9-§. Tartiblangan va muvozanatlashgan daraxtlar

9.1. AVL daraxti

AVL daraxt. AVL-daraxt (inglizcha AVL-Tree) bu muvozanatlashgan ikkilik qidiruv daraxti bo‘lib, unda quyidagi xususiyat qo‘llab-quvvatlanadi: uning har bir uchlari uchun uning ikkita ostki daraxtining balandligi 1 dan ko‘p bo‘lmagan qiymatga farq qiladi.

AVL daraxtlari birinchi marta 1962-yilda AVL daraxtlaridan foydalanishni taklif qilgan G. M. Adelson-Velskiy va E. M. Landisning ismlarining birinchi harflari bilan nomlangan.

Uchlarni muvozanatlash - bu $|h(L) - h(R)| = 2$ chap va o‘ng pastki daraxtlari balandliklari farqi bo‘lgan taqdirda, $|h(L) - h(R)| \leq 2$ daraxtining xususiyati tiklanishi uchun ushbu uchlarning pastki daraxtidagi ajdod va avlod munosabatlarini o‘zgartiradigan amal, aks holda hech narsani o‘zgartirilmaydi. Muvozanatlash uchun biz har bir uch uchun uning chap va o‘ng $diff(i) = h(L) - h(R)$ pastki daraxtlari balandliklari orasidagi farqni saqlaymiz.

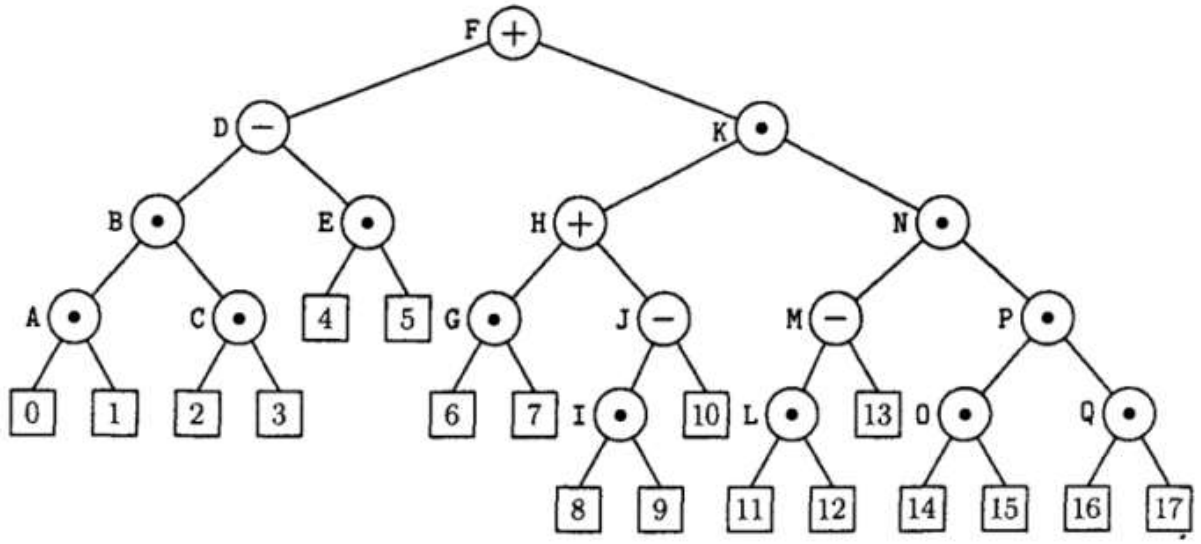
Daraxtning balandligi uning maksimal darajasi, ildizdan tashqi tugunga qadar eng uzun yo‘lning uzunligi sifatida aniqlanadi. Ikkilik qidiruv daraxti muvozanatli deyiladi, agar biron bir tugunning chap pastki daraxtining balandligi o‘ng pastki daraxtning balandligidan ± 1 dan oshmasa. Keyingi 36-rasmda ko‘rsatilgan 5 ta balandlikdagi 17 ta ichki tugunli muvozanatli daraxt; muvozanat koeffitsiyenti har bir tugun ichida belgilar bilan va o‘ng va chap pastki daraxtlar (+1, 0 yoki -1) balandliklari orasidagi farq kattaligiga muvofiq belgilanadi.

Muvozanatlashgan daraxtlar haqidagi teorema. Adelson-Velskiy va Landis quyidagi teoremani isbotladilar:

Teorema. n ichki tugunli muvozanatli daraxt balandligi $\lg(n + 1)$ va $1.4405 \lg(n + 2) - 0.3277$ qiymatlar bilan chegaralangan.

Shunday qilib, biz AVL-muvozanatlangan daraxtdagi qidirish yo‘li mukammal muvozanatlangan daraxtdagi qidirish yo‘lidan 45% dan oshmaydi degan xulosaga kelishimiz mumkin.

AVL daraxtiga yangi tugun kiritilganda paydo bo‘ladigan variantlarni ko‘rib chiqaylik:



36-rasm. AVL ikkilik daraxtiga ko‘ra muvozanatlash

• hl=hR. Yoqilgandan so‘ng, L va R har xil balandliklarga aylanadi, ammo muvozanat mezonlari buzilmaydi;

• hL<hR. Yoqilgandan so‘ng, L va R balandlikda teng bo‘ladi, ya’ni muvozanat mezonlari yanada yaxshilanadi;

• hL>hR. Yoqilgandan so‘ng muvozanat mezonlari buziladi va daraxtni qayta qurish kerak.

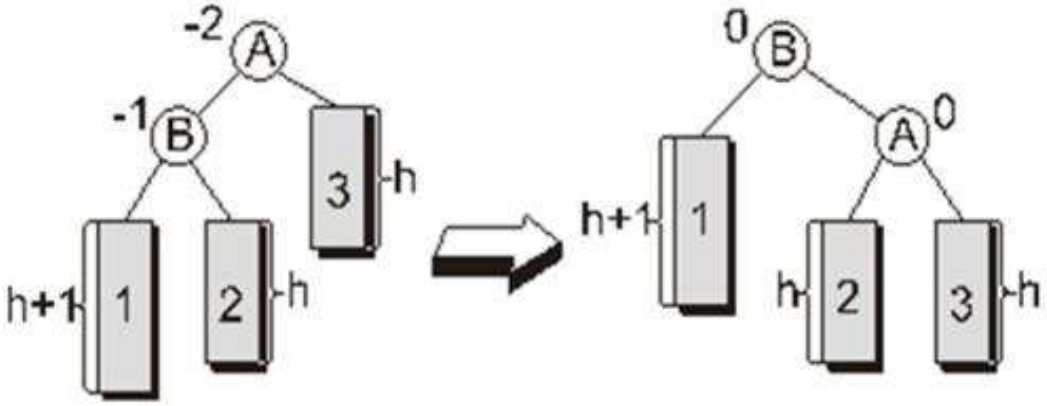
Shunday qilib, biz AVL daraxtiga yangi tugunni kiritish uchun umumiy algoritmni tuzamiz:

1. Kiritilgan daraxtning ichida emasligiga ishonch hosil qilish uchun daraxtni aylanib o‘tish;

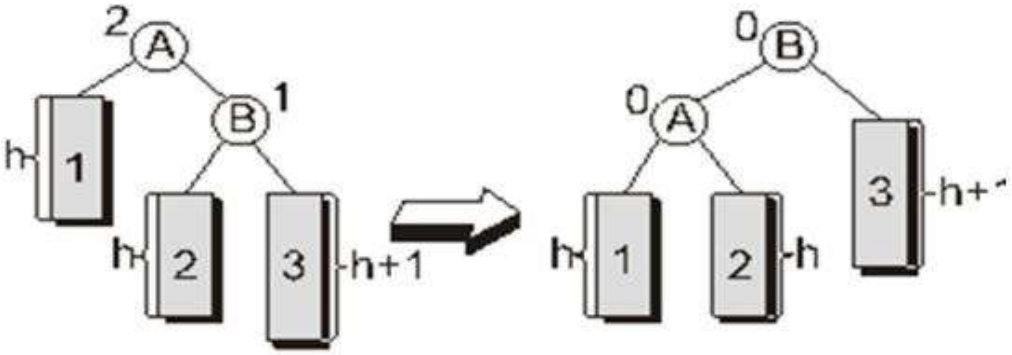
2. Yangi uchni kiritish va natijada balans ko‘rsatkichini aniqlash;

3. Qidiruv yo‘li bo‘ylab "orqaga chekinish" va har bir uchda balans ko‘rsatkichini tekshirish. Agar kerak bo‘lsa, muvozanatni saqlash.

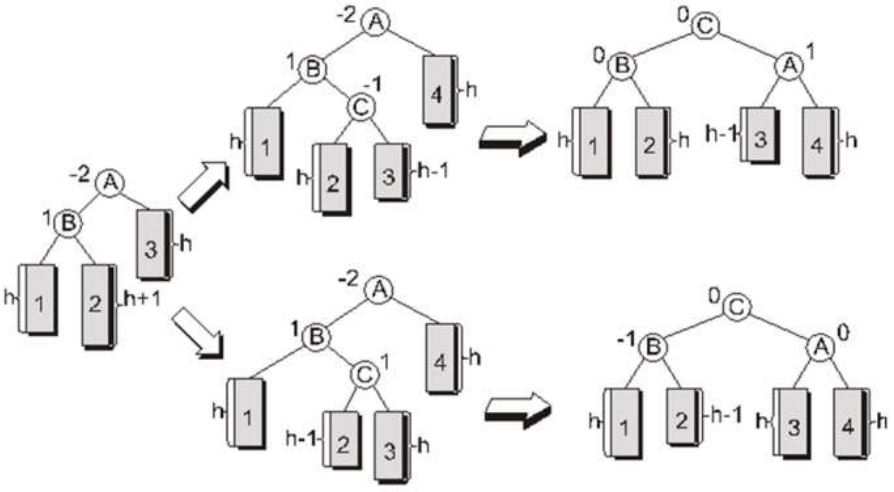
Amalda AVL balansini tiklashning 4 algoritmi qo‘llaniladi: muvozanat ko‘rsatkichlari qiymatiga qarab tanlangan kichik va katta chap burilish, kichik va katta o‘ng burilish (37-40-rasmlar)



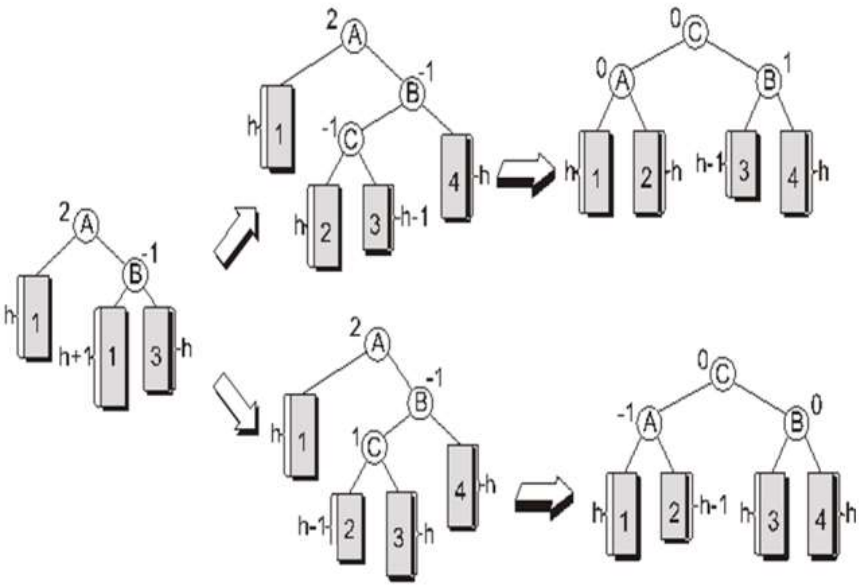
37-rasm. Kichik chap burilish algoritmi



38-rasm. Kichik o‘ng burilish algoritmi

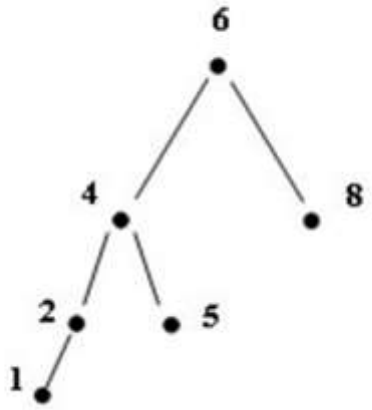


39-rasm. Katta chap burilish algoritmi



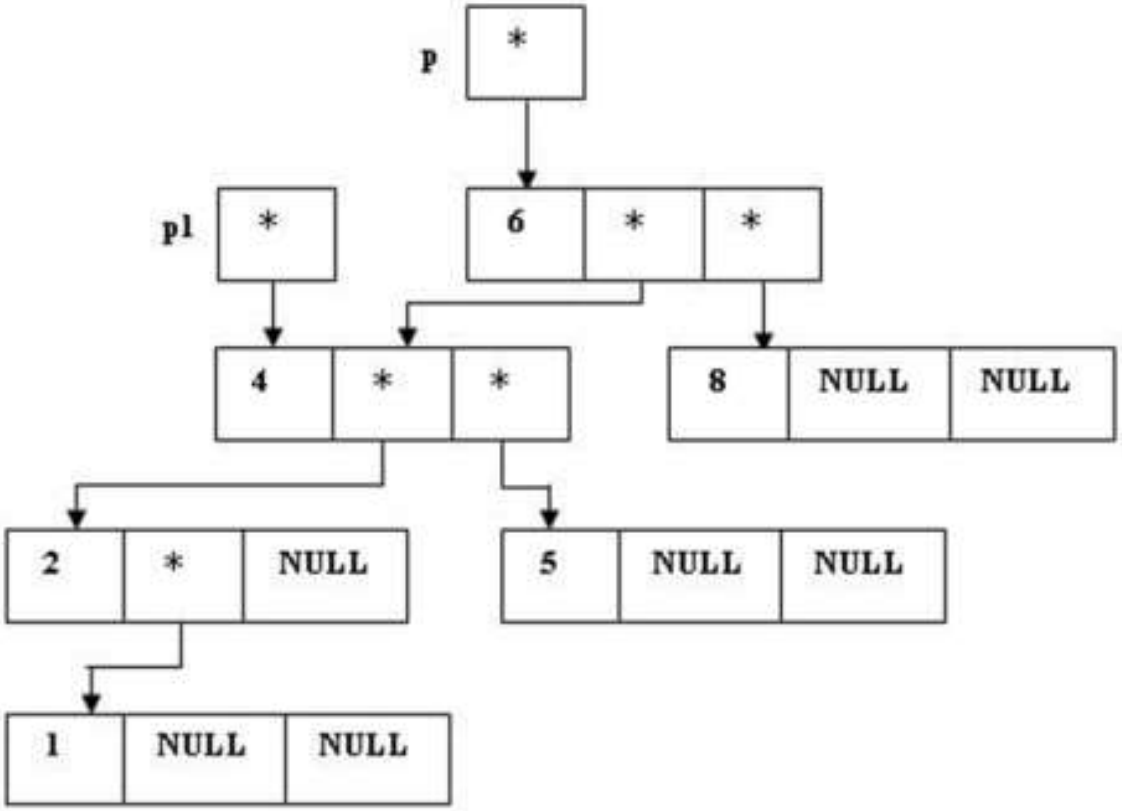
40-rasm. Katta o‘ng burilish algoritmi

Rasmlarda to‘rtburchaklar pastki daraxtlarni bildiradi, ichidagi raqamlar kichik daraxtlarning raqamlari, tugunlar yonidagi raqamlar balans ko‘rsatkichlari. Balanslash algoritmi chap tomonga burilishning quyidagi misolida keltirilgan.



41-rasm. Daraxtning dastlabki berilishi

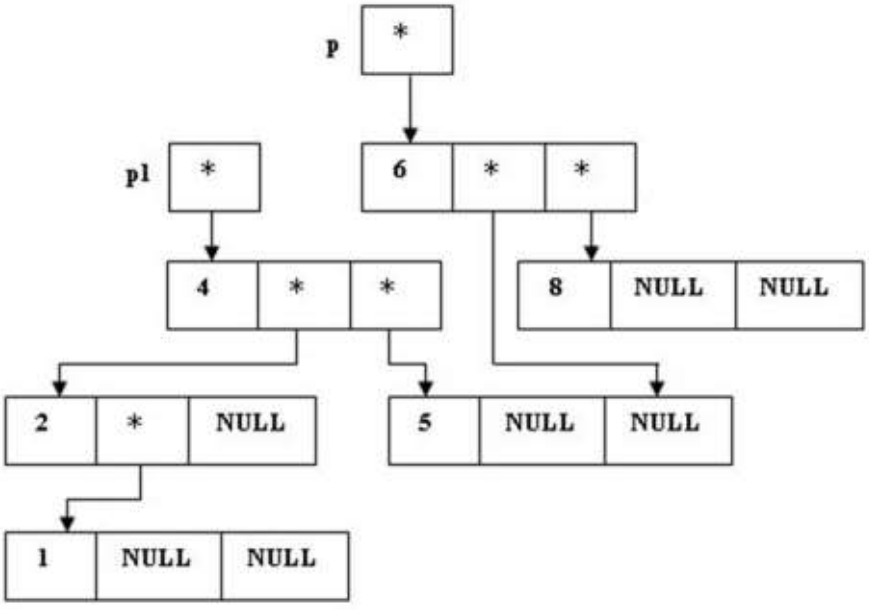
1. Daraxtning ildiziga aylanadigan uchining manzilini aniqlash:
2. P1=(*p).Left;



42-rasm. Yangi daraxt ildizining manzilini saqlash

3. "Yangi" ildizdan o‘ng pastki daraxtni qayta ulang, ushbu daraxtni "eski" ildizning chap pastki daraxtiga aylantiring:

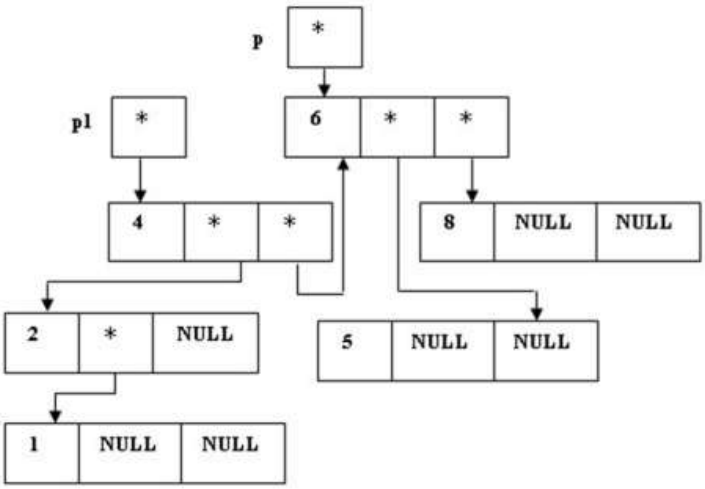
4. (*p).Left = (*p1).Right;



43-rasm. Qayta biriktirish

5. "Yangi" ildizning o‘ng pastki daraxtini "eski" ildizdan boshlanganligini aniqlash:

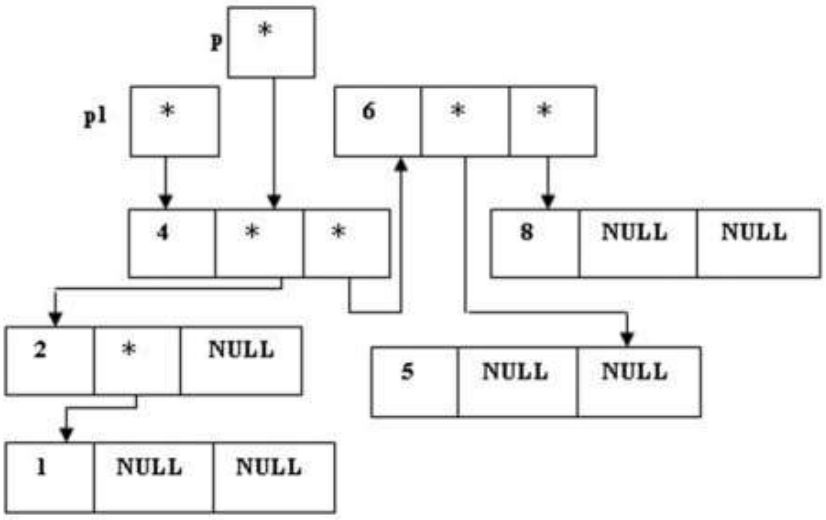
6. (*p1).Right = p;



44-rasm. “Yangi” ildizning o‘ng pastki daraxtini aniqlash

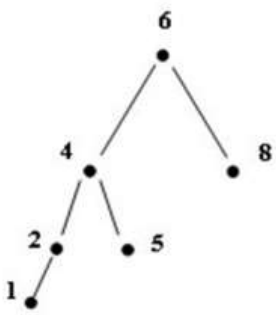
7. Ko‘rsatkichning qiymatini daraxtning ildiziga o‘zgartiring (p) va balans qiymatini tiklang:

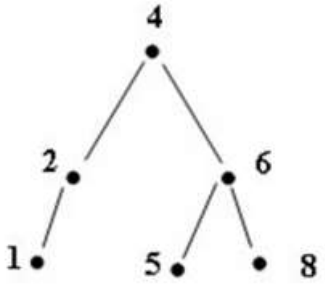
8. (*p).bal=0; p=p1;



45-rasm. "Yangi" ildizning o‘ng pastki daraxtini aniqlash

Muvonazatlash algortmidan so‘ng, AVL bo‘yicha muvozanatlashgan quyidagi daraxt hosil bo‘ldi:





a) Dastlabki daraxt **b) Muvozanatlashgan daraxt**

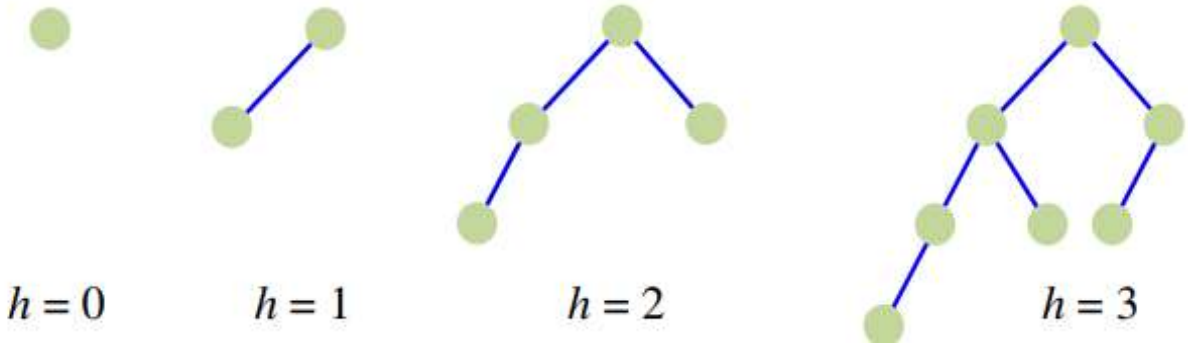
9.2. AVL daraxtlarining samaradorligini tahlil qilish

N elementni o‘z ichiga olgan AVL daraxtining balandligini yuqoridan baholaylik.

h balandlikdagi AVL daraxtini hosil qilish uchun zarur bo‘lgan minimal tugunlarni N (h) bilan belgilaymiz.

N(-1)=0, N(0)=1, N(1)=2, N(2)=4, N(3)=7, …

0,1, 2, 4, 7, 12, 20, 33, 54, …



$$N(h) = N(h-1) + N(h-2) + 1$$

- $N(h)$: 1, 2, 4, 7, 12, 20, 33, 54, …
- Fibonacci(h): 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

$$\boldsymbol{N(h) = F(h+3) - 1}, \text{ для } h \geq 0$$

Fibonachchi ketma-ketligining h –hadi uchun Binet formulasidan

$$F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}} = \frac{\varphi^n - (-\varphi)^{-n}}{\varphi - (-\varphi)^{-1}} = \frac{\varphi^n - (-\varphi)^{-n}}{2\varphi - 1}$$

$\varphi = \frac{1+\sqrt{5}}{2}$ – Oltin nisbat

$$F_n \sim \frac{\varphi^n}{\sqrt{5}}$$

AVL daraxtining h (n) balandligi uchun yuqori chegara:

$$\log(n+1) \le h(n) \le 1.4404 \cdot \log(n+2) - 0.328$$

AVL daraxtidan tugunlarni olib tashlash

```
#include <iostream>
using namespace std;
struct avltree
{

   int key;
   char *value;
   int height;
   struct avltree *left;
   struct avltree *rigth;
};
```

AVL daraxtidan barcha tugunlarni olib tashlash funksiyasi

```
void avltree_free (struct avltree *tree)
{
   if (tree == NULL)
      return;
   avltree_free(tree->left);
   avltree_free(tree->rigth);
   free(tree);
}
```

Tugundan kalit bo‘yicha izlash funksiyasi

```
struct avltree *avltree_lookup(struct avltree *tree, int key)
{
   while(tree !=NULL)
   {
      if(key == tree->key)
      {
         return tree;
      }
      else
      if(key<tree->key)
      {
         tree = tree->left;
      }
      else
      {
        tree = tree->rigth;
      }
   }
};
```

Tugun hosil qilish funksiyasi

```
struct avltree *avltree_create(int key,char *value)
{
   struct avltree *node;
   node = malloc(sizeof(*node));
   if (node != NULL)
   {
      node->key = key;
      node->value = value;
      node->left = NULL;
      node->right = NULL;
      node->height = 0;
   }
   return node;
}
```

Daraxtni ekranda chiqarish funksiyasi

```
void avltree_print_dfs(struct avltree *tree, int level)
{
   int i;
   if (tree == NULL)
      return;
   for (i = 0; i < level; i++)
      printf(" ");
   printf("%d\n", tree->key);
   avltree_print_dfs(tree->left, level + 1);
   avltree_print_dfs(tree->right, level + 1);
}
int main()
{
   struct avltree *tree = NULL;
   tree = avltree_add(tree, 5,    "5");
   tree = avltree_add(tree, 3,    "3");
   /* Code */
   avltree_free(tree);
   return 0;
}
```

Mavzu yuzasidan savollar:

1. AVL daraxti nima?
2. Uchlarni muvozanatlash deganda nimani tushunasiz.
3. Tugundan kalit bo‘yicha izlash funksiyasini tushuntirib bering.
4. Muvozanatlashgan daraxt tushunchasi nima?
5. Daraxt ma’lumotlar strukturasi qo‘llaniladigan sohalarga qaysilar kiradi?

Mustaqil ishlash uchun masalalar:

1. AVL daraxtida tugun olib tashlash funksiyasini yozing va uni daraxtda qo‘llang.
2. Kalit bo‘yicha izlash funksiyasini optimallashtiring.