Пирамида (Куча)

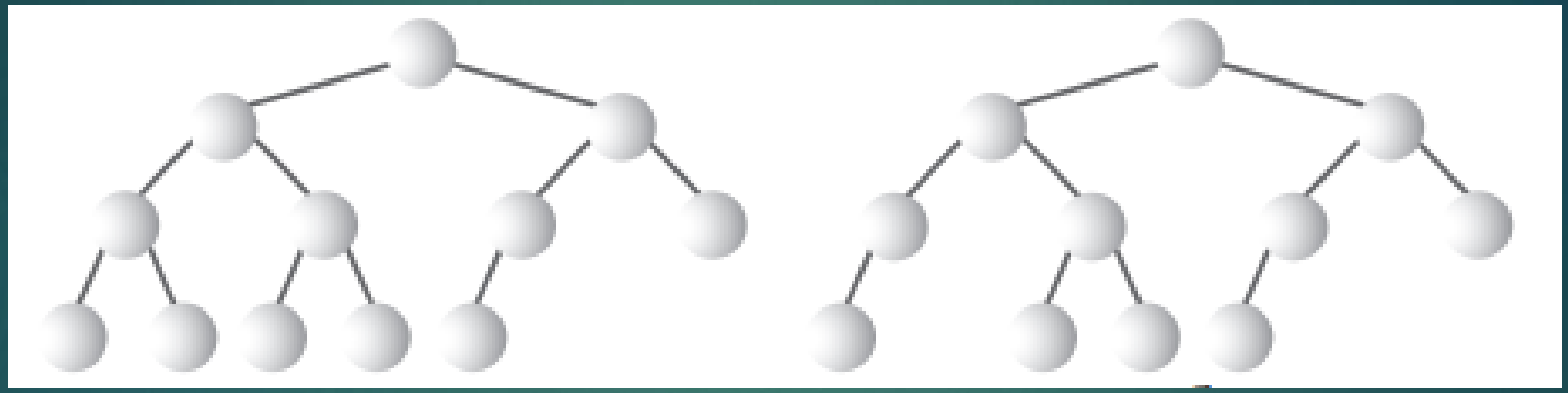
ЛЕКЦИЯ 10

Что есть пирамида?

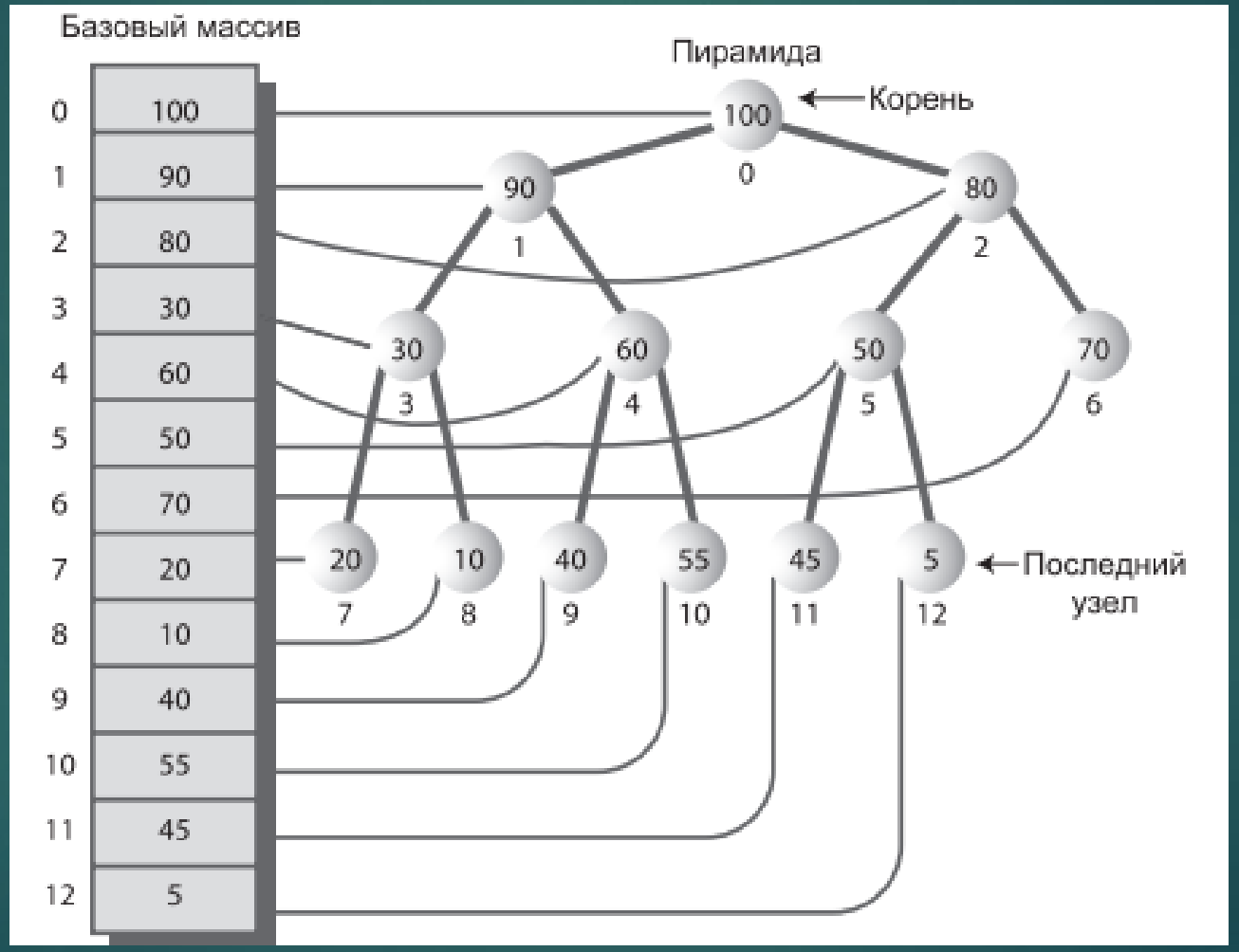
Пирамидой называется двоичное дерево, обладающее следующими характеристиками:

- Полнота. Все уровни дерева содержат все возможные узлы (хотя последний уровень может быть заполнен лишь частично).
- Пирамида (обычно) реализуется на базе массива.
- Для каждого узла в пирамиде выполняется основное условие, гласящее, что ключ каждого узла больше (либо равен) ключей его потомков.

Что есть пирамида?



Что есть пирамида?



Поводы использования пирамиды

По сравнению с деревом двоичного поиска, в котором ключ левого потомка каждого узла меньше ключа правого потомка, пирамида является слабо упорядоченной (квазиупорядоченной). В частности, ограничения дерева двоичного поиска позволяют выполнить перебор узлов в порядке сортировки по простому алгоритму.

В пирамиде упорядоченный перебор узлов затрудняется тем, что принцип организации пирамиды (условие пирамиды) не так силен, как принцип организации дерева. Единственное, что можно гарантировать по поводу пирамиды — то, что на каждом пути от корня к листу узлы упорядочены по убыванию. Как видно из рис. 12.2, ключи узлов, находящихся слева или справа от заданного узла, на более высоких или низких уровнях (и не принадлежащих тому же пути), могут быть больше или меньше ключа узла. Пути, не имеющие общих узлов, полностью независимы друг от друга.

Поводы использования пирамиды

Вследствие слабой упорядоченности пирамиды некоторые операции с ней затруднены или невозможны. Из-за отсутствия нормальной возможности перебора пирамида не предоставляет удобных средств поиска заданного ключа. Дело в том, что алгоритм поиска не располагает достаточной информацией для принятия решения о том, какого из двух потомков узла следует выбрать для перехода на нижней уровень. Соответственно узел с заданным ключом невозможно удалить (по крайней мере за время O(logN)), потому что его нельзя найти. (Операции можно выполнить последовательным просмотром всех ячеек последовательности, но это может быть сделано только за время O(N).)

Таким образом, структура пирамиды выглядит довольно хаотично. Тем не менее упорядоченности пирамиды достаточно для быстрого удаления наибольшего узла и быстрой вставки новых узлов. Этих операций достаточно для использования пирамиды в качестве приоритетной очереди

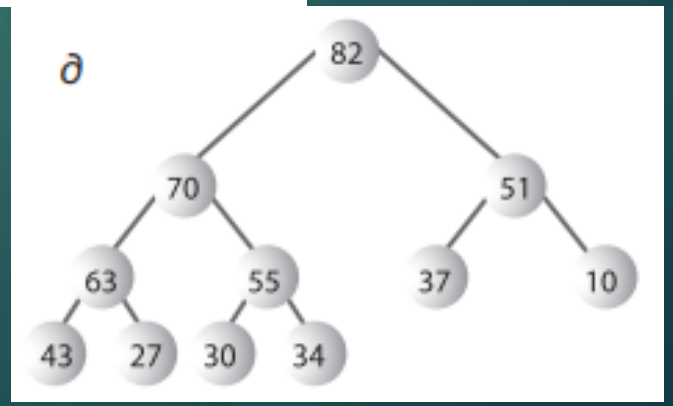
Удаление элемента

Под удалением подразумевается удаление с наибольшим ключом. Этот узел всегда является корневым, поэтому его удаление выполняется просто. Корень всегда хранится в ячейке 0 базового массива.

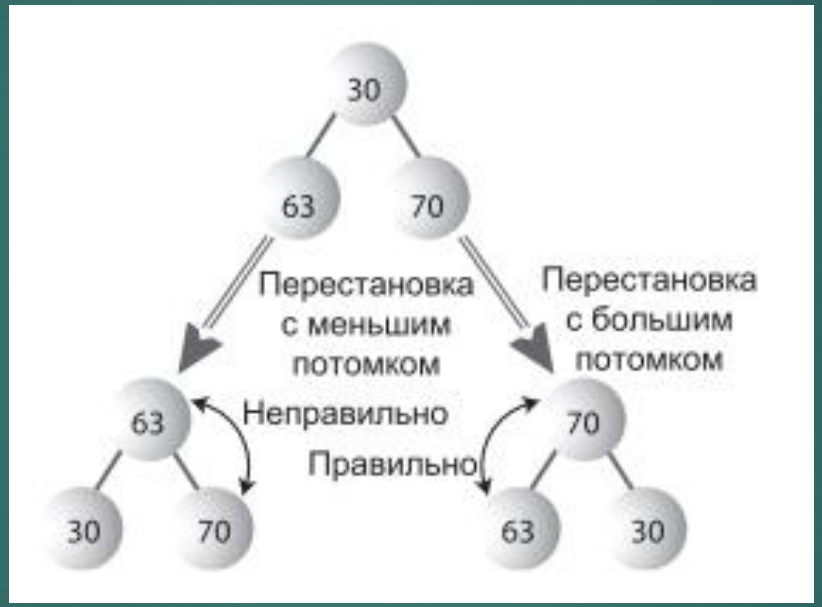
Проблема в том, что после удаления корня дерево перестает быть полным; в нем появляется пустая ячейка. «Дыру» необходимо заполнить. Все элементы массива можно было бы сдвинуть на одну ячейку, но существует другое, более быстрое решение. Последовательность действий при удалении наибольшего узла выглядит так:

1. Удалить корневой узел.
2. Переместить последний узел на место корневого.
3. Смещать его вниз до тех пор, пока он не окажется ниже большего и выше меньшего узла.

Удаление элемента



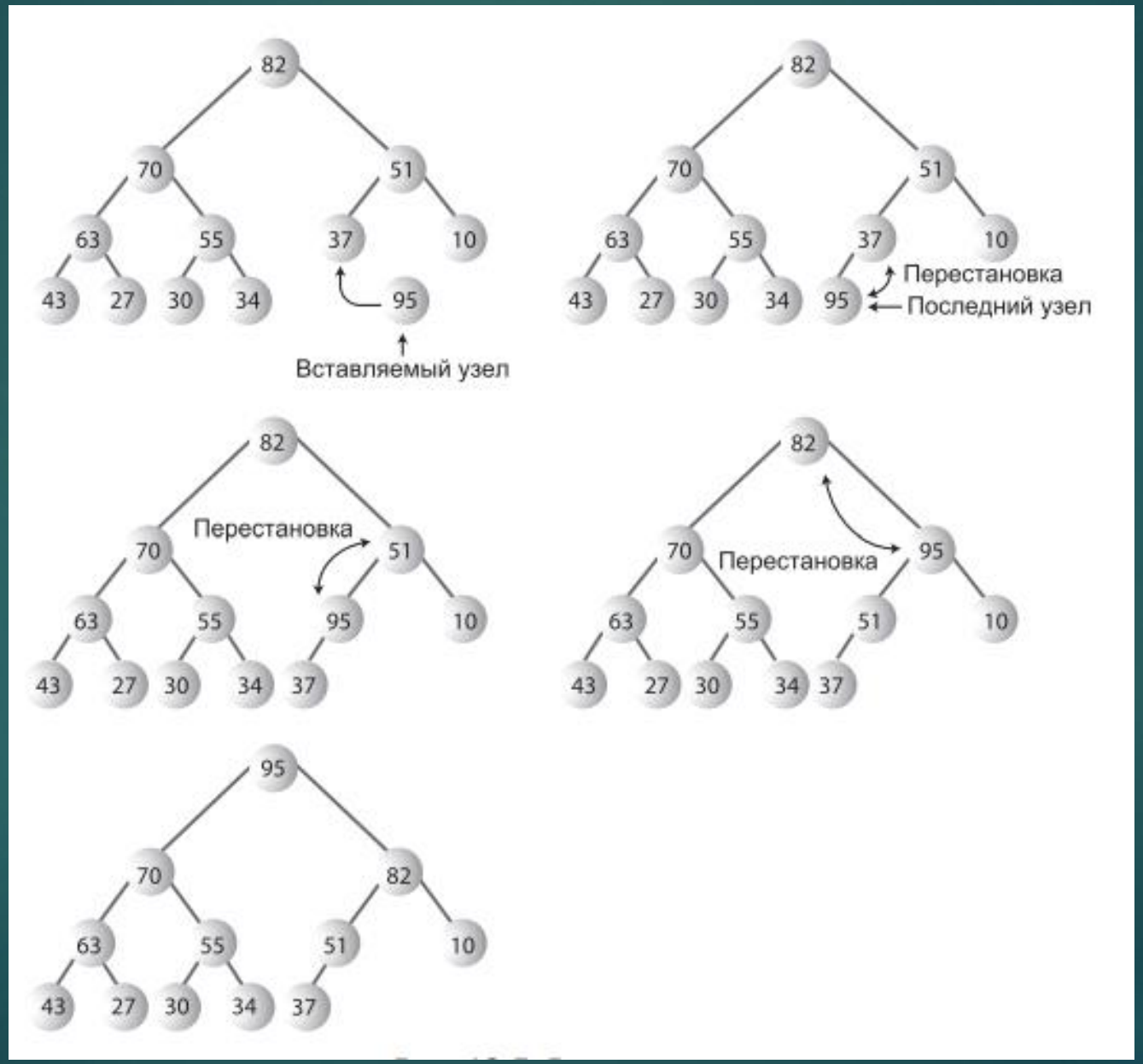
Удаление элемента – Выбор потомка



Вставка элемента

Вставка узла тоже выполняется относительно просто. При вставке используется смещение вверх (вместо смещения вниз). Сначала вставляемый узел помещается в первую свободную позицию в конце массива, в результате чего размер массива увеличивается на единицу

Вставка элемента



Размер массива

Размер массива (количество узлов в пирамиде) — важнейший атрибут состояния пирамиды и необходимое поле класса Heap. Узлы, скопированные из последней ячейки, не стираются, поэтому алгоритм может определить положение последней занятой ячейки только по текущему размеру массива

Эффективность

Для пирамиды со сколько-нибудь значительным количеством элементов алгоритмы смещения вверх и вниз оказываются наиболее затратными составляющими всех представленных операций. Эти алгоритмы выполняются в цикле: узлы шаг за шагом смещаются вверх и вниз по пути. Количество необходимых операций копирования ограничивается высотой пирамиды: если пирамида состоит из пяти уровней, то четыре операции копирования переместят «дыру» от верхнего уровня до нижнего. (Не будем учитывать два перемещения последнего узла во временное хранилище и обратно; они необходимы всегда и поэтому выполняются за постоянное время.)

Пирамидальная сортировка

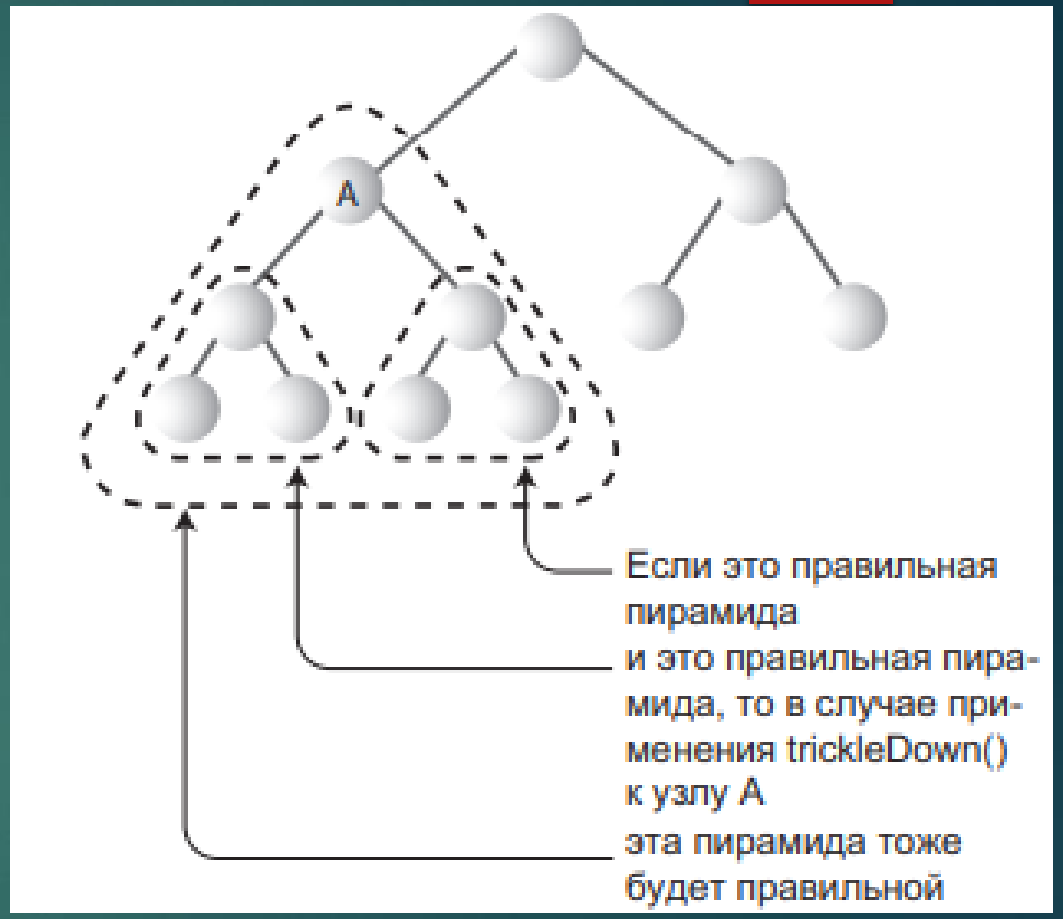
Эффективность пирамиды как структуры данных является одной из предпосылок на удивление простого и эффективного алгоритма сортировки, называемого пирамидальной сортировкой. Алгоритм строится на вставке всех неупорядоченных элементов в пирамиду обычным методом insert(). Затем многократные вызовы remove() извлекают элементы в порядке сортировки. Реализация может выглядеть примерно так:

```
for(j=0; j<size; j++)
 theHeap.insert( anArray[j] ); // Из несортированного массива
for(j=0; j<size; j++)
 anArray[j] = theHeap.remove(); // В отсортированный массив
```

Так как методы insert() и remove() выполняются за время O(logN), и каждый из них должен быть выполнена N раз, вся сортировка выполняется за время O(N*logN) — такое же, как при быстрой сортировке. Однако по скорости этот алгоритм все же уступает быстрой сортировке — отчасти из-за того, что во внутреннем цикле while метода trickleDown() выполняется больше операций, чем во внутреннем цикле быстрой сортировки.
Однако существует пара приемов, повышающих эффективность пирамидальной сортировки. Первый экономит время, а второй — память.

Ускорение – пункт 1

Вставляя в пирамиду N новых элементов, мы применяем метод trickleUp() N раз. Однако все элементы можно вставить в случайных позициях массива, а затем восстановить иерархию пирамиды всего N/2 применениями trickleDown(). Этот прием обеспечивает небольшой прирост скорости.

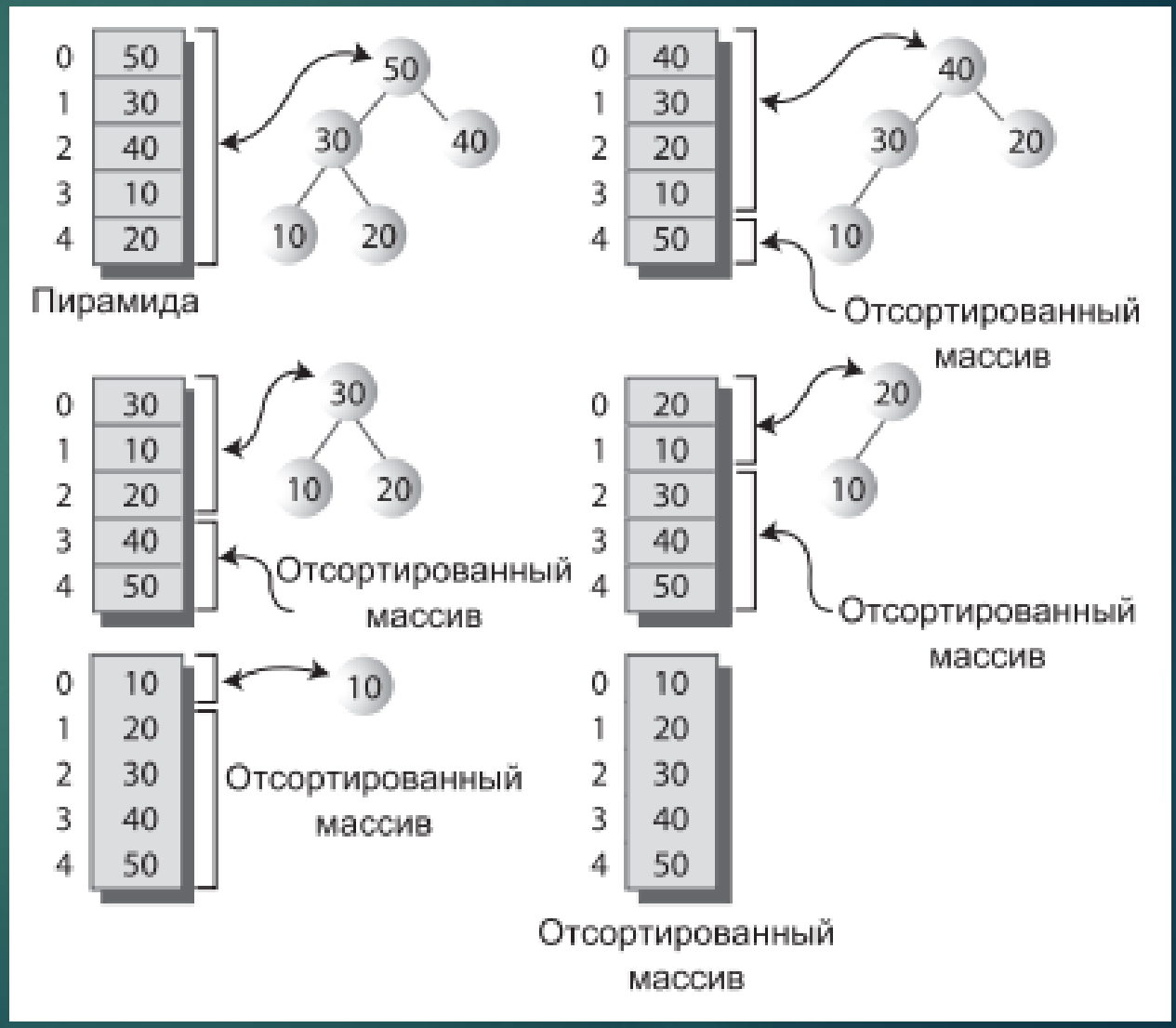


Ускорение – пункт 2

В исходном фрагменте кода неупорядоченные данные хранятся в массиве. Затем эти данные вставляются в пирамиду, затем извлекаются из нее и записываются обратно в массив в порядке сортировки. Для этой процедуры необходимы два массива размера N: исходный и массив, используемый для хранения пирамиды. В действительности для хранения пирамиды и исходных данных достаточно одного массива. Это вдвое сокращает объем памяти, необходимой для пирамидальной сортировки; алгоритм не требует дополнительных затрат памяти за пределами исходного массива. Мы уже видели, что массив может быть преобразован в пирамиду применением trickleDown() к половине элементов массива. Преобразование неупорядоченного массива в пирамиду осуществляется «на месте»; для выполнения этой операции достаточно одного массива. Таким образом, для первого шага пирамидальной сортировки достаточно одного массива. Однако ситуация усложняется с повторным применением remove() к пирамиде. Где хранить извлекаемые элементы?

Ускорение – пункт 2

При каждом извлечении элемента из пирамиды в конце массива появляется свободная ячейка; размер пирамиды уменьшается на единицу. Только что извлеченный элемент можно сохранить в освободившейся ячейке. С извлечением других элементов массив пирамиды становится все меньше, а массив упорядоченных данных — все больше. Таким образом, при соответствующем планировании упорядоченный массив и массив пирамиды смогут совместно использовать одно пространство



Эффективность

Как упоминалось ранее, пирамидальная сортировка выполняется за время O(N*logN). Хотя по скорости она слегка уступает быстрой сортировке, ее преимуществом перед быстрой сортировкой является меньшая чувствительность к исходному распределению данных. При некоторых конфигурациях ключей быстрая сортировка замедляется до O(N*N), тогда как пирамидальная сортировка выполняется за время O(N*logN) независимо от распределения данных.