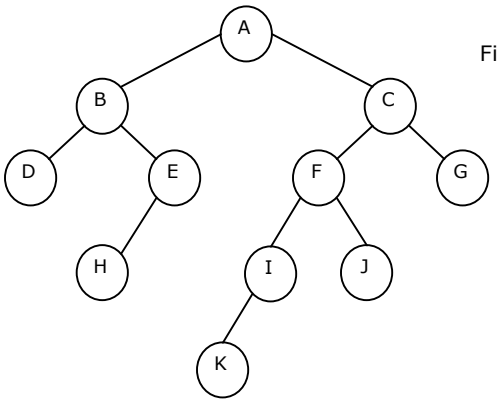
1. **Proponha algoritmos para:**
    a. Calcular a altura dos nós de uma árvore binária dada, armazenando o valor da altura no nó.

    b. Achar o maior elemento (campo numérico) de uma árvore binária dada.

    c. Trocar sub-árvore esquerda e direita de todos os nós.

    d. Percorrer árvore em nível (sugestão: use uma fila como estrutura auxiliar).



Fila: A | B,C | D,E,F,G | H, I, J | K

    e. Calcular o nº de nós das sub-árvores de cada nó, armazenando o resultado no nó correspondente.

2. Desenhar ABB construída pelo algoritmo de inserção impondo a seguinte ordem das chaves: s3, s7, s1, s2, s6, s5, s4; onde S = {s1, s2, ..., s7} e si<si+1, i<7.

3. Provar ou dar contra-exemplo: Sejam P1, e P2 duas permutações de um conjunto de chaves S e T1 e T2 as ABBs correspondentes a P1 e P2 respectivamente. Então P1 != P2 se e somente se T1 != T2.

4. Seja S = {s1, s2, ..., s7} um conjunto de chaves si < si+1, i<7; Desenhar ABB:
    a) com altura máxima
    b) com altura mínima
    c) quantas árvores distintas existem em cada caso?

5. Escrever o algoritmo de inserção em uma ABB, dado como entrada um conjunto de chaves ordenado. A ABB resultante deve ter altura mínima.

6. Responda o que se segue:

a. A que condições deve satisfazer uma árvore B de ordem d para que a inserção de qualquer chave ocasione o aumento da altura da árvore ?
b. A que condições deve satisfazer uma árvore B de ordem d para que a remoção de qualquer chave ocasione a redução da altura da árvore ?
c. Por que a redistribuição deve ser tentada antes da concatenação durante a remoção de uma chave situada em um nó com ocupação mínima de uma árvore B ?
d. Se uma chave não está situada em uma folha de uma árvore B, o que garante que sua sucessora imediata, se existir, estará obrigatoriamente localizada em uma folha ?
e. Qual é o pior caso do algoritmo de inserção de uma chave em uma árvore B de ordem d ? Como deve ser a árvore ?
f. Quais as diferenças entre as árvores B e B+, ressaltando: Quantidade de chaves no nó, localização dos dados, número de acessos e diferença entre nós internos e folhas ?

7. Inserir em uma árvore AVL as chaves: 99, 44, 71, 80, 74, 63, 59, 120, 98, 150, Indique os nós desregulados e as rotações. Exclua as chaves 59 e 63. Mostre a árvore resultante de cada exclusão indicando o nó desregulado e a rotação.

a. Qual o número máximo e mínimo de nós em uma árvore AVL de altura h?

b. A complexidade da exclusão em uma árvore binária de busca é sempre menor do que em uma árvore AVL ?

c. A árvore AVL é uma árvore binária de busca e vice-versa ?

d. Como você implementaria a ordenação de um vetor utilizando árvore AVL ? Compare a complexidade deste algoritmo com o bubble sort e o selection sort.

8. Seja uma árvore AVL T. Considere a inserção de um nó q em T, que tornou T desregulada. Seja p o nó desregulado mais próximo das folhas.

a. Qual o valor exato de |he(p) - hd(p)|? Por que não pode ser nem mais nem menos?

b. Supondo hd(p) > he(p) então existe um filho direito u de p. Por que necessariamente temos |hd(u) - he(u)| = 1? Por que não pode ser 2? Por que não pode ser 0?

c. De acordo com o ítem b, quando hd(p) > he(p) existem dois sub-casos a serem considerados: (i) he(u) = hd(u) +1 ou (ii) hd(u) = he(u) + 1.

Para cada um dos sub-casos acima, apresente a transformação que regula p (diga qual e apresente um esquema). Mostre que realmente todos os nós originalmente em T ficam regulados (através da análise das alturas das subárvores).

d. Por que a regulagem de p (nó desregulado mais próximo das folhas) regula toda a árvore?

9. A estrutura do nó de uma árvore B de ordem 2 é a seguinte:

```
#define MAX 4
#define MIN 2
typedef struct no t_no;
struct no
{
     int       chave[MAX+1];
     t_no *ramo[MAX+1];
     int       ndesc;
};
```

a. Escreva em C uma função **`intervalo(t_no *arv, int lim_inf, int lim_sup)`** que percorre a arvore B (de ordem 2) apontada por **`arv`** em ordem simétrica e imprime todas as chaves x tais que: **`lim_inf < x < lim_sup`**. Assuma que lim_inf ≤ lim_sup.

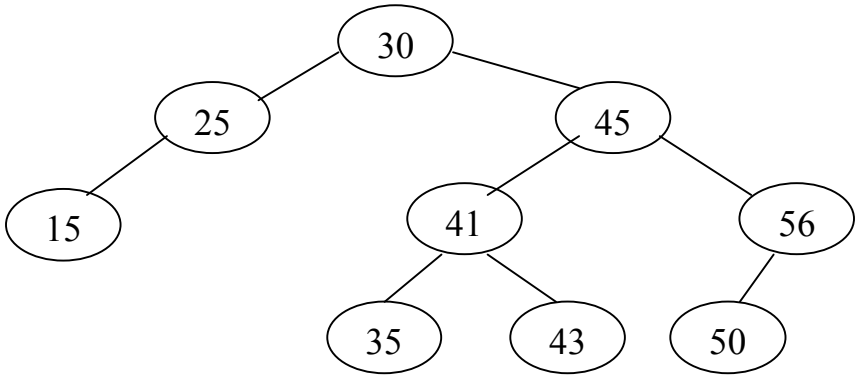
10. Responda Certo ou Errado, justificando brevemente.

a. Para que a inserção de qualquer chave em uma árvore B de ordem $d > 0$ provoque cisão, é necessário que todos os nós tenham ocupação máxima.

b. Uma árvora binária na qual o número de nós em cada nível j é no mínimo 2* (j-1), para j > 1 é balanceada.

c. Durante a inserção de uma chave em uma árvore AVL, o número de rotações eventualmente necessárias para que a árvore se mantenha AVL depende da altura da árvore.

11. Considere a árvore AVL a seguir:



Realize, na árvore acima, as inserções das seguintes chaves 49, 60, 65, e em seguida a remoção das chaves 45 e 41, escolhendo necessariamente imediatamente precedente para a posição da chave removida. Mostre todas as rotações e o formato da árvore após cada operação.

Seja **q** um nó recém inserido e **p** o seu ancestral mais próximo que se tornou desregulado. Quais os possíveis valores para o fator de balanço de **p** após a inserção? Examinar o fator de balanço de **p** é suficiente para concluir se a inserção foi à esquerda ou a direita de p? Por que?

12. Numa AVL, sem utilizar estruturas auxiliares (pilhas, filas, ...) implemente de forma não recursiva a função :

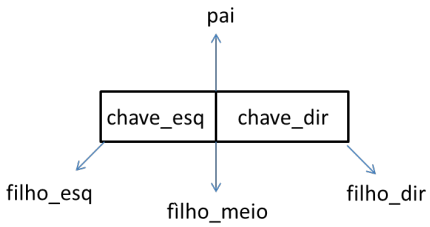
```
int altura(No *raiz);
```

visitando o menor número de nós possível. Use a estrutura:

```
typedef struct _no No;
struct _no {
            int chave;
            int bal; /* fator de balanço: hdir - hesq */
            No  *pai, *esq, *dir;
};
```

13. Considere uma árvore B de chaves inteiras positivas implementada na sua forma mais simples como a árvore 2-3 do trabalho T2. Mostre passo a passo, partindo de uma árvore contendo apenas a chave 19, como ficariam os nós e a estrutura da árvore se nela forem inseridas, em ordem, as seguintes chaves {7, 53, 81, 28, 86,90}.

A figura ao lado ilustra o nó desta árvore:



14. Considere que a árvore 2-3 da questão anterior foi implementada através da seguinte estrutura:

```
typedef struct _bt23 Btree23;
struct _bt23
{
      int    chave_esq,chave_dir;
      Btree23  *pai, *filho_esq, *filho_meio, *filho_dir;
};
```

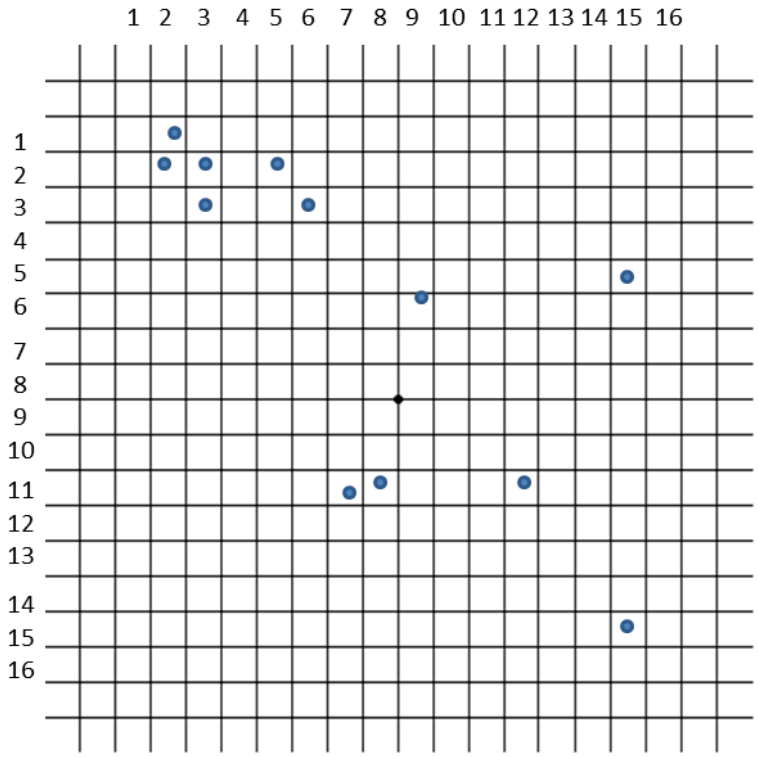
Implemente a função **`bt23_busca`** cujo protótipo está mostrado abaixo que acha de forma não recursiva o nó que contenha a chave dada.

```
Btree23* bt23_busca (Btree23* raiz, int chave);
```

PS-Quando a chave da direita não existir ela tem o valor -1 e o ponteiro do filho da direita é NULL.

15. Descreva como inserir e deletar um ponto em uma quad-tree.

16. Construa a quad-tree para o conjuto de pontos no quadrado abaixo. Em seguida, construa uma quad-tree balanceada para este mesmo conjunto de pontos.



17. Suponha que obriguemos que a condição de balanceamento das quad-trees seja mais severa: quadrados adjacentes não podem mais deferir por um fator de dois em tamanha, mas agora precisam ser iguais, i.e. do mesmo tamanho. O que podemos dizer do tamanho da quad-tree em número de nós? Ainda tem uma proporção linear em relação ao número de nós da quad-tree original?
    Caso contrário, o que pode-se dizer do número de nós na quad-tree?

18. Descreva com inserir e deletar um ponto numa quad-tree.

19. Construa uma kd-tree para o conjunto de pontos da figura da questão 16.

20. Seja P um conjunto de pontos em um espaço d-dimensional. d é uma constante (O(1)). Descreva um algoritmo que construa uma kd-tree d-dimensional para os pontos em P. Analise o seu algoritmo e a necessidade de memória para a estrutura de dados.