Введение в Python

Сортировки. Продолжение.

Алексей Сорокин

спецкурс, ОТИПЛ МГУ,
осенний семестр 2016–2017 учебного года

Пирамида

Определение

Неубывающая пирамида — это массив, в котором для любого $0 < i < n$ выполняется неравенство $a[(i-1)//2] \geqslant a[i]$.

Пирамида

Определение

Неубывающая пирамида — это массив, в котором для любого $0 < i < n$ выполняется неравенство $a[(i-1)//2] \geqslant a[i]$.

Например, $[9, 5, 8, 3, 4, 6, 7, 1, 2]$ — невозрастающая пирамида:
$9 > 5,\ 9 > 8,\ 5 > 3,\ 5 > 4,\ 8 > 6,\ 8 > 7,\ 3 > 1,\ 3 > 2$.

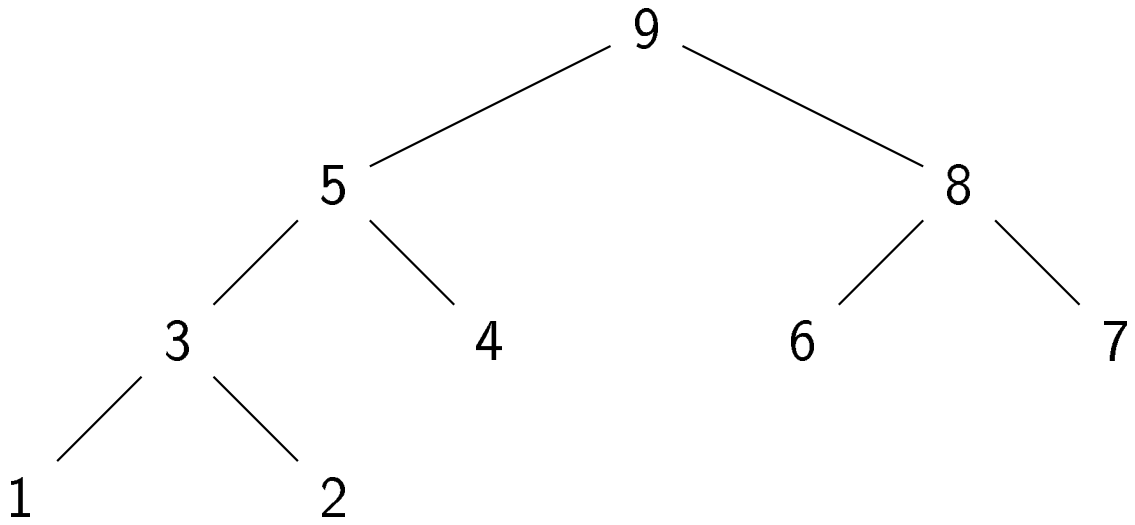
Пирамида

Определение

Неубывающая пирамида — это массив, в котором для любого $0 < i < n$ выполняется неравенство $a[(i-1)//2] \geqslant a[i]$.

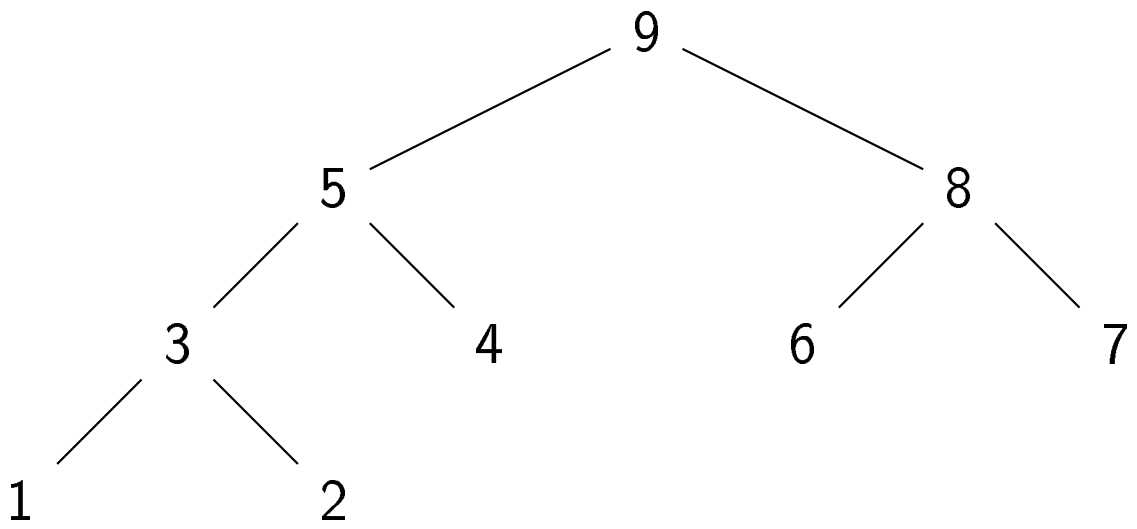
Например, $[9, 5, 8, 3, 4, 6, 7, 1, 2]$ — невозрастающая пирамида:
$9 > 5$, $9 > 8$, $5 > 3$, $5 > 4$, $8 > 6$, $8 > 7$, $3 > 1$, $3 > 2$.

Пирамиды можно изображать в виде дерева:



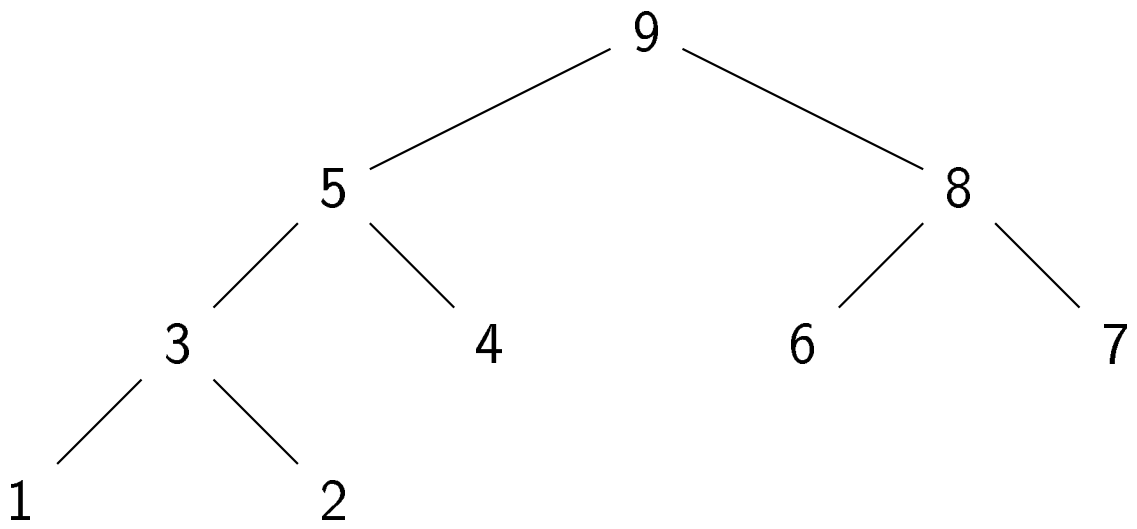
Пирамида

Пирамиды можно изображать в виде дерева:



Пирамида

Пирамиды можно изображать в виде дерева:

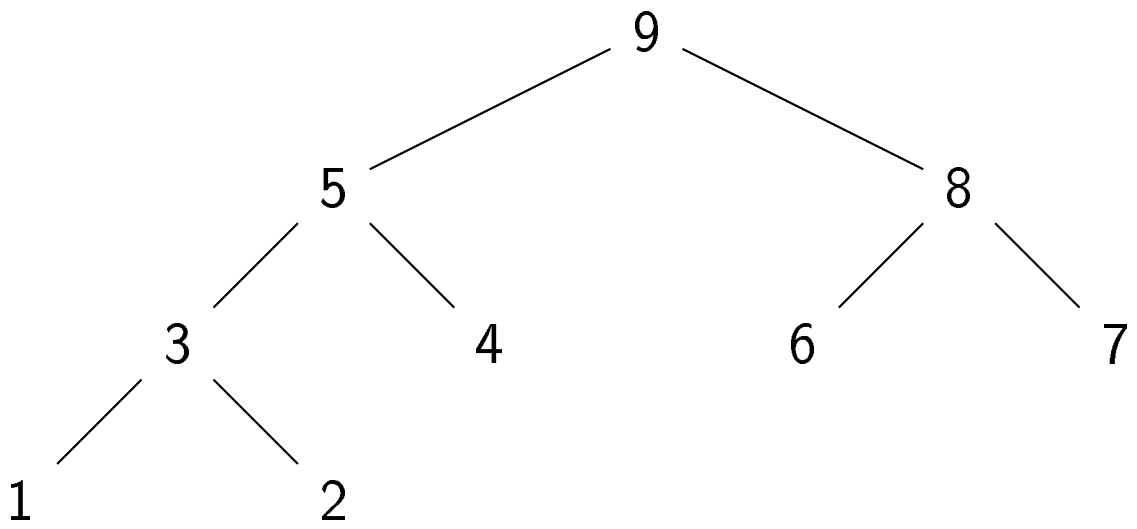


То есть пирамида — это неполное двоичное дерево, где

- Каждый уровень заполняется слева направо.

Пирамида

Пирамиды можно изображать в виде дерева:



То есть пирамида — это неполное двоичное дерево, где

- Каждый уровень заполняется слева направо.
- Родитель обязательно не меньше сына.

В корне пирамиды стоит максимальный элемент.

Пирамида и сортировка

- Как пирамида может пригодиться для сортировки?

Пирамида и сортировка

- Как пирамида может пригодиться для сортировки?
- Можно переставить максимальный элемент на последнюю позицию и выкинуть его из пирамиды
- После этого пирамида может испортиться...

Пирамида и сортировка

- Как пирамида может пригодиться для сортировки?
- Можно переставить максимальный элемент на последнюю позицию и выкинуть его из пирамиды
- После этого пирамида может испортиться...
- Если бы нам удалось быстро восстановить её правильность (невозрастание), то мы могли бы продолжить сортировку.

Пирамида и сортировка

- Как пирамида может пригодиться для сортировки?
- Можно переставить максимальный элемент на последнюю позицию и выкинуть его из пирамиды
- После этого пирамида может испортиться...
- Если бы нам удалось быстро восстановить её правильность (невозрастание), то мы могли бы продолжить сортировку.
- Заметим, что второй по максимальности элемент был одним из детей корневого.

Пирамида и сортировка

- Как пирамида может пригодиться для сортировки?
- Можно переставить максимальный элемент на последнюю позицию и выкинуть его из пирамиды
- После этого пирамида может испортиться...
- Если бы нам удалось быстро восстановить её правильность (невозрастание), то мы могли бы продолжить сортировку.
- Заметим, что второй по максимальности элемент был одним из детей корневого.
- Значит, если свойство пирамиды нарушено, надо обменять корень с максимальным из его детей.

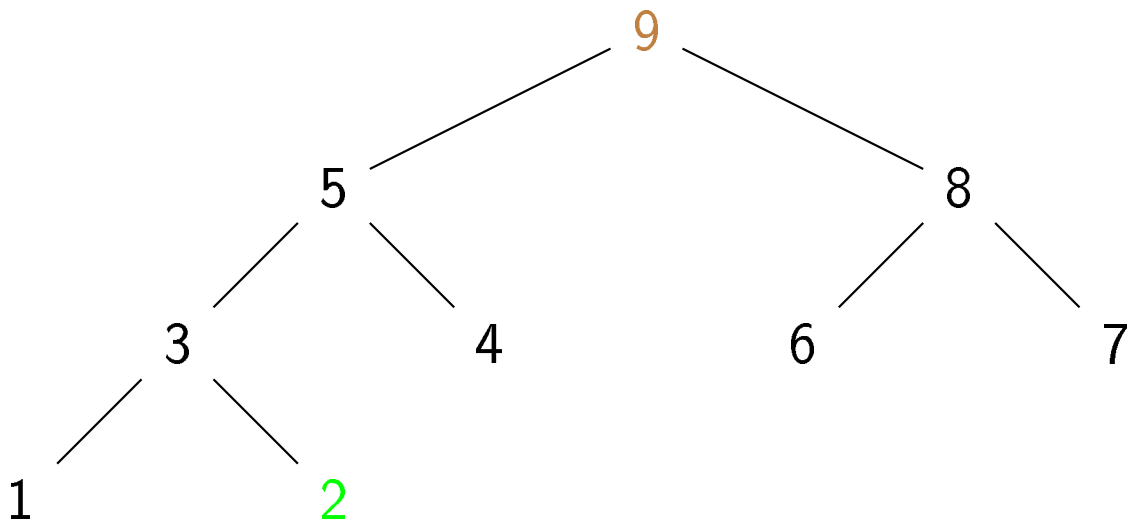
Пирамида и сортировка

- Как пирамида может пригодиться для сортировки?
- Можно переставить максимальный элемент на последнюю позицию и выкинуть его из пирамиды
- После этого пирамида может испортиться...
- Если бы нам удалось быстро восстановить её правильность (невозрастание), то мы могли бы продолжить сортировку.
- Заметим, что второй по максимальности элемент был одним из детей корневого.
- Значит, если свойство пирамиды нарушено, надо обменять корень с максимальным из его детей.
- После этого корректность может нарушаться на втором уровне...

Пирамида и сортировка

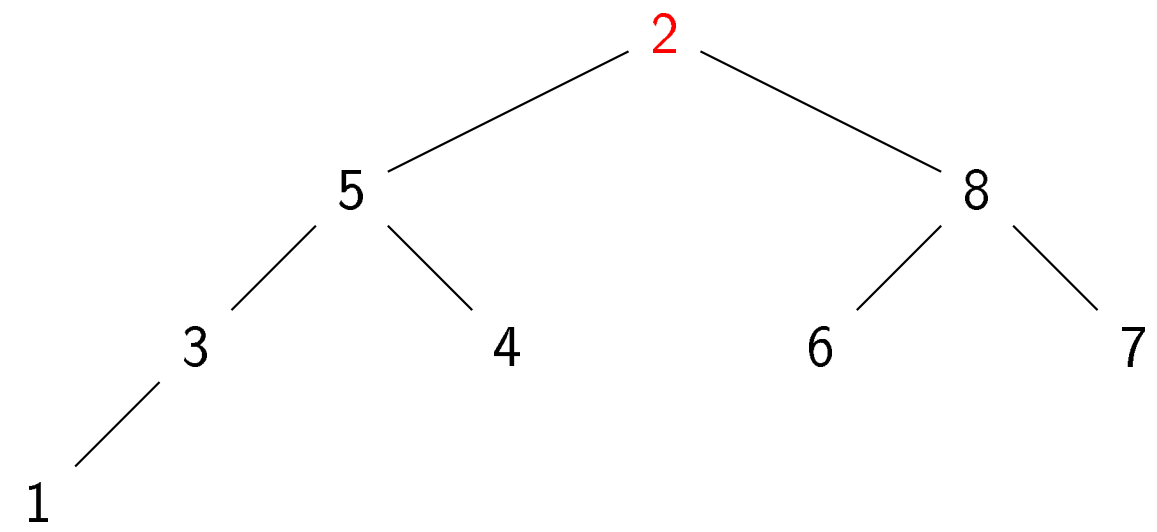
- Как пирамида может пригодиться для сортировки?
- Можно переставить максимальный элемент на последнюю позицию и выкинуть его из пирамиды
- После этого пирамида может испортиться...
- Если бы нам удалось быстро восстановить её правильность (невозрастание), то мы могли бы продолжить сортировку.
- Заметим, что второй по максимальности элемент был одним из детей корневого.
- Значит, если свойство пирамиды нарушено, надо обменять корень с максимальным из его детей.
- После этого корректность может нарушаться на втором уровне...
- Будем повторять процедуру, пока не спустимся до корня.

Преобразование пирамиды



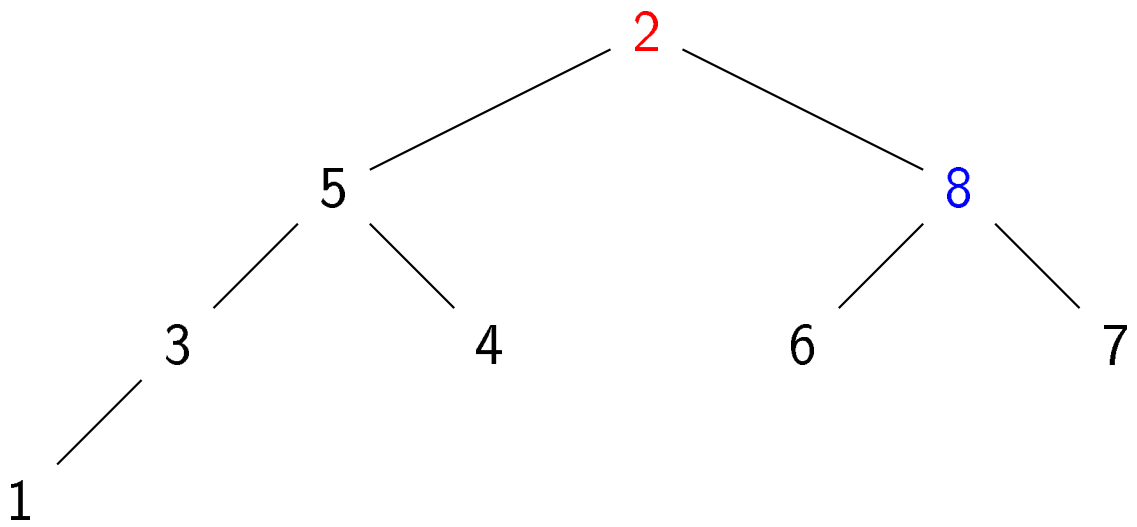
Преобразование пирамиды

[2, 5, 8, 3, 4, 6, 7, 1, 9]



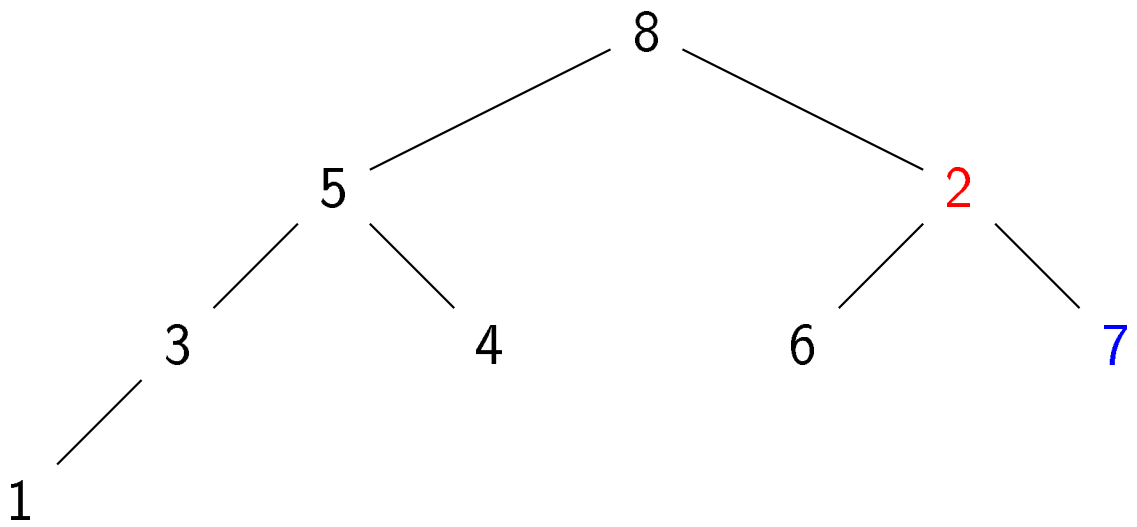
Преобразование пирамиды

[2, 5, 8, 3, 4, 6, 7, 1, 9]



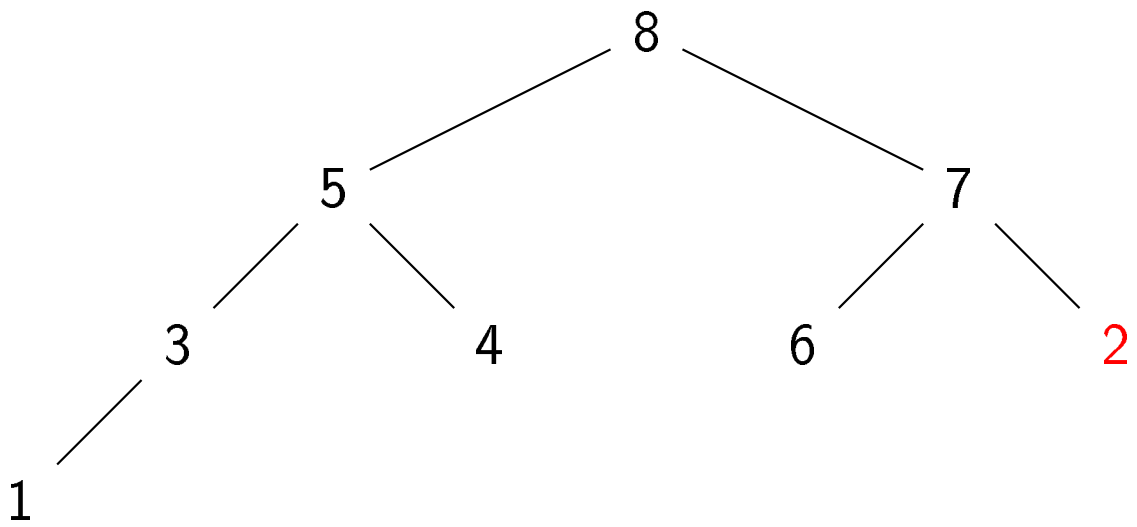
Преобразование пирамиды

[8, 5, 2, 3, 4, 6, 7, 1, 9]



Преобразование пирамиды

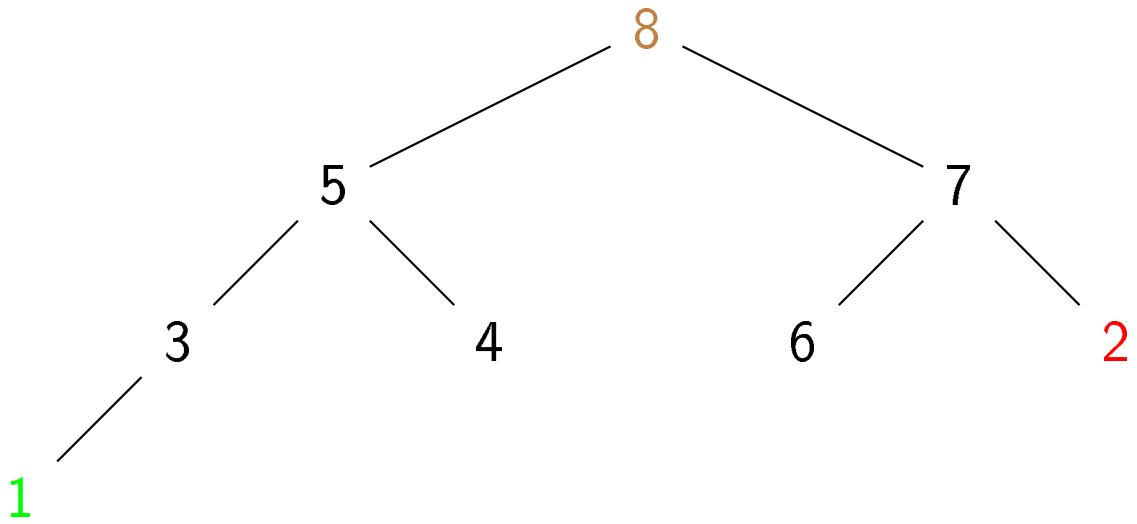
[8, 5, 7, 3, 4, 6, 2, 1, 9]



Преобразование пирамиды

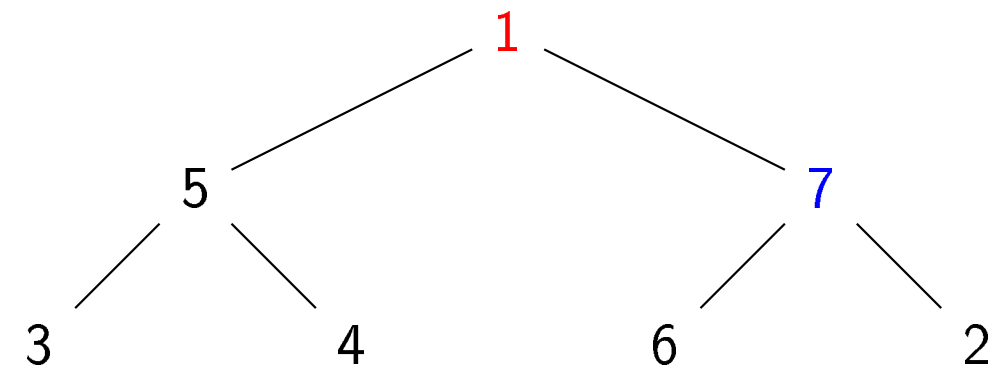
Преобразование пирамиды

[8, 5, 7, 3, 4, 6, 2, 1, 9]



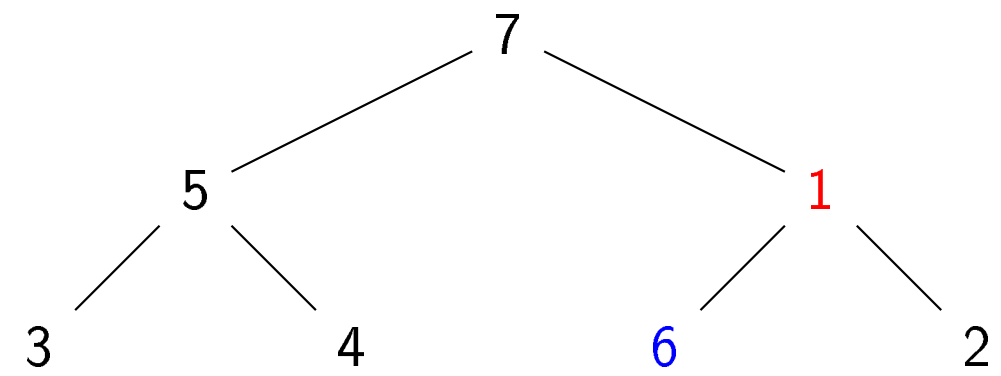
Преобразование пирамиды

[1, 5, 7, 3, 4, 6, 2, 8, 9]



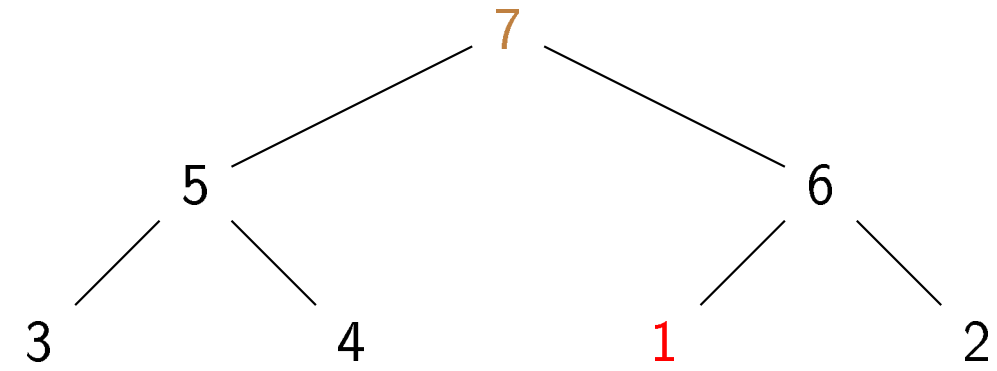
Преобразование пирамиды

[7, 5, 1, 3, 4, 6, 2, 8, 9]



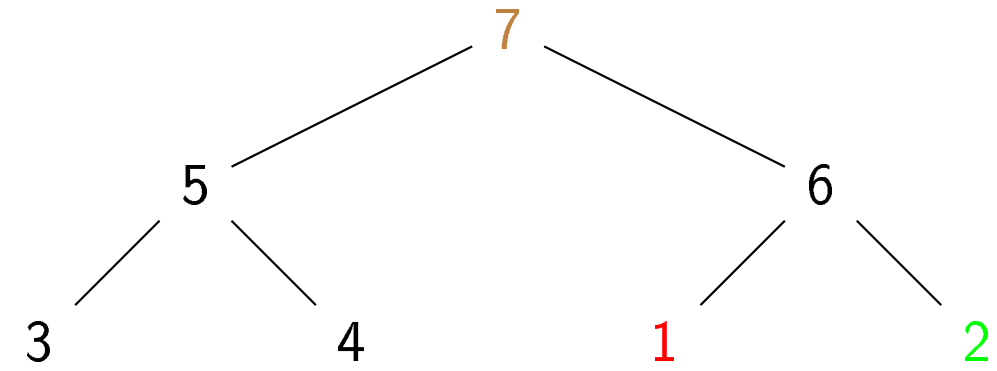
Преобразование пирамиды

[7, 5, 6, 3, 4, 1, 2, 8, 9]



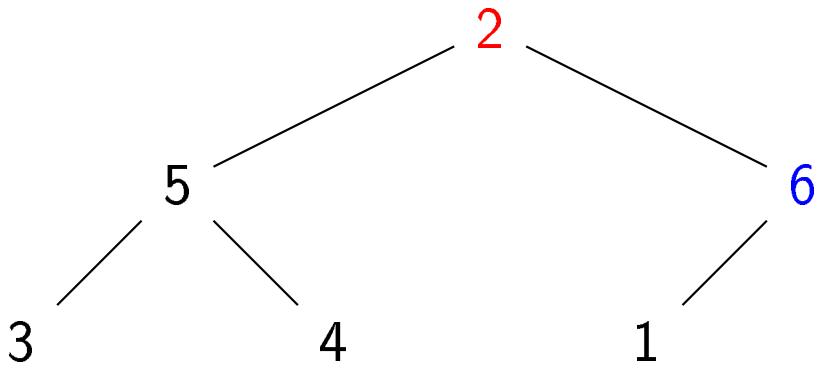
Преобразование пирамиды

[7, 5, 6, 3, 4, 1, 2, 8, 9]



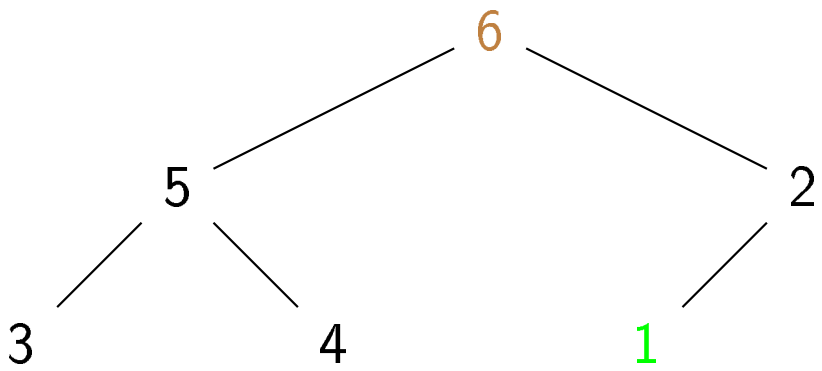
Преобразование пирамиды

[2, 5, 6, 3, 4, 1, 7, 8, 9]



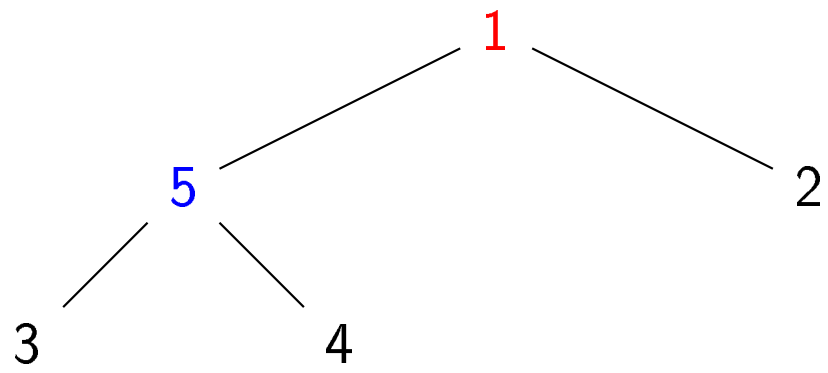
Преобразование пирамиды

[6, 5, 2, 3, 4, 1, 7, 8, 9]



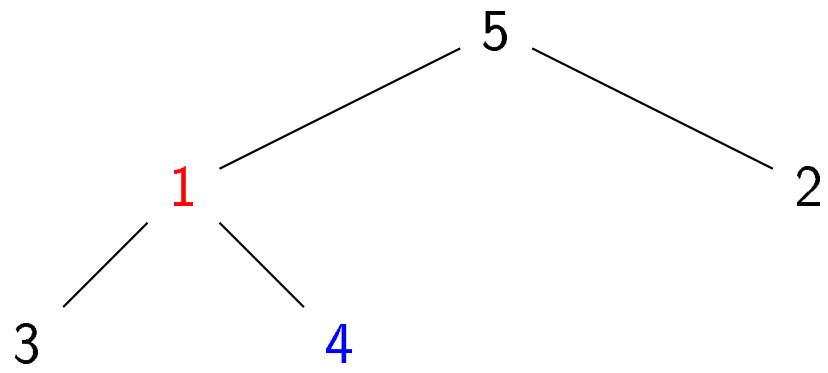
Преобразование пирамиды

[1, 5, 2, 3, 4, 6, 7, 8, 9]



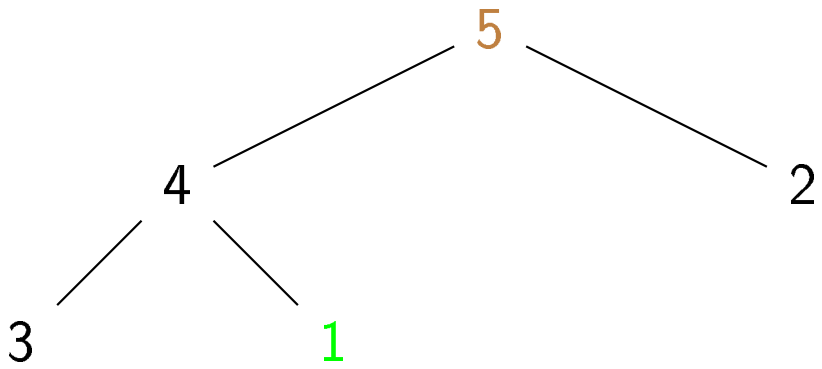
Преобразование пирамиды

[5, 1, 2, 3, 4, 6, 7, 8, 9]



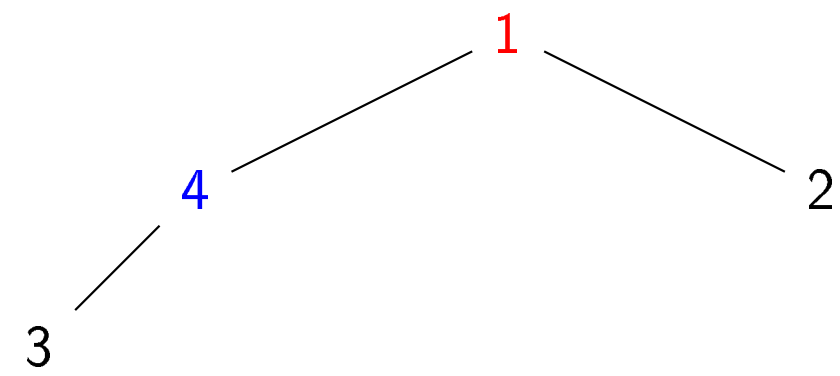
Преобразование пирамиды

[5, 4, 2, 3, 1, 6, 7, 8, 9]



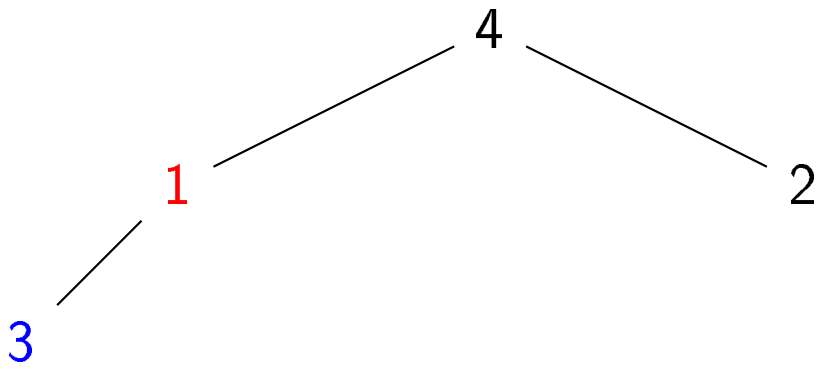
Преобразование пирамиды

[1, 4, 2, 3, 5, 6, 7, 8, 9]



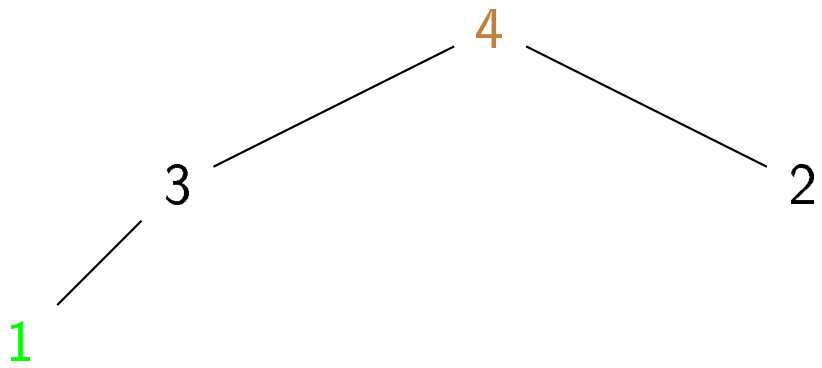
Преобразование пирамиды

[4, 1, 2, 3, 5, 6, 7, 8, 9]



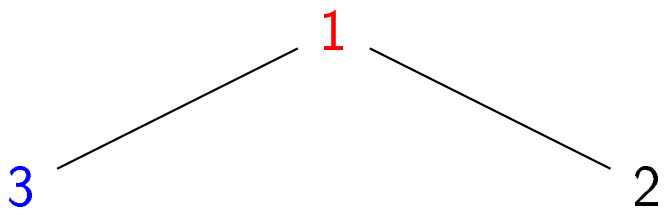
Преобразование пирамиды

[4, 3, 2, 1, 5, 6, 7, 8, 9]



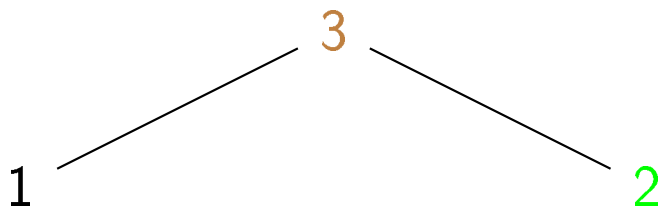
Преобразование пирамиды

[1, 3, 2, 4, 5, 6, 7, 8, 9]



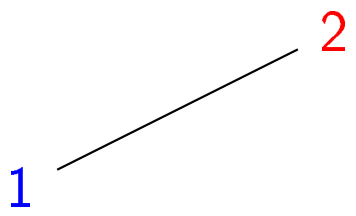
Преобразование пирамиды

[3, 1, 2, 4, 5, 6, 7, 8, 9]



Преобразование пирамиды

[2, 1, 3, 4, 5, 6, 7, 8, 9]



Преобразование пирамиды

[1, 2, 3, 4, 5, 6, 7, 8, 9]

Преобразование пирамиды

Сортировка — многократный обмен корня с последним элементом, уменьшение пирамиды и “проталкивание” нового корня по ветви.

Преобразование пирамиды

Сортировка — многократный обмен корня с последним элементом, уменьшение пирамиды и “проталкивание” нового корня по ветви

Утверждение

Сложность преобразования невозрастающей пирамиды в упорядоченный массив — $O(n \log_2 n)$ арифметических операций.

Преобразование пирамиды

Сортировка — многократный обмен корня с последним элементом, уменьшение пирамиды и “проталкивание” нового корня по ветви

Утверждение

Сложность преобразования невозрастающей пирамиды в упорядоченный массив — $O(n \log_2 n)$ арифметических операций.

Доказательство

- Преобразование занимает n итераций. На каждой из них тратится $O(1)$ операций на обмен корня и последнего элемента и уменьшение пирамиды.

Преобразование пирамиды

Сортировка — многократный обмен корня с последним элементом, уменьшение пирамиды и “проталкивание” нового корня по ветви

Утверждение

Сложность преобразования невозрастающей пирамиды в упорядоченный массив — $O(n \log_2 n)$ арифметических операций.

Доказательство

- Преобразование занимает n итераций. На каждой из них тратится $O(1)$ операций на обмен корня и последнего элемента и уменьшение пирамиды.

Преобразование пирамиды

Сортировка — многократный обмен корня с последним элементом, уменьшение пирамиды и “проталкивание” нового корня по ветви

Утверждение

Сложность преобразования невозрастающей пирамиды в упорядоченный массив — $O(n \log_2 n)$ арифметических операций.

Доказательство

- Преобразование занимает n итераций. На каждой из них тратится $O(1)$ операций на обмен корня и последнего элемента и уменьшение пирамиды.
- Кроме того, $O(m)$ операций занимает проталкивание нового корня вниз по ветви. Здесь m — длина ветви

Преобразование пирамиды

Сортировка — многократный обмен корня с последним элементом, уменьшение пирамиды и “проталкивание” нового корня по ветви

Утверждение

Сложность преобразования невозрастающей пирамиды в упорядоченный массив — $O(n \log_2 n)$ арифметических операций.

Доказательство

- Преобразование занимает n итераций. На каждой из них тратится $O(1)$ операций на обмен корня и последнего элемента и уменьшение пирамиды.
- Кроме того, $O(m)$ операций занимает проталкивание нового корня вниз по ветви. Здесь m — длина ветви
- Однако длина максимальной ветви в дереве не превышает $\lceil \log_2 n \rceil$ элементов.

Преобразование пирамиды

Сортировка — многократный обмен корня с последним элементом, уменьшение пирамиды и “проталкивание” нового корня по ветви

Утверждение

Сложность преобразования невозрастающей пирамиды в упорядоченный массив — $O(n \log_2 n)$ арифметических операций.

Доказательство

- Преобразование занимает n итераций. На каждой из них тратится $O(1)$ операций на обмен корня и последнего элемента и уменьшение пирамиды.
- Кроме того, $O(m)$ операций занимает проталкивание нового корня вниз по ветви. Здесь m — длина ветви
- Однако длина максимальной ветви в дереве не превышает $\lceil \log_2 n \rceil$ элементов.
- Тогда на всю итерацию будет затрачено $O(\log_2 n)$ операций.

Построение пирамиды

- Из невозрастающей пирамиды за $O(n \log_2 n)$ можно построить упорядоченный массив.

Построение пирамиды

- Из невозрастающей пирамиды за $O(n \log_2 n)$ можно построить упорядоченный массив.
- Как построить саму пирамиду?

Построение пирамиды

- Из невозрастающей пирамиды за $O(n \log_2 n)$ можно построить упорядоченный массив.
- Как построить саму пирамиду?
- Будем добавлять элементы по одному с сохранением свойства пирамиды.

Построение пирамиды

- Из невозрастающей пирамиды за $O(n \log_2 n)$ можно построить упорядоченный массив.
- Как построить саму пирамиду?
- Будем добавлять элементы по одному с сохранением свойства пирамиды.
- Если раньше мы проталкивали корень вниз, то теперь будем поднимать последний лист вверх.

Построение пирамиды

- Из невозрастающей пирамиды за $O(n \log_2 n)$ можно построить упорядоченный массив.
- Как построить саму пирамиду?
- Будем добавлять элементы по одному с сохранением свойства пирамиды.
- Если раньше мы проталкивали корень вниз, то теперь будем поднимать последний лист вверх.
- Более быстрый вариант: продвигать вниз по максимуму все элементы, начиная с последних.

Добавление в пирамиду

```
 1: function AddToHeap(a, x)
 2:     n = len(a)
 3:     a.append(x                          ▷ Добавляем элемент в пирамиду
 4:     pos = n
 5:     next_pos = pos//2
 6:     ▷ продвигаем элемент в пирамиде вверх
 7:     while pos > 0 and a[next_pos] < a[pos] do
 8:         ▷ Обмениваем элемент с родителем
 9:         a[next_pos], a[pos] = a[pos], a[next_pos]
10:         pos = next_pos
11:         next_pos = pos//2
12:     end while
13: return a
14: end function
```

Преобразование списка в пирамиду

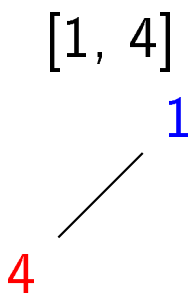
Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.

$[1]$

1

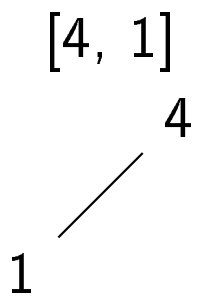
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



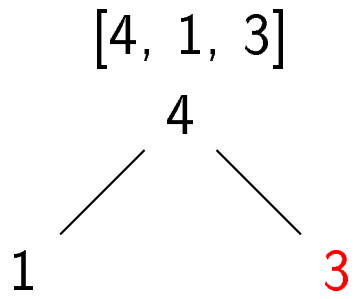
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



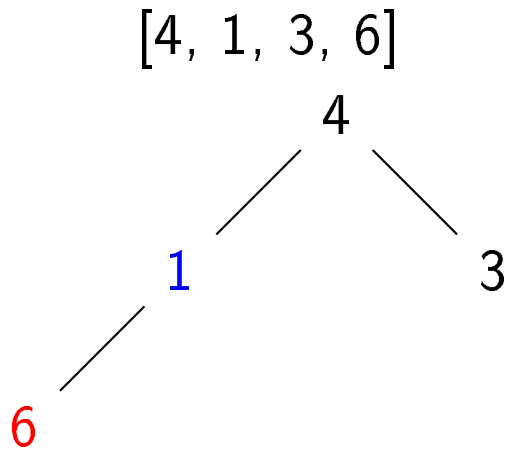
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



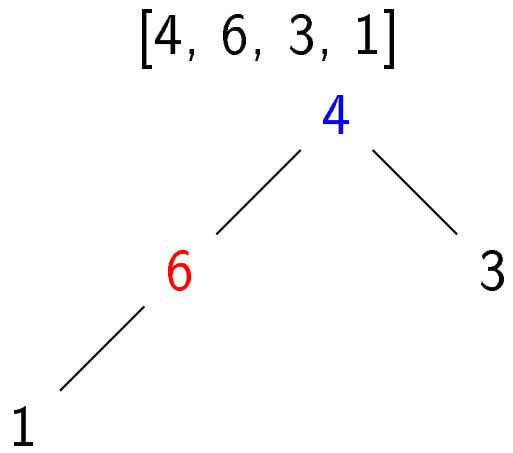
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



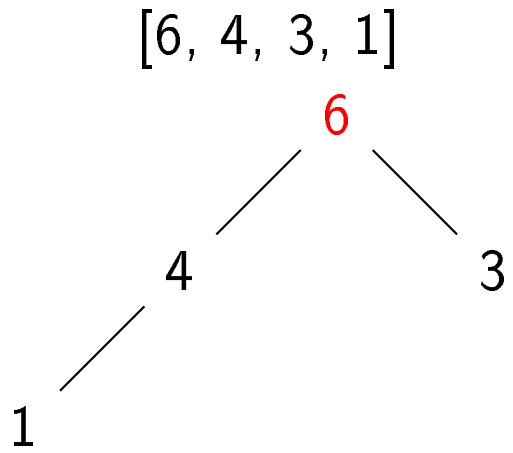
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



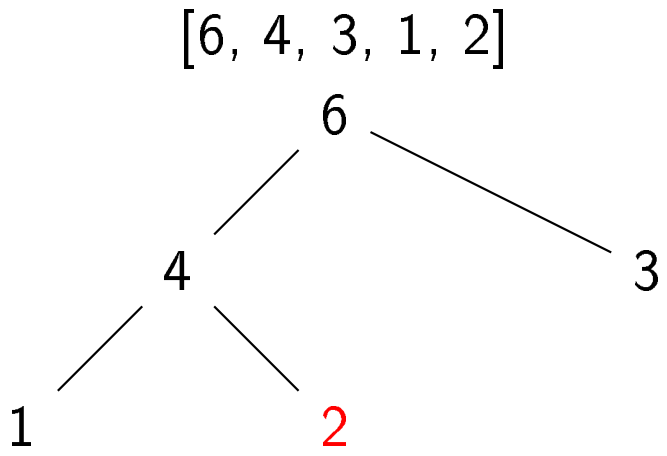
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



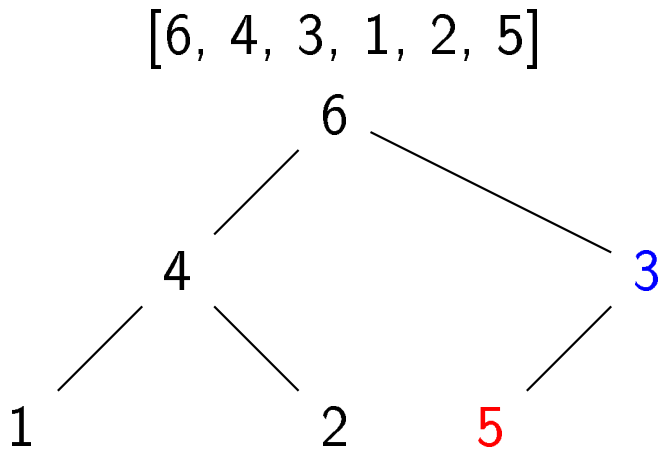
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



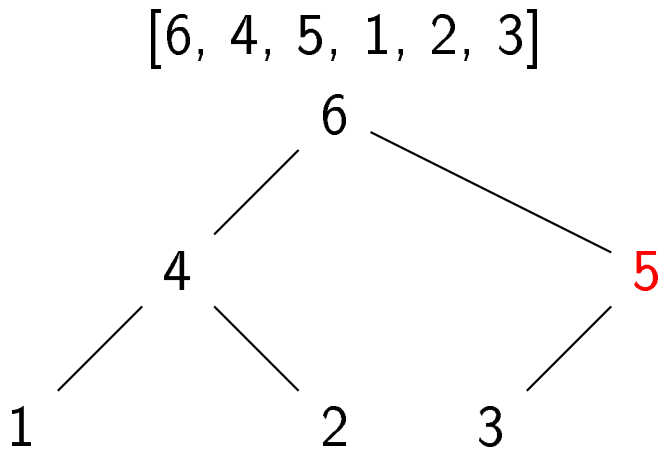
Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



Преобразование списка в пирамиду

Преобразуем список $[1, 4, 3, 6, 2, 5]$ в пирамиду.



Пирамида как структура данных

- Структура данных “пирамида” содержит операции:
  - $\mathrm{Top}()$ — посмотреть минимальный элемент.
  - $\mathrm{ExtractMin}()$ — извлечь минимальный элемент.
  - $\mathrm{Add}()$ — добавить элемент в пирамиду.
  - $\mathrm{size}()$ — число элементов в пирамиде.

Пирамида как структура данных

- Структура данных “пирамида” содержит операции:
  - $\mathrm{Top}()$ — посмотреть минимальный элемент.
  - $\mathrm{ExtractMin}()$ — извлечь минимальный элемент.
  - $\mathrm{Add}()$ — добавить элемент в пирамиду.
  - $\mathrm{size}()$ — число элементов в пирамиде.
- Дополнительная операция $\mathrm{ChangeKey}(i, x)$ — поменять значение элемента в позиции x.

Пирамида как структура данных

- Структура данных “пирамида” содержит операции:
  - $\mathrm{Top}()$ — посмотреть минимальный элемент.
  - $\mathrm{ExtractMin}()$ — извлечь минимальный элемент.
  - $\mathrm{Add}()$ — добавить элемент в пирамиду.
  - $\mathrm{size}()$ — число элементов в пирамиде.
- Дополнительная операция $\mathrm{ChangeKey}(i, x)$ — поменять значение элемента в позиции x.
- Вспомогательные операции:
  - $\mathrm{SiftUp}(i)$ — протолкнуть вверх на своё место элемент из позиции i.
  - $\mathrm{SiftUp}(i)$ — протолкнуть вниз на своё место элемент из позиции i.

Пирамида как структура данных

- Структура данных “пирамида” содержит операции:
  - $\mathrm{Top}()$ — посмотреть минимальный элемент.
  - $\mathrm{ExtractMin}()$ — извлечь минимальный элемент.
  - $\mathrm{Add}()$ — добавить элемент в пирамиду.
  - $\mathrm{size}()$ — число элементов в пирамиде.
- Дополнительная операция $\mathrm{ChangeKey}(i, x)$ — поменять значение элемента в позиции x.
- Вспомогательные операции:
  - $\mathrm{SiftUp}(i)$ — протолкнуть вверх на своё место элемент из позиции i.
  - $\mathrm{SiftUp}(i)$ — протолкнуть вниз на своё место элемент из позиции i.
- Через SiftUp реализуется добавление в пирамиду и $\mathrm{ChangeKey}(i, x)$ при $x < a[i]$.
- Через $\mathrm{SiftDown}$ реализуется удаление минимального и $\mathrm{ChangeKey}(i, x)$ при $x > a[i]$.

Очередь с приоритетом

- Очередь с приоритетом поддерживает операции:
  - $\mathrm{Add}(key, value)$ — добавить объект *key* с приоритетом *value*.
  - $\mathrm{ExtractMin}$ — извлечь объект минимального приоритета.
  - $\mathrm{ChangeKey}(key, value)$ — изменить приоритет объекта *key* на *value*.

Очередь с приоритетом

- Очередь с приоритетом поддерживает операции:
  - $\mathrm{Add}(\mathit{key}, \mathit{value})$ — добавить объект *key* с приоритетом *value*.
  - $\mathrm{ExtractMin}$ — извлечь объект минимального приоритета.
  - $\mathrm{ChangeKey}(\mathit{key}, \mathit{value})$ — изменить приоритет объекта *key* на *value*.
- Реализация очереди с приоритетом — словарь+пирамида :
  - Словарь хранит пары $\langle \text{ключ} \rangle : \langle \text{позиция\_в\_пирамиде} \rangle$.
  - Пирамида хранит пары $\langle \text{ключ}, \text{значение} \rangle$ со значением в роди приоритета.
  - Это нужно, чтобы изменения словаря можно было отразить в пирамиде и наоборот.

Применение очереди с приоритетом

- Вычисление расстояния в графе (алгоритм Дейкстры): в очереди хранятся вершины, приоритет — текущее значение расстояния.

Применение очереди с приоритетом

- Вычисление расстояния в графе (алгоритм Дейкстры): в очереди хранятся вершины, приоритет — текущее значение расстояния.
- Сжатие информации: кодирование Хаффмена — символы хранятся по возрастанию частотности.

Применение очереди с приоритетом

- Вычисление расстояния в графе (алгоритм Дейкстры): в очереди хранятся вершины, приоритет — текущее значение расстояния.
- Сжатие информации: кодирование Хаффмена — символы хранятся по возрастанию частотности.
- Задачи ранжирования гипотез (морфологический и синтаксический анализ, машинный перевод, ...): гипотезы хранятся по возрастанию текущего штрафа, на каждом шаге извлекается и обрабатывается наилучшая частичная гипотеза.
- Задачи планирования и распределения заданий.

Реализация очередей с приоритетом в Python

- Реализация в Python: модуль Heapdict (`https://pypi.python.org/pypi/HeapDict`). Очередь с приоритетом: heapdict.heapdict().
- Поддерживает все методы словаря + метод popitem() (peekitem()), удаляющий (возвращающий без удаления) пару (*key*, *value*) для минимального значения value в очереди.

Реализация очередей с приоритетом в Python

- Реализация в Python: модуль Heapdict (`https://pypi.python.org/pypi/HeapDict`). Очередь с приоритетом: heapdict.heapdict().
- Поддерживает все методы словаря + метод popitem() (peekitem()), удаляющий (возвращающий без удаления) пару (*key*, *value*) для минимального значения value в очереди.
- Также можно использовать SortedDict из модуля SortedContainers (`https://pypi.python.org/pypi/sortedcontainers`).