

Лекция 2. Медианы, порядковые статистики и сбалансированные деревья

Дано множество из n различных чисел

Найти элемент множества, который будет i -м по счету, если расположить числа по возрастанию.

Медианой называют элемент, находящийся «посередине», если упорядочить элементы. Точнее при нечетном n , $i = (n + 1)/2$; при четном n их две $i = n/2$, $i = n/2 + 1$, но медианой будем считать меньшую их них.

Алгоритмы:

- 1) упорядочить и взять i -й элемент, $O(n \cdot \log n)$;
- 2) i шагов пирамидального алгоритма, $O(n + i \cdot \log n)$.

Выбор за линейное в среднем время

Randomized-Select (A, p, r, i)

1. **if** $p = r$
2. **then return** $A[p]$
3. $q \leftarrow$ Randomized-Partition (A, p, r)
4. $k \leftarrow q - p - 1$
5. **if** $i \leq k$
6. **then return** Randomized-Select (A, p, q, i)
 else return Randomized-Select ($A, q + 1, r, i - k$)

На шаге 3 массив $A[p, r]$ разбивается на два:

$A[p, q]$, $A[q + 1, r]$, причем всякий элемент из $A[p, q]$ не больше любого элемента из $A[q + 1, r]$.

Время в худшем случае — $O(n^2)$.

Randomized – Partition (A, p, r)

1. $i \leftarrow \text{Random}(p, r)$
2. поменять $A[p] \leftrightarrow A[i]$
3. **return** Partition (A, p, r)

Выбрали случайный элемент и поставили его первым.

Partition (A, p, r)

1. $x \leftarrow A(p)$
2. $i \leftarrow p - 1$
3. $j \leftarrow r + 1$
4. **while** TRUE
5. **do repeat** $j \leftarrow j - 1$
6. **until** $A[j] \leq x$
7. **repeat** $i \leftarrow i + 1$
8. **until** $A[i] \geq x$
9. **if** $i < j$
10. **then** поменять $A[i] \leftrightarrow A[j]$
11. **else return** j

На выходе: $p \leq j < r$ и массивы $A[p, j], A[j + 1, r]$ всегда непустые.

Анализ разбиений

Рангом элемента $x = A[p]$ назовем число элементов массива, непревосходящих x .

Поскольку все элементы имеют равные шансы попасть на место $A[p]$, то все значения ранга от 1 до n равновероятны (т.е. их вероятность равна $1/n$).

- 1) Если $rank(x) > 1$, то массив $A[p, q]$ будет содержать $(rank(x) - 1)$ элементов, т.е. все элементы, меньше x .
- 2) Если $rank(x) = 1$, то $q = p$ и $A[p, q]$ содержит только один элемент.

Итак: левая часть будет содержать 2, 3, ... или $n - 1$ элементов с вероятностью $1/n$, и один элемент с вероятностью $2/n$.

Все числа разные.

Анализ алгоритма Randomized – Select

Пусть $T(n)$ — математическое ожидание времени работы алгоритма (среднее по всем массивам длины n). В худшем случае i -я статистика всегда будет оказываться в большей доле при разбиении $A[p, q]$ и $A[q + 1, r]$.

Тогда

$$\begin{aligned} T(n) &\leq \frac{1}{n} \left(T(\max(1, n-1)) + \sum_{k=1}^{n-1} T(\max(k, n-k)) \right) + O(n) \leq \\ &\leq \frac{1}{n} \left(T(n-1) + 2 \sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k) \right) + O(n) = \frac{2}{n} \sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k) + O(n). \end{aligned}$$

Так как $T(n-1)$ не более $O(n^2)$, то $T(n-1)/n$ можно включить в $O(n)$.

Покажем, что $T(n) < cn$ для некоторой подходящей константы $c > 0$.

Индукция по n

Пусть $T(j) \leq cj$ для $j < n$.

Тогда

$$T(n) \leq \frac{2}{n} \sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} ck + O(n) \leq \frac{2c}{n} \frac{n}{2} \frac{\frac{n}{2} + 1 + n - 1}{2} + O(n) = \frac{3cn}{4} + O(n) \leq cn.$$

Итак: любая порядковая статистика и, в частности, медиана могут быть найдены за линейное в среднем время.

Чем плох этот алгоритм?

Выбор за линейное в худшем случае время

Алгоритм $\text{Select}(i)$

1. Разбить n элементов на пятерки, $\lfloor n/5 \rfloor$ групп.
2. В каждой пятерке найти медиану.
3. Вызвать рекурсивно Select для медиан и найти X — медиану медиан.
4. Разбить массив на два подмассива $A[p, q]$ и $A[q + 1, r]$ относительно элемента X . Пусть k — число элементов, не превосходящих X , т.е. длина первого массива.
5. Вызвать рекурсивно $\text{Select}(i)$ для $A[p, q]$, если $i \leq k$ или $\text{Select}(i - k)$, если $i > k$.

Анализ алгоритма $\text{Select}(i)$

Сколько чисел будет больше X ?

Не меньше половины медиан дадут по 3 числа, т.е. половина из $\lceil n/5 \rceil$ групп за двумя возможными исключениями: группа содержащая X и последняя, возможно, неполная группа, т.е.

$$3 \left(\left\lceil \frac{1}{2} \lceil \frac{n}{5} \rceil \right\rceil - 2 \right) \geq \frac{3n}{10} - 6$$

столько заведомо больше X и столько же заведомо меньше X .

Значит, на шаге 5 алгоритм Select будет обрабатывать массив длиной не более $\frac{7n}{10} + 6$.

Пусть $T(n)$ — время работы алгоритма в худшем случае. Тогда

$$T(n) \leq T\left(\left\lceil n/5 \right\rceil\right) + T\left(\left\lfloor 7n/10 + 6 \right\rfloor\right) + O(n).$$

Индукцией по n покажем, что $T(n) \leq cn$ для некоторой константы c .

Пусть что $T(m) \leq cm$ при $m < n$. Тогда

$$\begin{aligned} T(n) &\leq c\left(\left\lceil n/5 \right\rceil\right) + c\left(\left\lfloor 7n/10 + 6 \right\rfloor\right) + O(n) \\ &\leq c\left(n/5 + 1\right) + c\left(7n/10 + 6\right) + O(n) \\ &\leq 9cn/10 + 7c + O(n) = cn - c\left(n/10 + 7\right) + O(n). \end{aligned}$$

При $n > 70$ получаем $O(n)$, выбрав c не меньше, чем в $O(n)$.

При $n \leq 70$ возможно дополнительное увеличение константы c .

Почему пятерки? А тройки? Семерки?

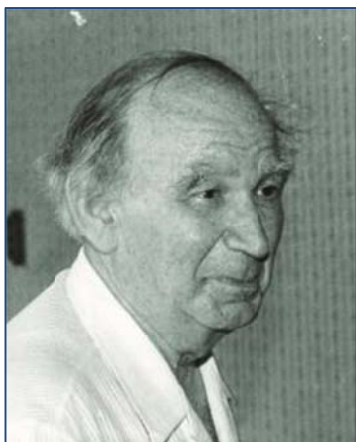
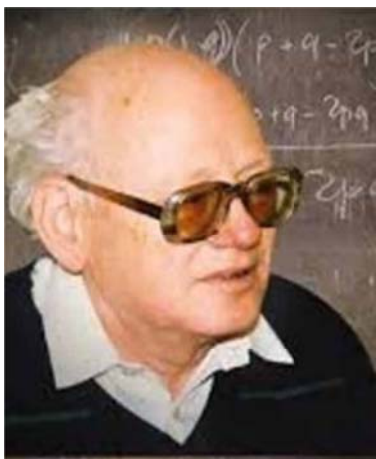
Сбалансированные деревья

Для массива данных требуется

1. Найти элемент;
2. Найти k -й по порядку элемент;
3. Вставить элемент;
4. Удалить элемент.

Сбалансированные деревья требуют $O(\log n)$ для этих операций.

Впервые такие деревья были предложены московскими математиками Г.М. Адельсоном-Вельским и Е.М. Ландисом в 1962 году



[История открытия AVL-деревьев](#)

Определение 1. **Высотой** дерева называется максимальная длина пути от корня до листа.

Определение 2. Бинарное дерево называется **сбалансированным** (или **AVL-деревом**), если для любой его вершины высота правого поддеревя отличается от высоты левого поддеревя не более чем на единицу.

Теорема. Длина ветвей в n -вершинном сбалансированном дереве заключена между $\log_2 n$ и $\frac{3}{2}\log_2 n$.

Доказательство.

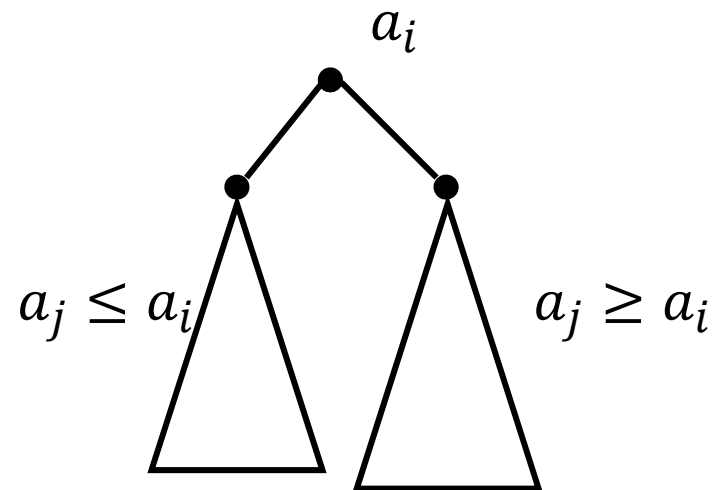
1. Бинарное дерево высоты h не может содержать больше 2^{h+1} вершины, то есть $n \leq 2^{h+1}$ или $h + 1 \geq \log_2 n$.
2. Наиболее ассиметричное AVL-дерево T_h высоты h имеет наиболее ассиметричное AVL-дерево T_{h-1} высоты $h - 1$ в качестве одного из своих поддеревьев и наиболее ассиметричное AVL-дерево T_{h-2} в качестве другого. Обозначим через $n(h)$ число вершин в дереве T_h . Тогда

$$n(h) = n(h - 1) + n(h - 2) + 1; \quad n(0) = 1, \quad n(-1) = 0.$$

Для $h = 3, 4$ можно непосредственно проверить, а затем по индукции доказать, что $n(h) \alpha^{h+1} >$, где $\alpha = \frac{1+\sqrt{5}}{2}$

Следовательно, $n \geq n(h) > \alpha^{h+1}$, откуда $h + 1 \leq \log n / \log \alpha \approx 1,44 \log n$.

Пусть в вершинах AVL–дерева расположены элементы массива так, что для любой вершины в левом ее поддереве расположены элементы не больше чем в данной вершине, а в правом поддереве — не меньше, чем в этой вершине.



Пример. Поиск в AVL–дереве потребует более 25 сравнений, только если дерево состоит из не менее 196 417 вершин.

Случайные деревья

Для сбалансированного дерева длина пути из корня в лист не превышает $1,44 \log n$.

Для случайного дерева **средняя** длина пути из корня в лист составляет $1,39 \log n$, но в худшем случае может оказаться равной n .

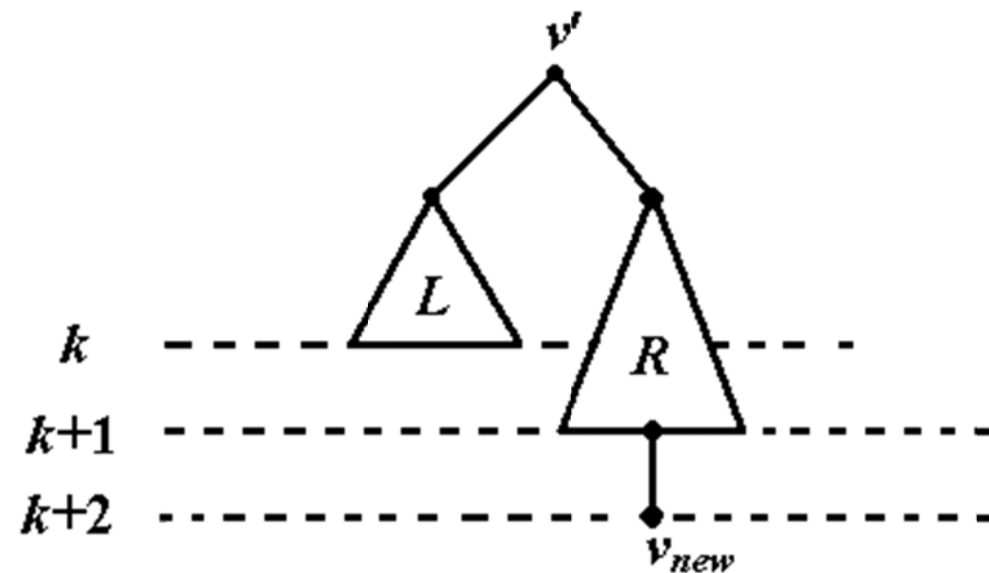
Для сбалансированного дерева **средняя** длина пути составляет $c \log n$,
$$c = \frac{\alpha}{\sqrt{5} \log \alpha} \approx 1.04$$

Включение новой вершины в AVL–дерево

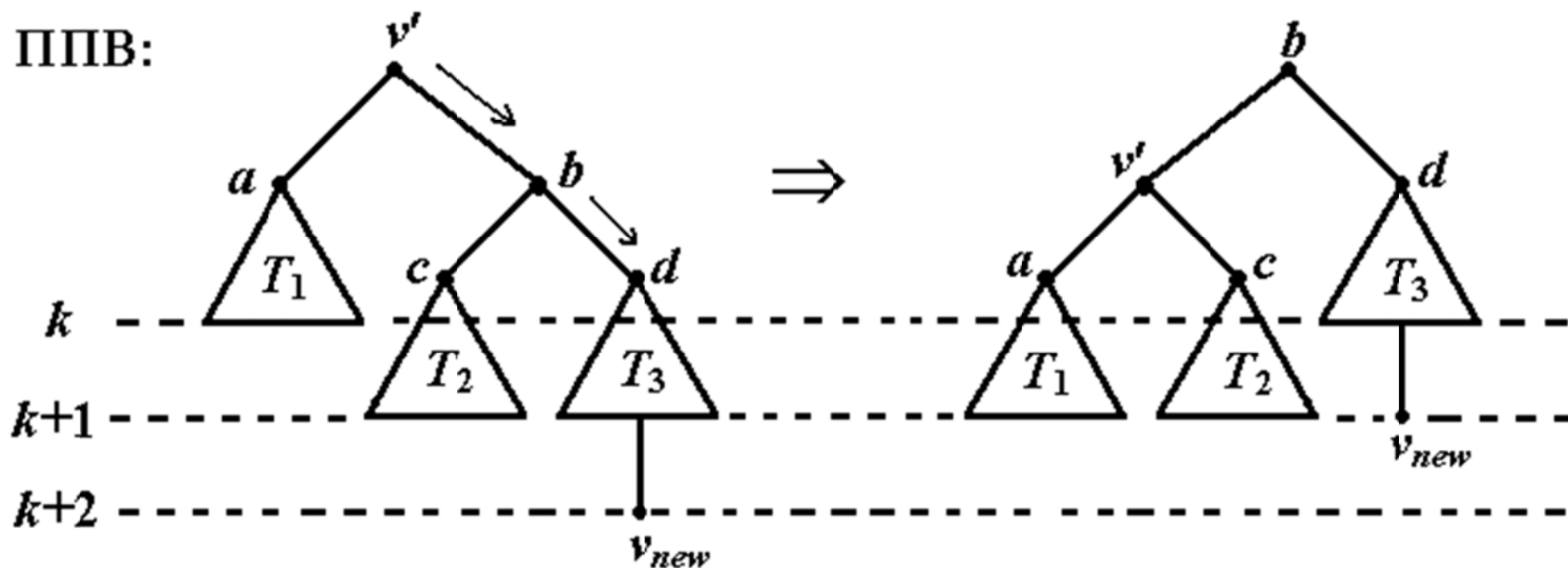
При добавлении новой вершины v_{new} к AVL–дереву мы «скатываем» ее от корня вдоль веток и получаем новый лист (висячую вершину). Дерево остается бинарным, но баланс может нарушиться. Эти нарушения могут возникнуть только у вершин, лежащих на пути от корня к новой вершине.

Будем последовательно подниматься от новой вершины к корню и восстанавливать баланс, если это необходимо.

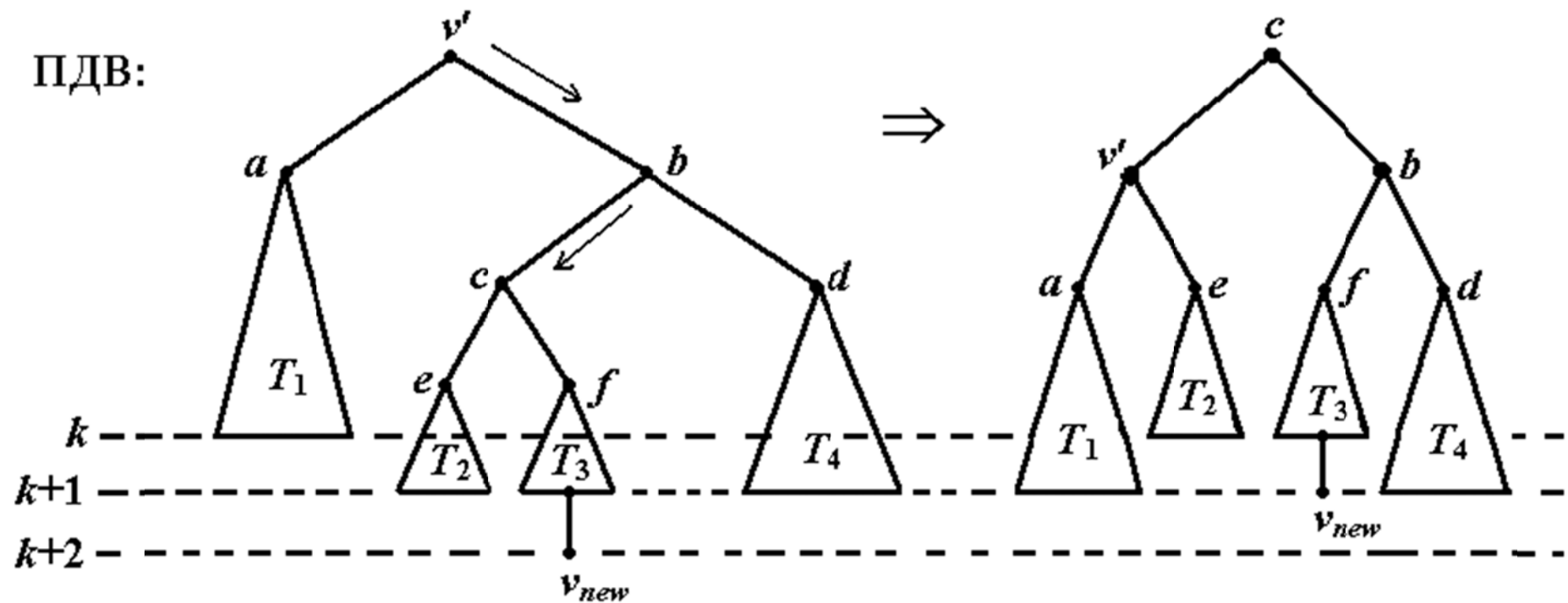
Пусть v' — самая нижняя вершина дисбаланса, то есть наиболее удаленная от корня вершина такая, что ее поддерево с вершиной v_{new} имеет высоту $k + 2$, а другое поддерево — высоту k :



Будем восстанавливать баланс в v' следующим образом. Если первые два шага на (единственном пути) от v' к v_{new} делаются в одном направлении (оба вправо, или оба влево), то применяем правило простого вращения (ППВ).

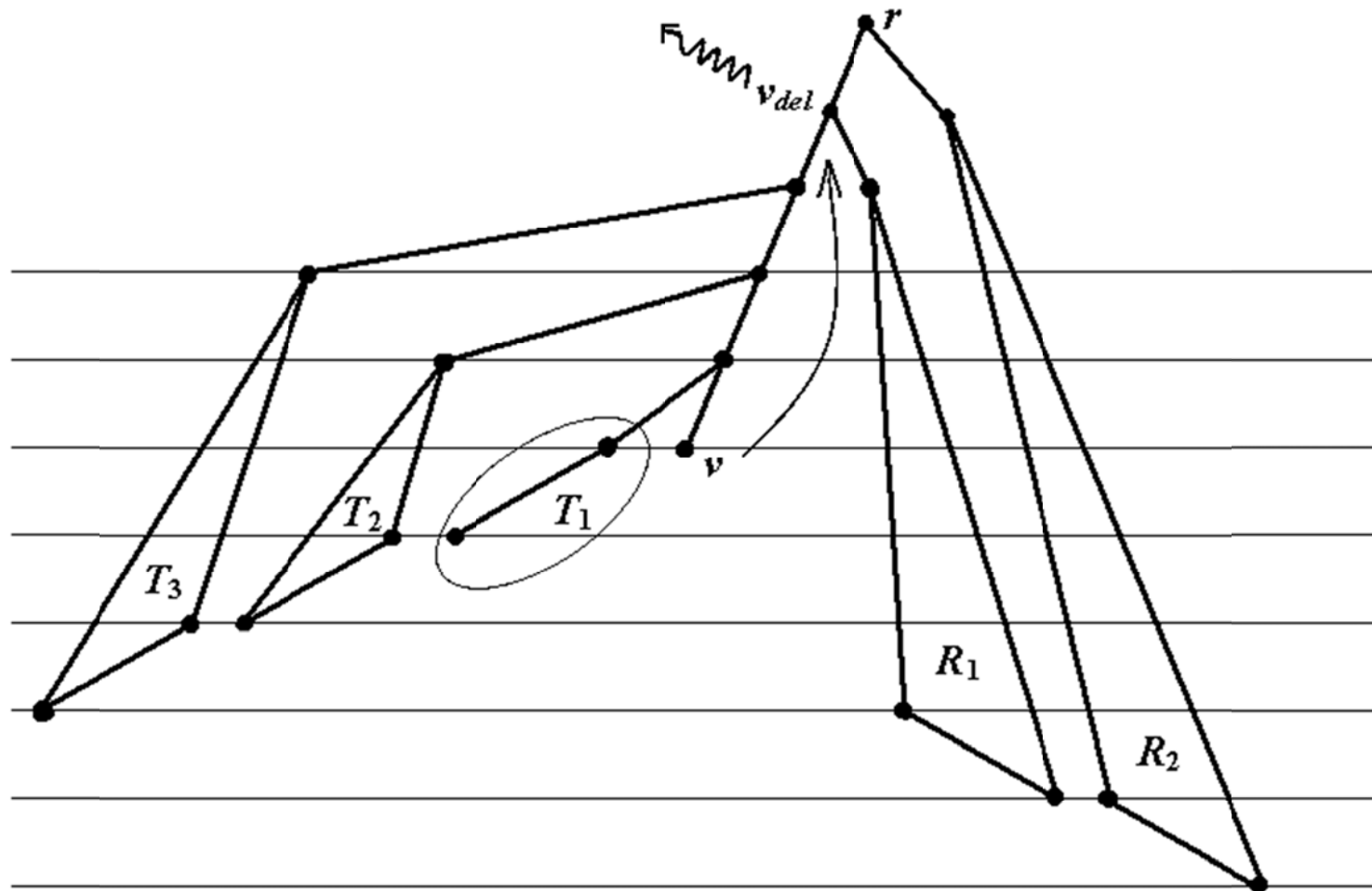


Если первые два шага делаются в разных направлениях, то применяем правило двойного вращения (ПДВ):

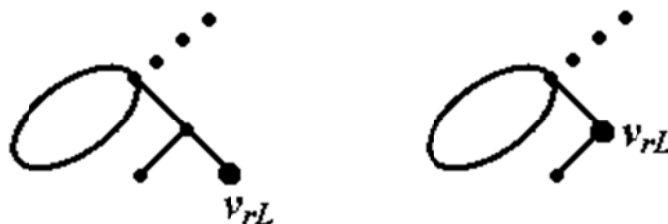


Удаление вершины

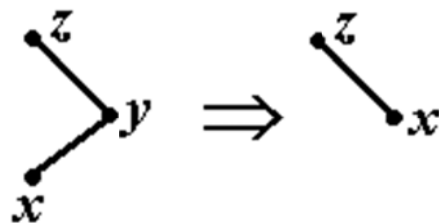
На место удаленной вершины v ставим либо самую правую вершину v_{rL} левого поддерева, либо самую левую вершину v_{Lr} правого поддерева



Нюанс. Каждая из вершин v_{rL} и v_{Lr} может быть либо висячей, либо предвисячей, то есть имеющей в качестве потомков лишь одну вершину (разумеется, висячую):



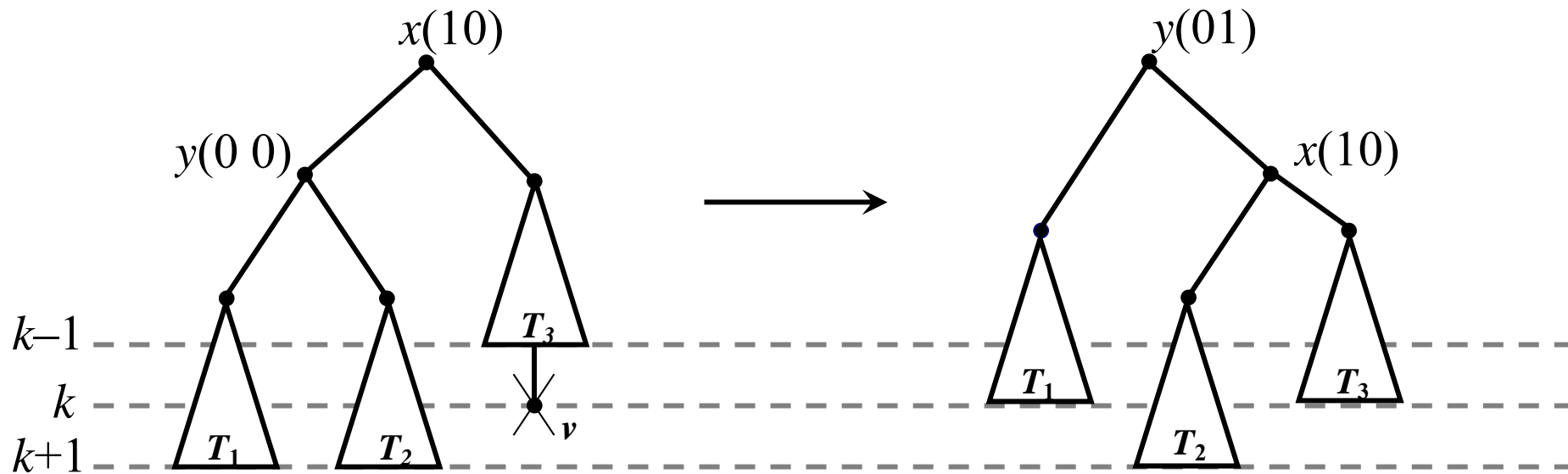
Если на место удаленной вершины встает предвисячая вершина y , то ее (вершины y) потомок x подключается к ее предку z :



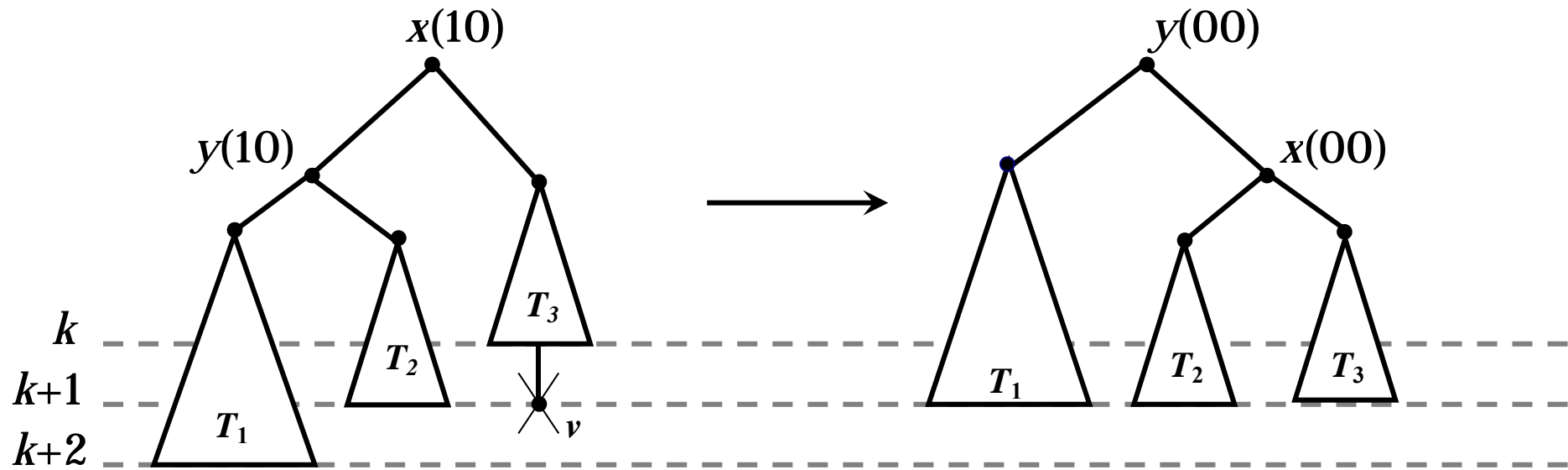
Итак, можно считать, что **всегда удаляем лист**. Последовательно поднимаемся от удаленной вершины v к корню и исправляем структуру дерева, если необходимо.

Пусть к текущей вершине x на пути от v к корню мы пришли справа по короткой ветке. Тогда возможно три случая:

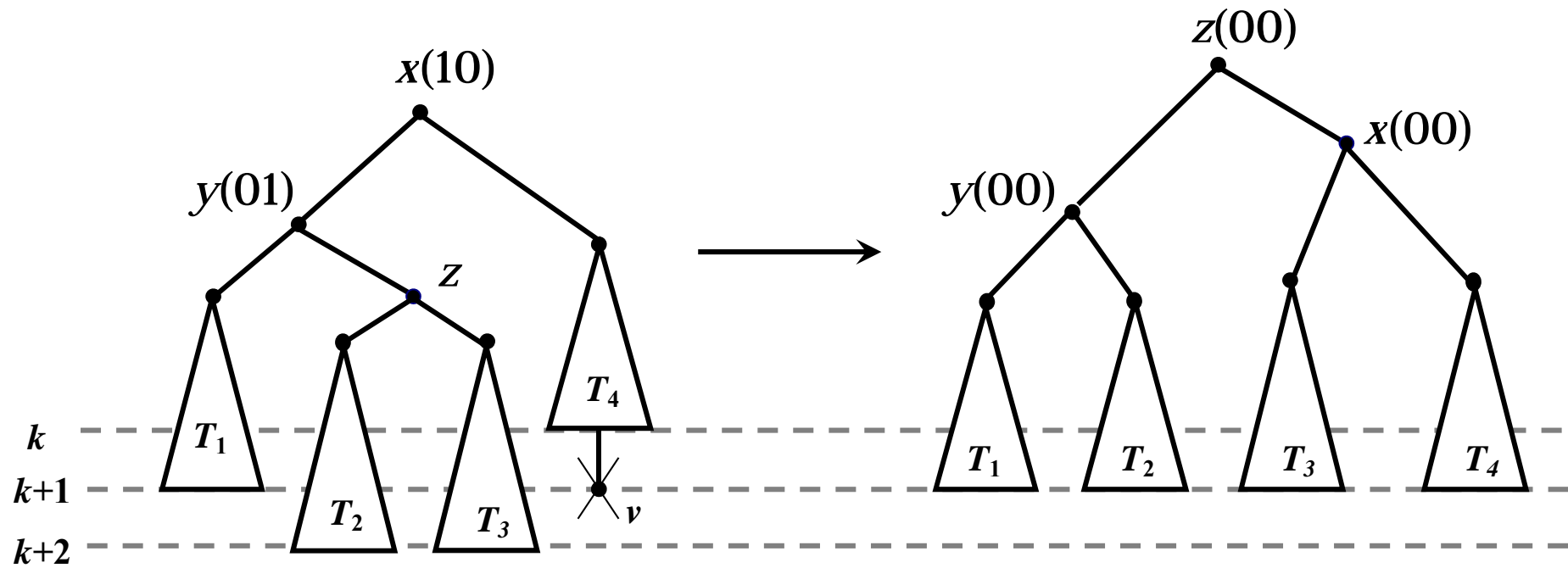
а) В вершине y высоты поддеревьев равны:



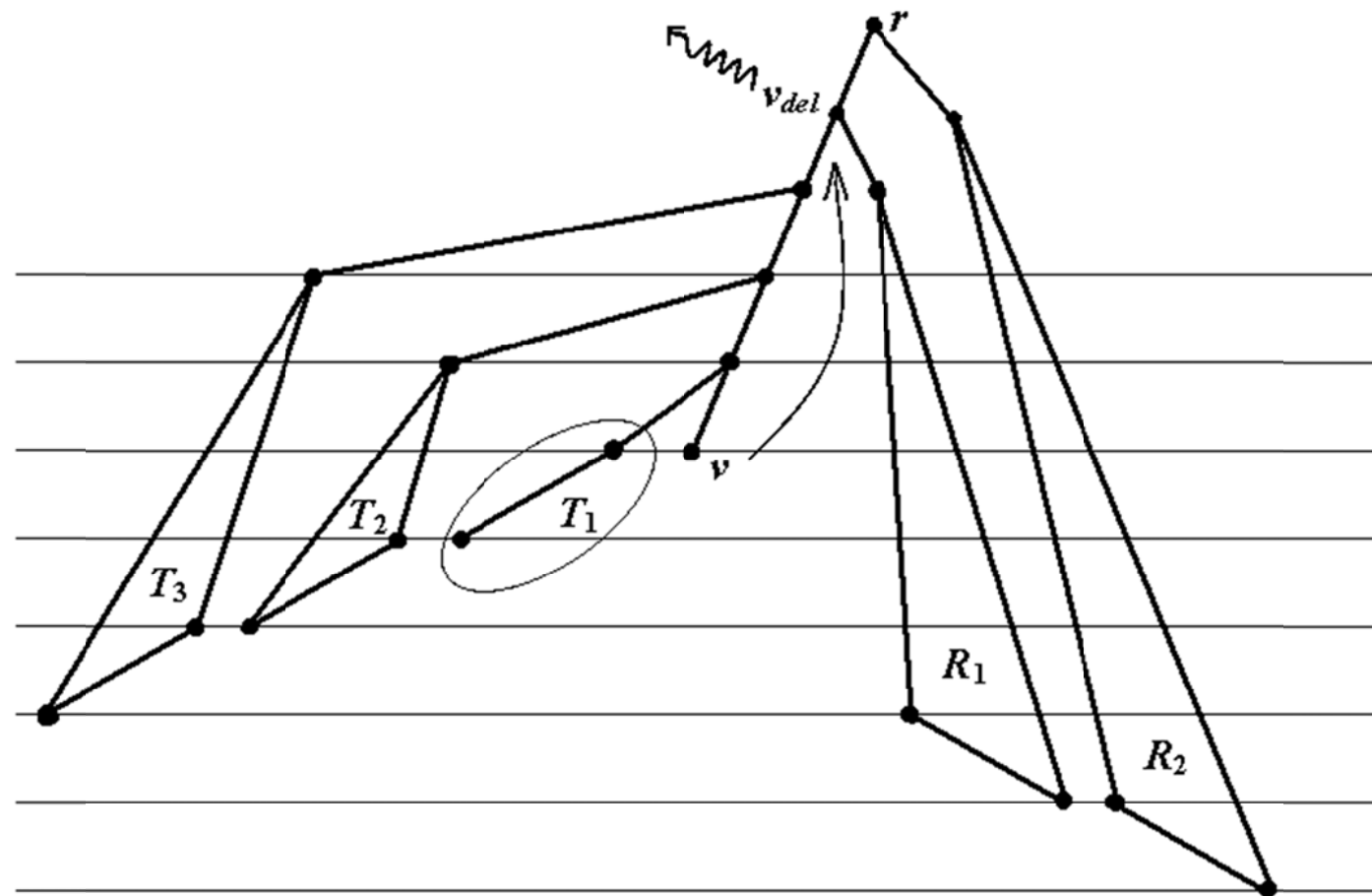
б) В вершине y высота левого поддерева больше высоты правого поддерева:



в) В вершине u высота левого поддерева меньше высоты правого поддерева

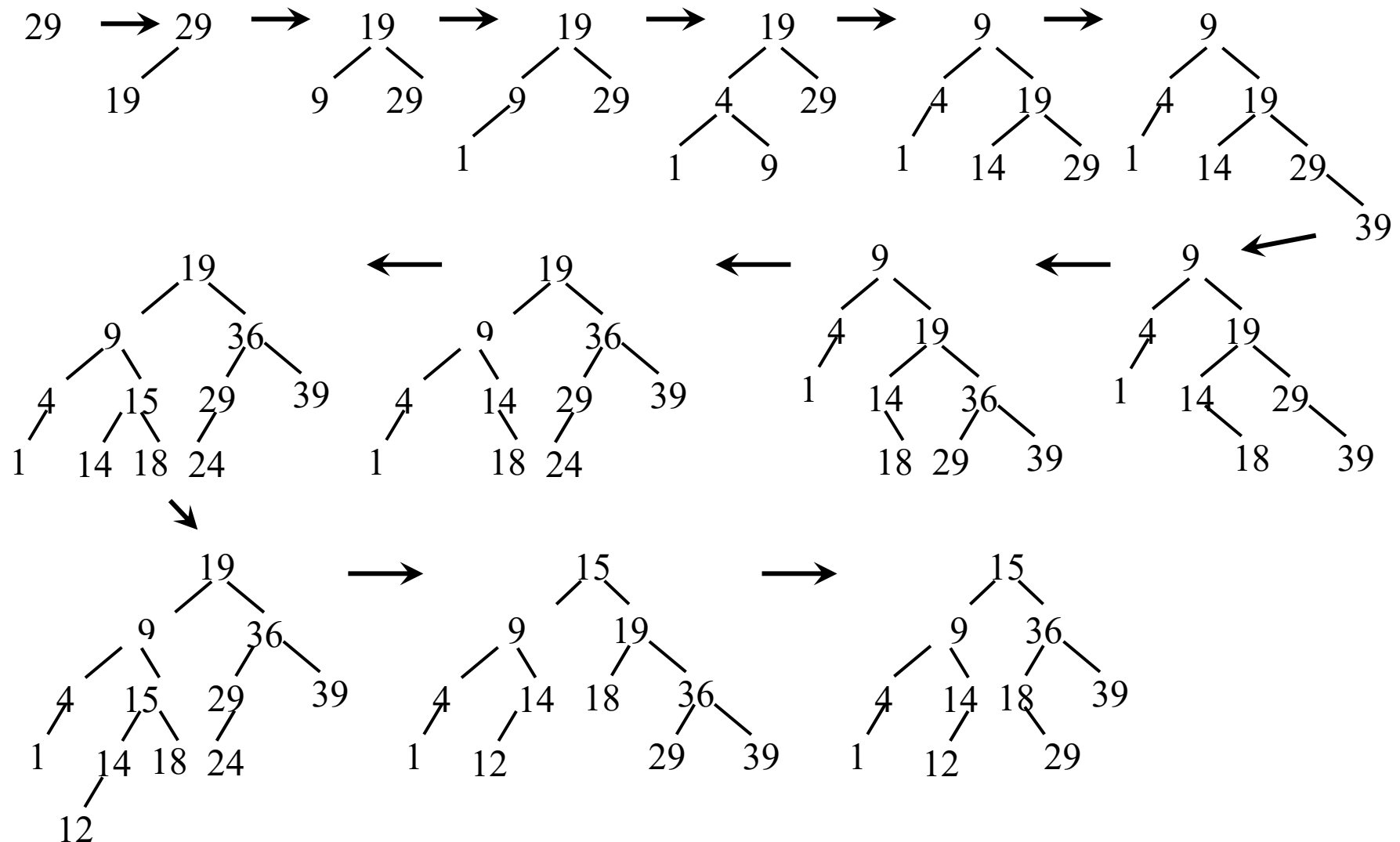


Отметим, что устранение дисбаланса в одной из вершин, может нарушить баланс в вышестоящих вершинах. На рисунке показан предельный случай этого явления. При начальном дисбалансе лишь в одной вершине (лежащей непосредственно под v) приходится, тем не менее, производить перестройку дерева во всех вершинах на пути к корню.



Пример. Построить последовательность AVL-деревьев для данных, поступающих в следующем порядке: 29, 19, 9, 1, 4, 14, 39, 18, 36, 24, 15, 12.

Удалить из полученного дерева 24, а затем 19.



Упражнение. Построить последовательность AVL–деревьев для данных, поступающих в следующем порядке: 15, 7, 17, 5, 4, 3, 56, 23, 22, 6, 10, 25, 26. Удалить из полученного дерева 6 и 10, а затем 15 и 25.