Курс «Алгоритмы и алгоритмические языки»

Лекция 18

Деревья Фибоначчи

- ***Определение*** дерева Фибоначчи
  (это тоже искусственное дерево).

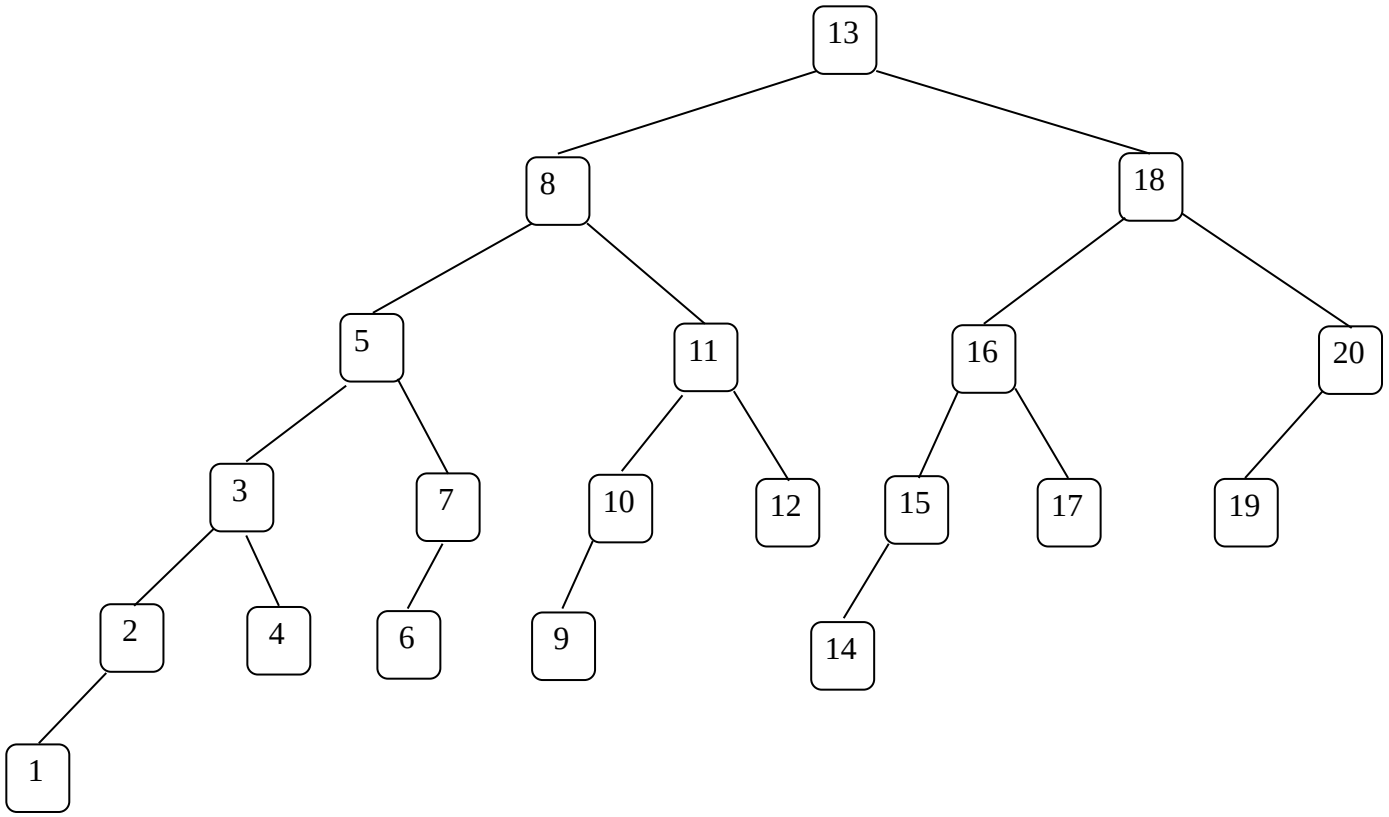
  (1) Пустое дерево – это дерево Фибоначчи с высотой $h = 0$.

  (2) Двоичное дерево, левое и правое поддерево которого есть деревья Фибоначчи с высотами соответственно $h-1$ и $h-2$ (либо $h-2$ и $h-1$), есть дерево Фибоначчи с высотой h.

Из определения следует, что в дереве Фибоначчи значения высот левого и правого поддерева отличаются ровно на 1.

Деревья Фибоначчи

- ***Пример***. Дерево Фибоначчи с $h = 6$.



Деревья Фибоначчи

- ***Теорема 1***. Число вершин в дереве Фибоначчи F_h высоты h равно $m(h) = f_{h+2} - 1$.

Доказательство (по индукции).

$h = 0$: $m(0) = f_2 - 1 = 0$

$m(1) = f_3 - 1 = 1$.

Шаг: по определению $m(h) = m(h-1) + m(h-2) + 1$.

Имеем $m(h) = (f_{h+1} - 1) + (f_h - 1) + 1 = f_{h+2} - 1$,

так как $f_h + f_{h+1} = f_{h+2}$

Деревья Фибоначчи

◈ ***Теорема 2***. Пусть C_1 и C_2 таковы, что уравнение

$$r^2 - C_1 r - C_2 = 0 \qquad (*)$$

имеет два различных корня r_1 и r_2, $r_1 \neq r_2$.

Тогда для $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$

выполняется соотношение $a_n = C_1 a_{n-1} + C_2 a_{n-2}$.

Доказательство. r_1 и r_2 – корни уравнения (*),

то $r_1^2 = C_1 r_1 + C_2$

$r_2^2 = C_1 r_2 + C_2$.

Имеем:

$$C_1 a_{n-1} + C_2 a_{n-2} = C_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1}) + C_2(\alpha_1 r_1^{n-2} + \alpha_2 r_2^{n-2}) =$$
$$= \alpha_1 r_1^{n-2}(C_1 r_1 + C_2) + \alpha_2 r_2^{n-2}(C_1 r_2 + C_2) =$$
$$= \alpha_1 r_1^{n-2} r_1^2 + \alpha_2 r_2^{n-2} r_2^2 = \alpha_1 r_1^n + \alpha_2 r_2^n = a_n \qquad (**)$$

Теорема доказана.

Деревья Фибоначчи

◈ ***Теорема 3***. Пусть C_1 и C_2 таковы, что уравнение

$$r^2 - C_1 r - C_2 = 0 \qquad (*)$$

имеет два корня r_1 и r_2, $r_1 \neq r_2$.
Тогда

из $a_n = C_1 a_{n-1} + C_2 a_{n-2}$ и начальных условий a_0 и a_1

следует $a_n = \alpha_1 r_1^{\,n} + \alpha_2 r_2^{\,n}$ для n = 1, 2, ...

Доказательство. Нужно не только повторить в обратном порядке вывод (**), но и подобрать такие α_1 и α_2, чтобы

$$a_0 = \alpha_1 + \alpha_2,\ a_1 = \alpha_1 r_1 + \alpha_2 r_2 \qquad (***)$$

Рассматривая (***) как систему линейных уравнений относительно α_1 и α_2, получим:

$$\alpha_1 = \frac{a_1 - a_0 \cdot r_2}{r_1 - r_2}, \qquad \alpha_2 = \frac{-a_1 + a_0 \cdot r_1}{r_1 - r_2}$$

Теорема доказана.

Деревья Фибоначчи

- Применим доказанные теоремы к числам Фибоначчи:

$f_n = f_{n-1} + f_{n-2}$.

Уравнение $r^2 - r - 1 = 0$ имеет корни

$$r_1 = \frac{1+\sqrt{5}}{2} \qquad r_2 = \frac{1-\sqrt{5}}{2}$$

Следовательно, согласно теореме 3

$$f_n = \alpha_1 \cdot r_1^n + \alpha_2 \cdot r_2^n = \alpha_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n + \alpha_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n,$$

$$f_0 = \alpha_1 + \alpha_2 = 0,$$

$$f_1 = \alpha_1 \cdot \left(\frac{1+\sqrt{5}}{2}\right) + \alpha_2 \cdot \left(\frac{1-\sqrt{5}}{2}\right) = 1$$

$$\alpha_1 = \frac{1}{\sqrt{5}}, \alpha_2 = -\frac{1}{\sqrt{5}}$$

Деревья Фибоначчи

Откуда

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

Согласно теореме 1

$$m(h) = f_{h+2} - 1 = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{h+2} - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^{h+2} - 1$$

$$\left| \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^{h+2} \right| < 1$$

.

$$m(h) + 1 > \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^{h+2}$$

Деревья Фибоначчи

Обозначение $\gamma = \frac{1+\sqrt{5}}{2}$

$$m(h)+1 > \frac{1}{\sqrt{5}}\gamma^{h+2} \qquad (****)$$

Логарифмируя обе части (****) , получаем

$$h+2 < \frac{\log_2(m+1)}{\log_2 \gamma} + \frac{\log_2 \sqrt{5}}{\log_2 \gamma}$$

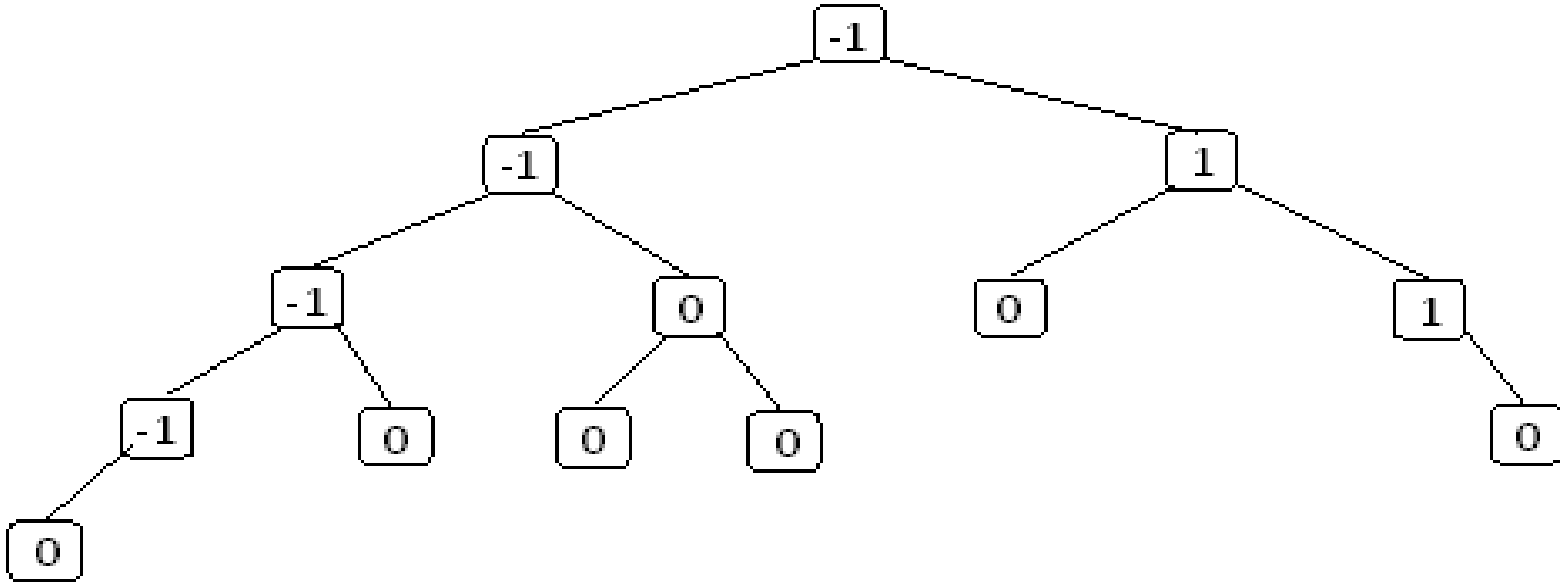
откуда $$h < 1{,}44 \cdot \log_2(m+1) - 0{,}32$$

Таким образом, мы доказали, что для деревьев Фибоначчи с числом вершин m количество сравнений в худшем случае не превышает

$1{,}44 \cdot \log_2(m+1) - 0{,}32$

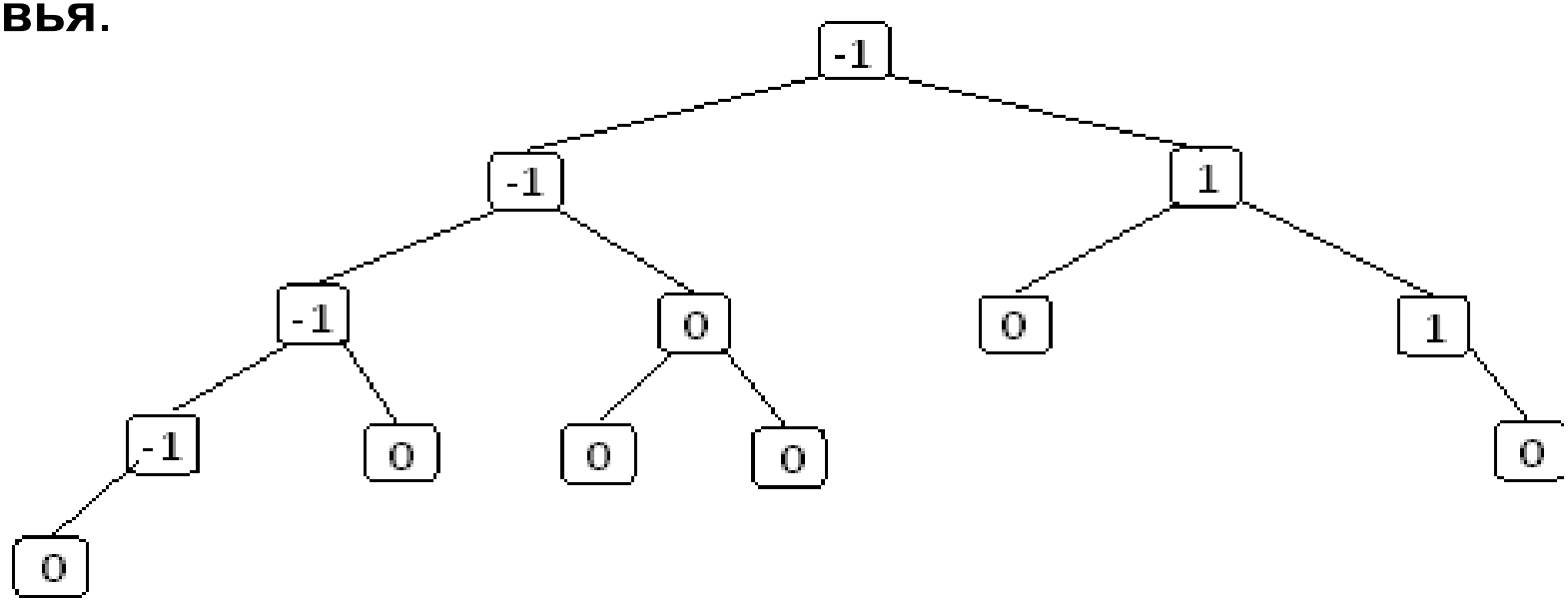
АВЛ -деревья.

- В АВЛ-деревьях (Адельсон-Вельский, Ландис) оценка сложности не лучше, чем в совершенном дереве, но не хуже, чем в деревьях Фибоначчи для всех операций: поиск, исключение, занесение.
- *АВЛ-деревом* (подравненным деревом) называется такое двоичное дерево, когда для любой его вершины высоты левого и правого поддерева отличаются не более, чем на 1.



Пример АВЛ-дерева.

АВЛ -деревья.



- В узлах дерева записаны значения *показателя сбалансированности* (*balance Factor*), определяемого по формуле:

  $$balance\ Factor = height(right\ subtree) - height(left\ subtree)$$

  Показатель сбалансированности может иметь одно из трех значений

  –1: Высота левого поддерева на 1 больше высоты правого поддерева.

  0: Высоты обоих поддеревьев одинаковы.

  +1: Высота правого поддерева на 1 больше высоты левого поддерева.

- У совершенного дерева *все* узлы имеют показатель баланса 0 (это самое «хорошее» АВЛ-дерево) а у дерева Фибоначчи *все* узлы имеют показатель баланса +1 (либо –1) (это самое «плохое» АВЛ-дерево).

АВЛ -деревья.

- Типичная структура узла АВЛ-дерева:

```
typedef int KeyType;
struct AvlNode;
typedef struct AvlNode *Position;
typedef struct AvlNode *AvlTree;
struct AvlNode {
  KeyType   Key;          //ключ
  AvlTree   Left;         //левое поддерево
  AvlTree   Right;        //правое поддерево
  int       Balance;      //показатель баланса
  int       Hight;        //высота поддерева
};
```

АВЛ -деревья.

- Базовые операции над АВЛ-деревьями.

```
AvlTree MakeEmpty( AvlTree T);          //удалить дерево
Position Find (KeyType X, AvlTree T);   //поиск по ключу
Position FindMin (AvlTree T);           //минимальный ключ
Position FindMax (AvlTree T);           //максимальный ключ
AvlTree Insert (KeyType X, AvlTree T); //вставить узел
AvlTree Delete (KeyType X, AvlTree T); //исключить узел
```

Реализация простейших базовых операций.

- Удалить дерево:

```
AvlTree MakeEmpty (AvlTree T) {
  if (T != NULL) {
    MakeEmpty (T->Left);
    MakeEmpty (T->Right);
    free (T);
  }
  return NULL;
}
```

- Поиск по ключу:

```
Position Search (KeyType X, AvlTree T) {
 if (T == NULL)
   return NULL;
 if (X < T->Element)
   return Find(X, T->Left);
 else if(X > T->Element)
   return Find(X, T->Right);
 else
   return T;
}
```

Реализация простейших базовых операций.

- Минимальный и максимальный ключи:

```
Position FindMin (AvlTree T) {
  if (T == NULL)
    return NULL;
  else if (T->Left == NULL)
    return T;
  else
    return FindMin (T->Left);
}

Position FindMax (AvlTree T) {
  if (T != NULL)
    while (T->Right != NULL)
      T = T->Right;
  return T;
}
```

Включение узла в АВЛ-дерево.

- ***Поддержка балансировки АВЛ-дерева при выполнении операции включения ключей.***

  Рассматриваемое дерево состоит из корневой вершины r и левого (L) и правого (R) поддеревьев, имеющих высоты h_L и h_R соответственно.

  Для определенности будем считать, что новый ключ включается в поддерево L.

- h_L **не изменяется** $\Rightarrow$ не изменяются соотношения между h_L и h_R $\Rightarrow$ свойства АВЛ-дерева сохраняются.

- h_L **увеличивается на 1** $\Rightarrow$ возможны три случая:

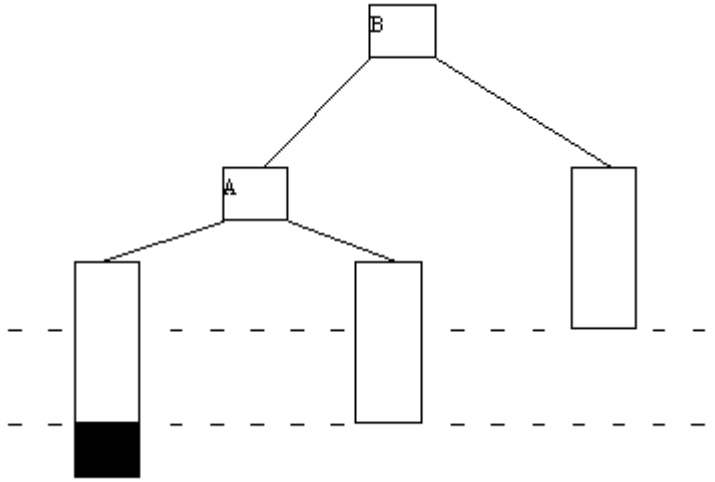
  (1) $h_L = h_R \Rightarrow$ после добавления вершины L и R станут разной высоты, но свойство сбалансированности сохранится

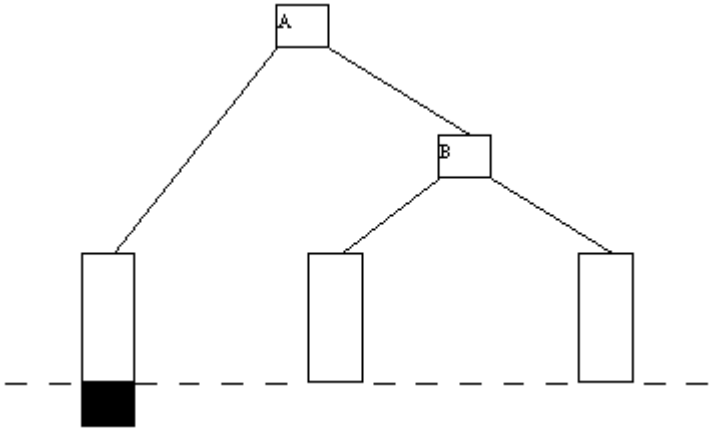
  (2) $h_L < h_R \Rightarrow$ после добавления новой вершины L и R станут равной высоты, т.е. сбалансированность общего дерева даже улучшится

  (3) $h_L > h_R \Rightarrow$ после включения ключа сбалансированность нарушится, и *потребуется перестройка дерева*.

Включение узла в АВЛ-дерево.

- (3а) **Новая вершина добавляется к левому поддереву поддерева** L. В результате поддерево с корнем в узле B разбалансировалось: разность высот его левого и правого поддеревьев стала равной –2 .



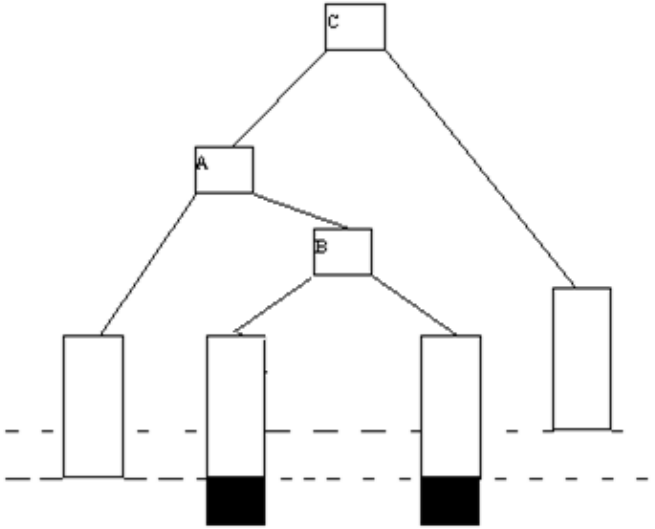


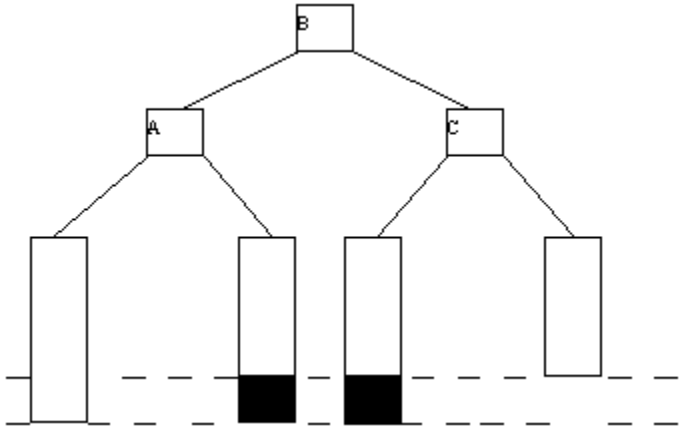
Преобразование, разрешающее ситуацию (3а) (однократный поворот $\boldsymbol{RR}$):

Делаем узел A корневым узлом поддерева, в результате правое поддерева с корнам в узле B «опускается» и разность высот становится равной –1

Включение узла в АВЛ-дерево.

- (3b) **Новая вершина добавляется к левому поддереву поддерева** L. В результате поддерево с корнем в C разбалансировалось: разность высот его левого и правого поддеревьев стала равной - 2.





Преобразование, разрешающее ситуацию (3b) (двукратный поворот LR):

«Вытягиваем» узел B на самый верх, чтобы его поддеревья поднялись. Для этого сначала делаем левый поворот, меняя местами поддеревья с корневыми узлами A и B, а потом – правый поворот, меняя местами поддеревья с корневыми узлами B и C.