Жадные алгоритмы: кучи

Александр Куликов

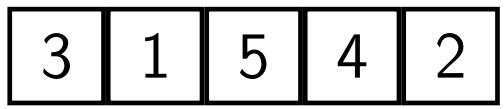
Онлайн-курс «Алгоритмы: теория и практика. Методы»
http://stepic.org/217

Очередь с приоритетами

Insert(p)	добавляет новый элемент с приоритетом p
Remove(it)	удаляет элемент, на который указывает итератор it
GetMin()	возвращает элемент с минимальным приоритетом
ExtractMin()	извлекает из очереди элемент с минимальным приоритетом
ChangePriority(it, p)	изменяет приоритет элемента, на который указывает итератор it, на p

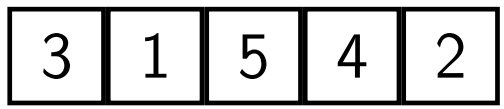
Простейшие реализации

- массив: `GETMIN` имеет время работы $O(n)$

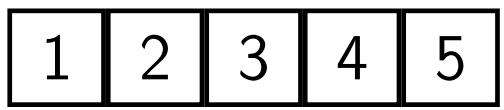


Простейшие реализации

- массив: GetMin имеет время работы $O(n)$



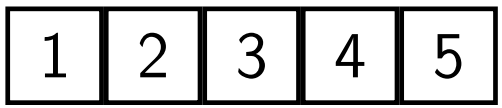
- упорядоченный массив: GetMin — $O(1)$, Remove — $O(n)$



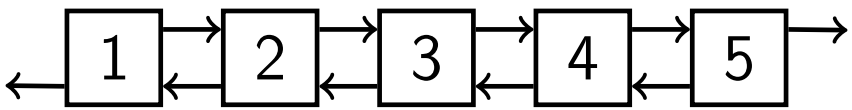
Простейшие реализации

- массив: GetMin имеет время работы $O(n)$

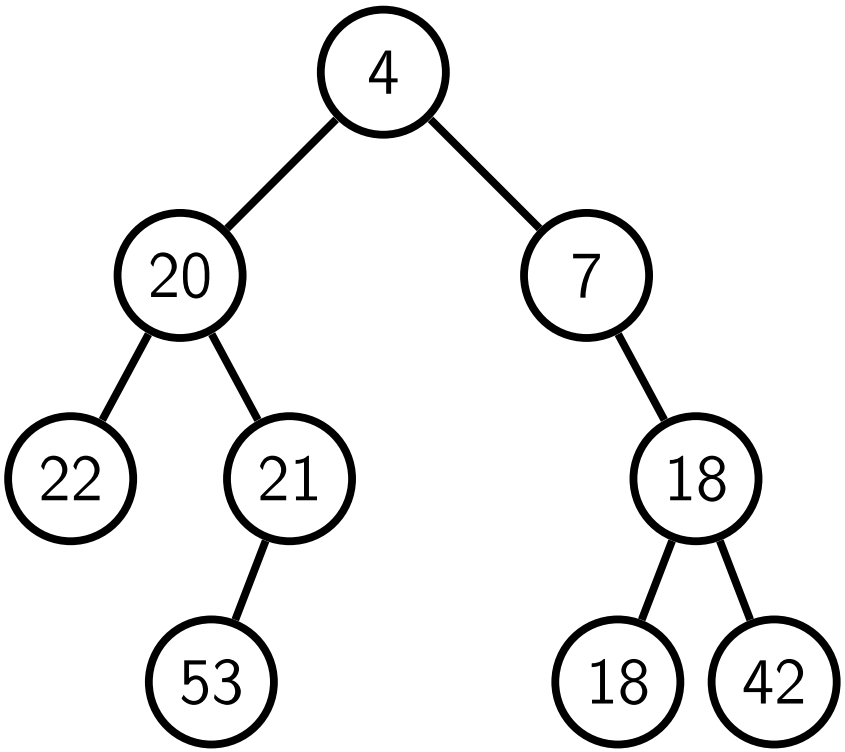
- упорядоченный массив: GetMin — $O(1)$, Remove — $O(n)$



- упорядоченный список: GetMin and Remove — $O(1)$, Insert — $O(n)$

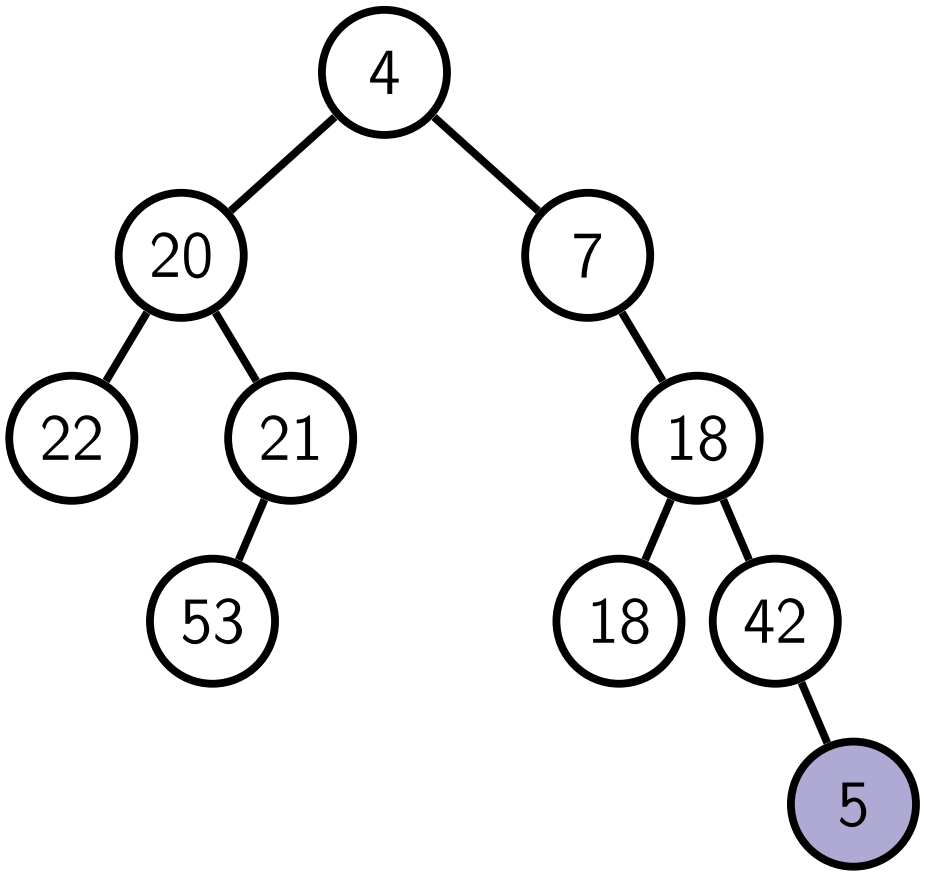


Двоичная (мин-)куча



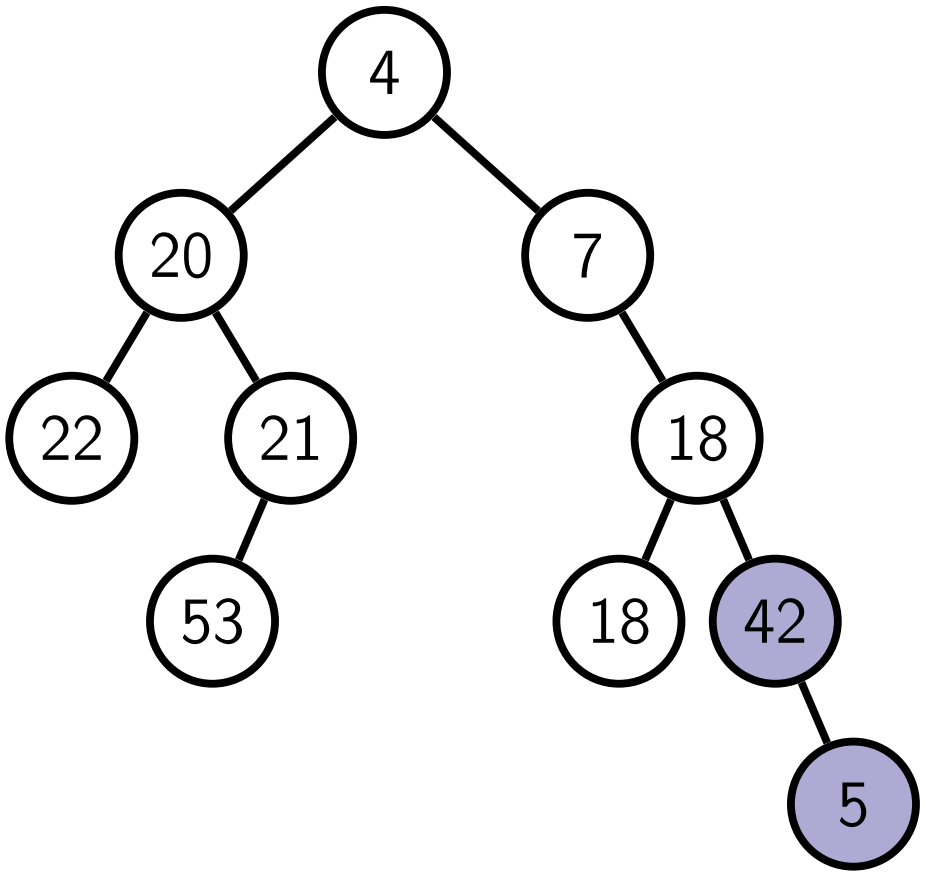
- основное свойство (мин-)кучи: значение вершины $\leq$ значений её детей
- минимальное значение хранится в корне, поэтому GetMin работает за время $O(1)$

Вставка и просеивание вверх



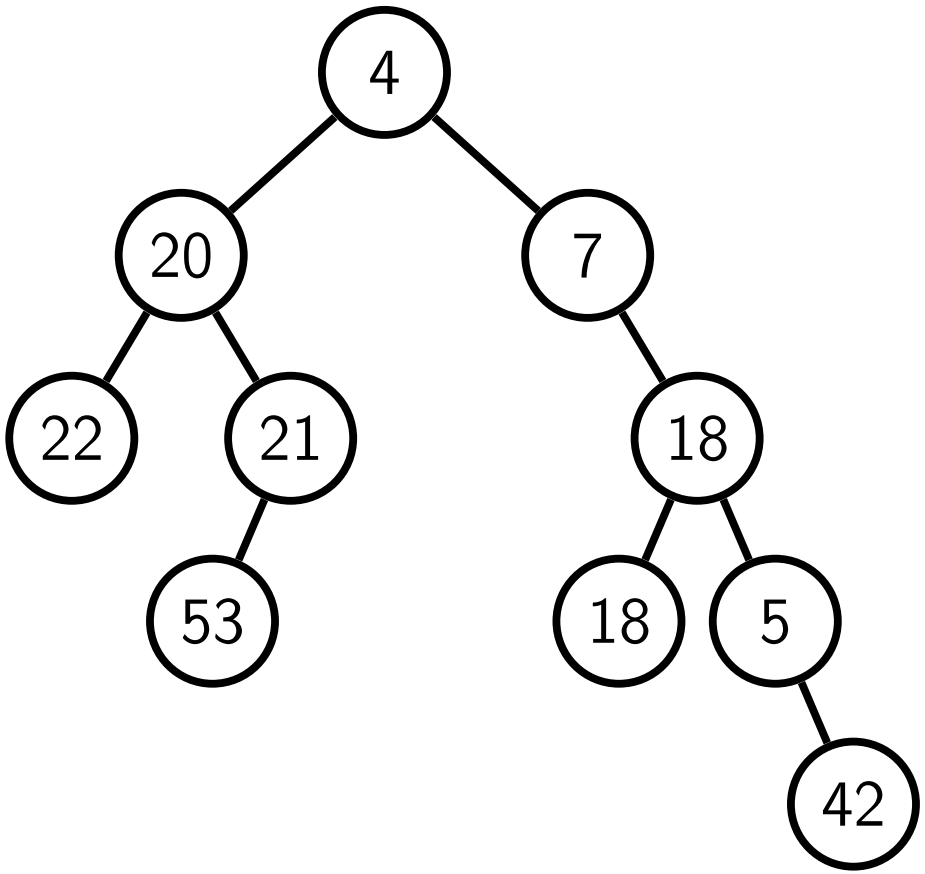
- подвесим новый элемент листом в произвольное место

Вставка и просеивание вверх



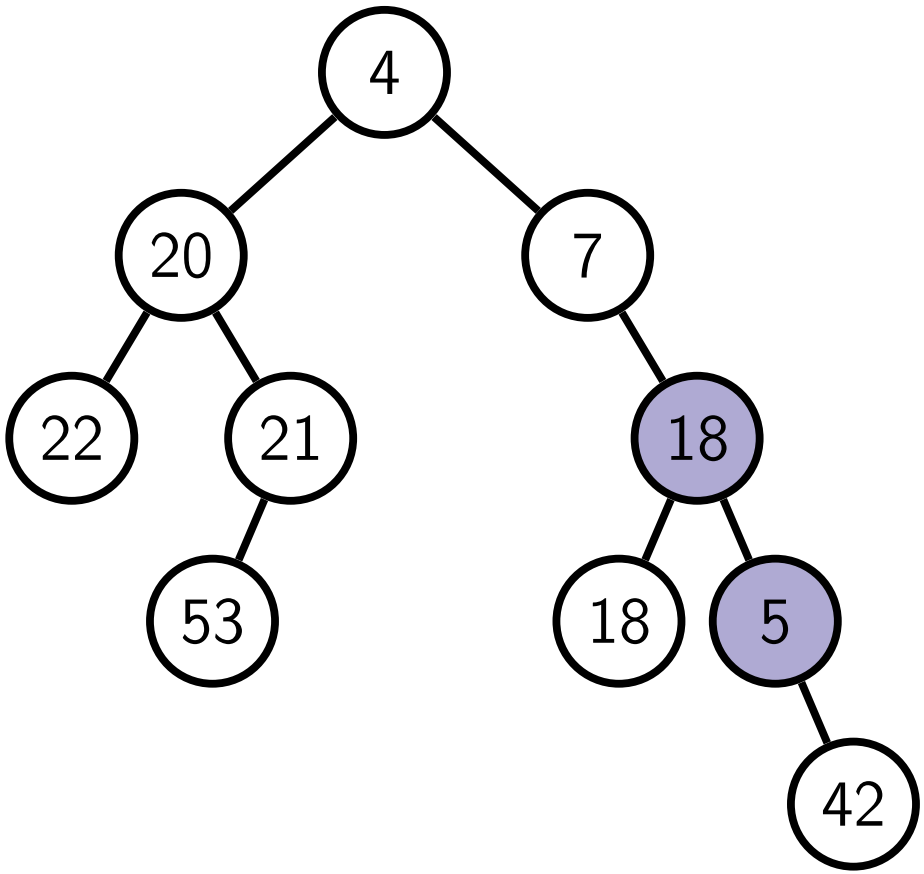
- подвесим новый элемент листом в произвольное место
- начнём чинить свойство кучи, просеивая проблемную вершину вверх

Вставка и просеивание вверх



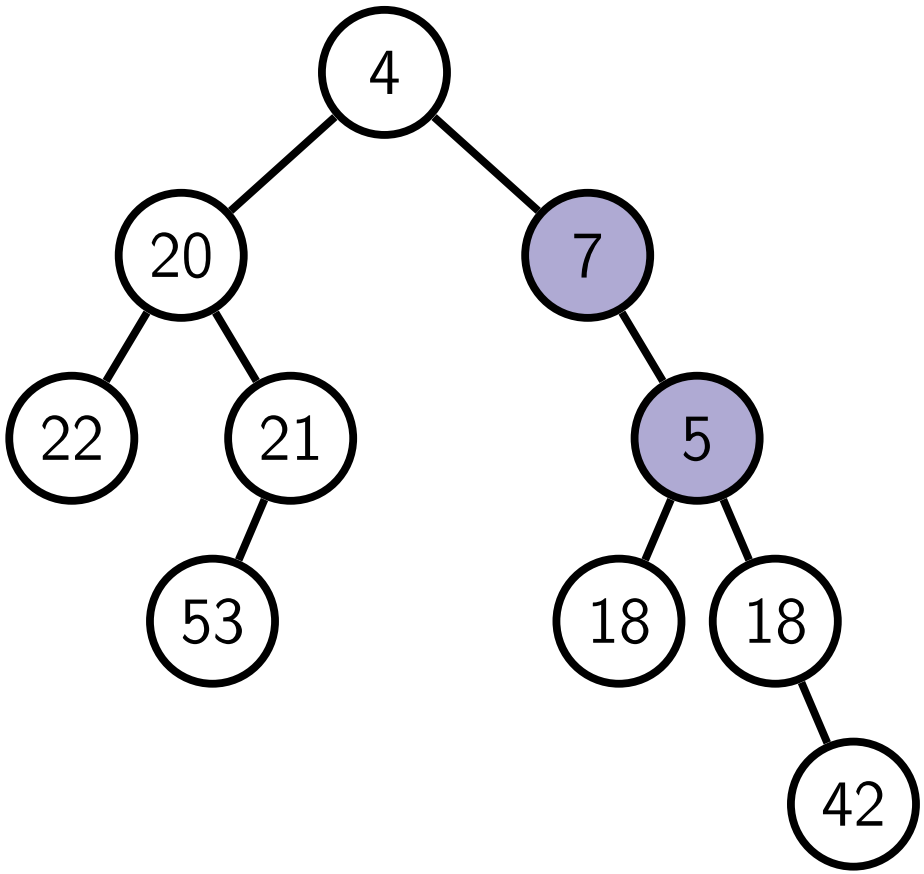
- подвесим новый элемент листом в произвольное место
- начнём чинить свойство кучи, просеивая проблемную вершину вверх

Вставка и просеивание вверх



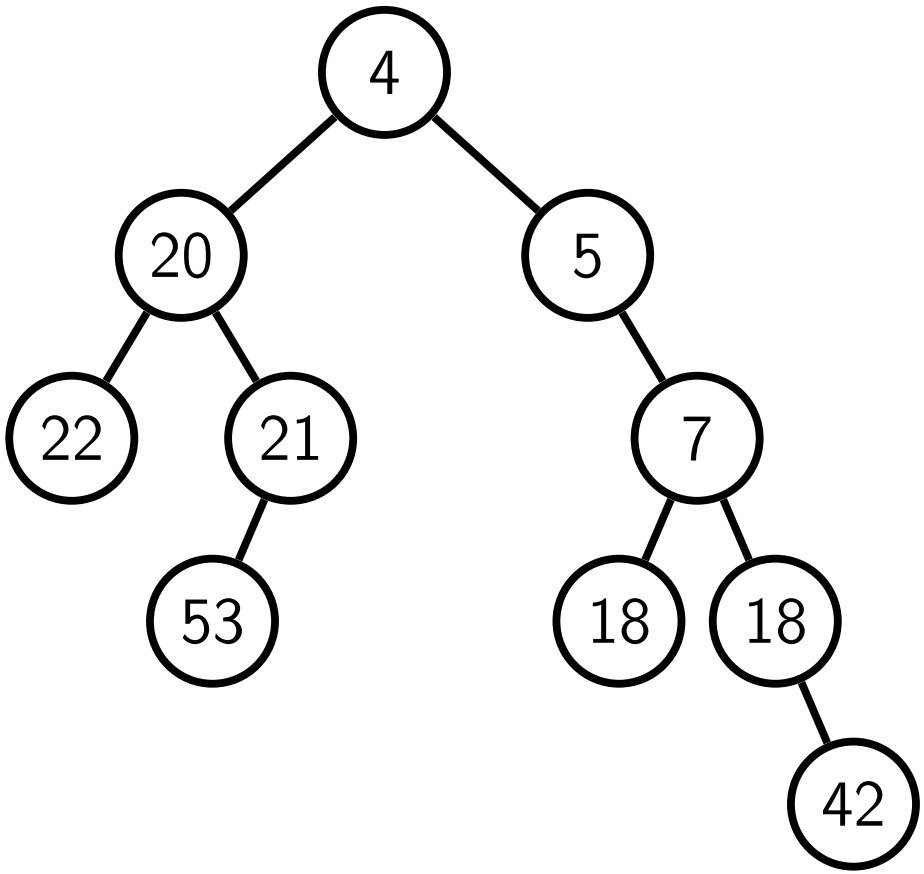
- подвесим новый элемент листом в произвольное место
- начнём чинить свойство кучи, просеивая проблемную вершину вверх

Вставка и просеивание вверх



- подвесим новый элемент листом в произвольное место
- начнём чинить свойство кучи, просеивая проблемную вершину вверх

Вставка и просеивание вверх



- подвесим новый элемент листом в произвольное место
- начнём чинить свойство кучи, просеивая проблемную вершину вверх

Время работы операции вставки

- важный инвариант: в каждый момент времени свойство кучи нарушено не более чем в одной вершине

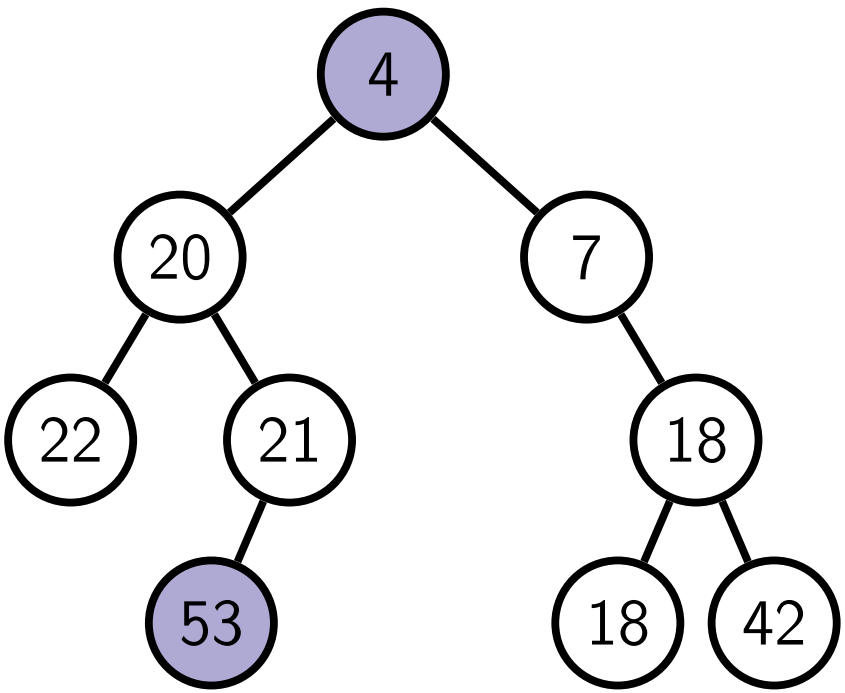
Время работы операции вставки

- важный инвариант: в каждый момент времени свойство кучи нарушено не более чем в одной вершине
- при просеивании вверх эта вершина становится ближе к корню

Время работы операции вставки

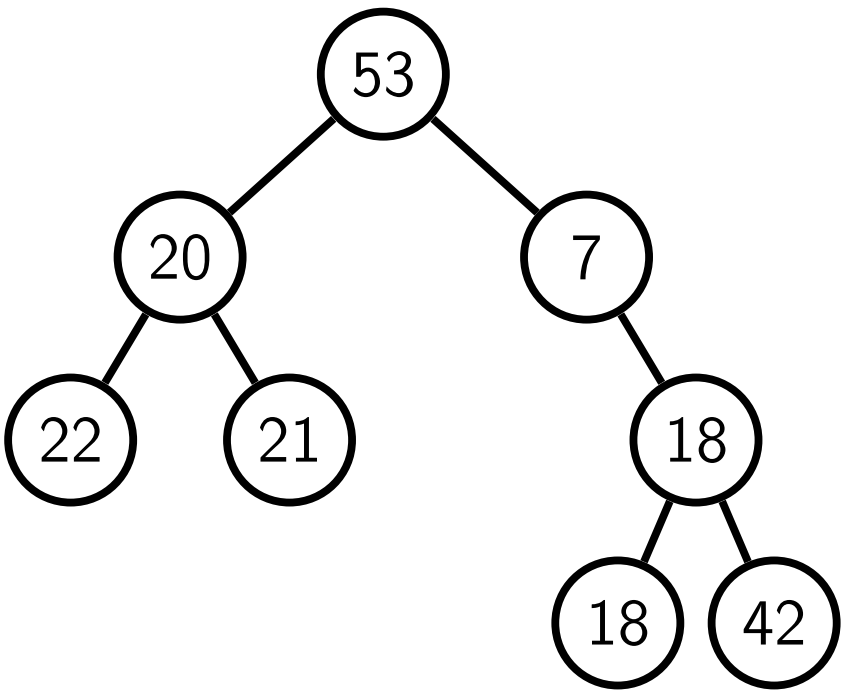
- важный инвариант: в каждый момент времени свойство кучи нарушено не более чем в одной вершине
- при просеивании вверх эта вершина становится ближе к корню
- время работы есть $O(h)$, где h — высота кучи

Извлечение минимума и просеивание вниз



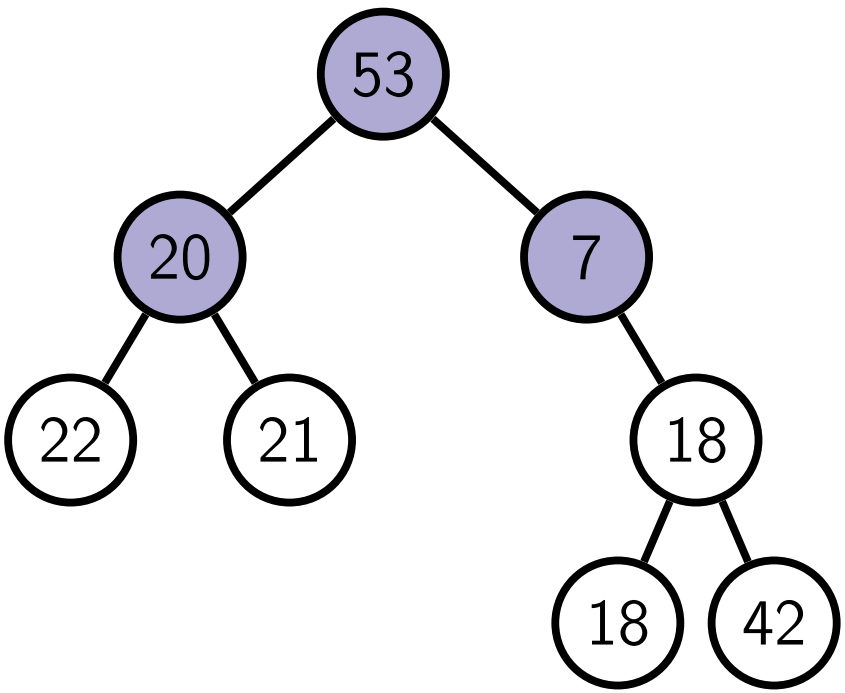
- заменим корень на любой лист

Извлечение минимума и просеивание вниз



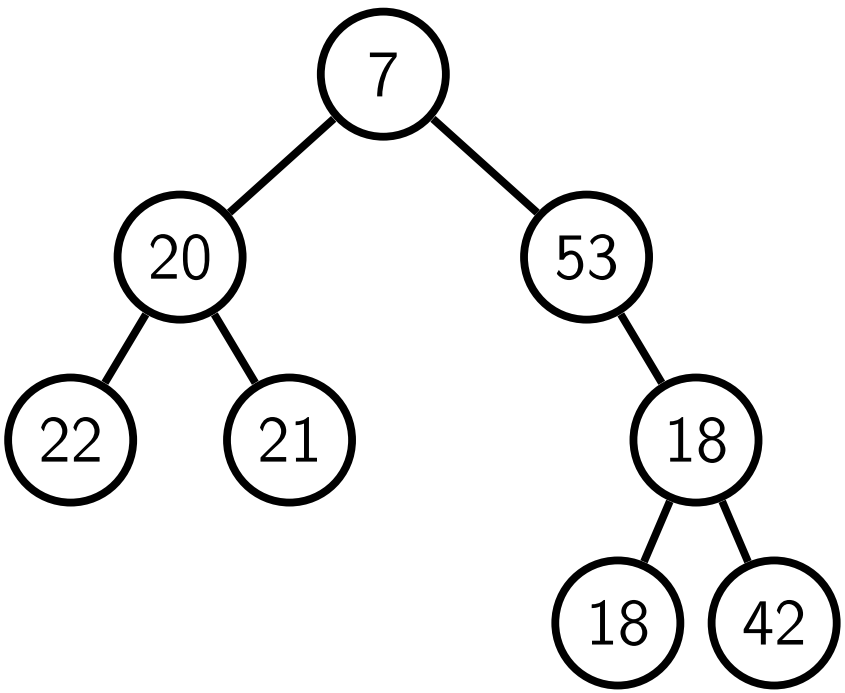
- заменим корень на любой лист

Извлечение минимума и просеивание вниз



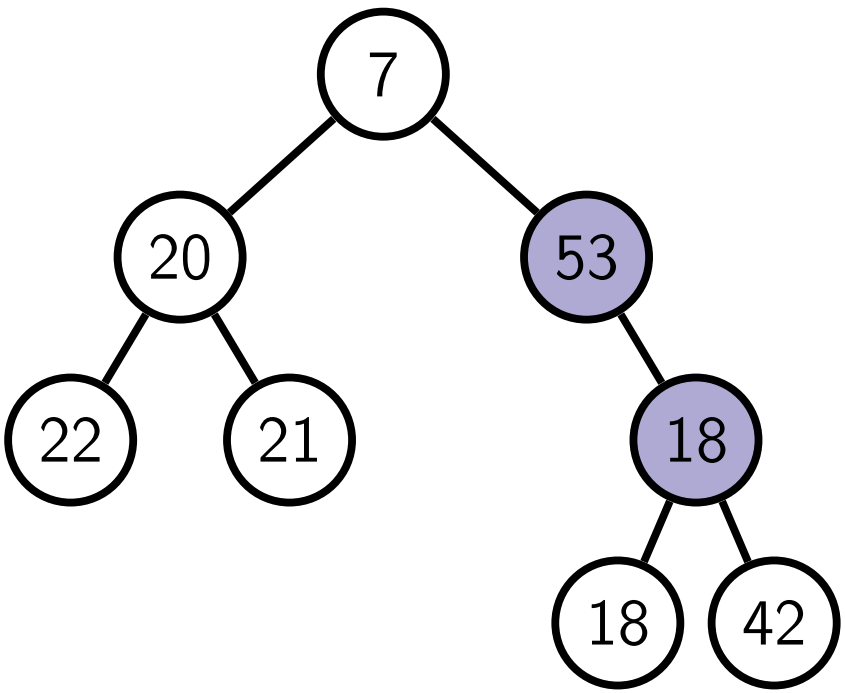
- заменим корень на любой лист

Извлечение минимума и просеивание вниз



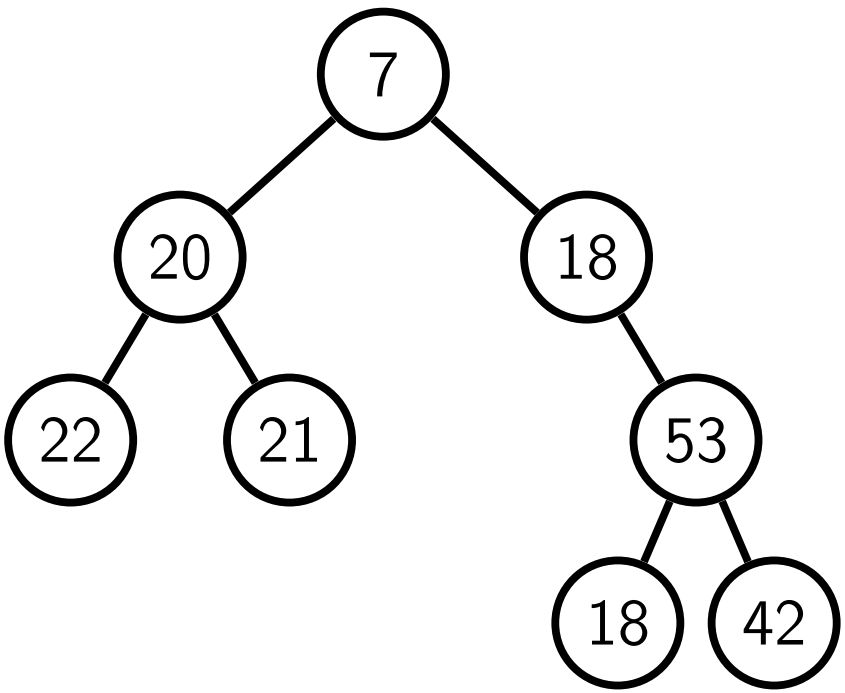
- заменим корень на любой лист
- обменяем проблемную вершину с меньшим из её детей, чтобы быть уверенными, что продолжать чинить кучу нужно только в одном из поддеревьев

Извлечение минимума и просеивание вниз



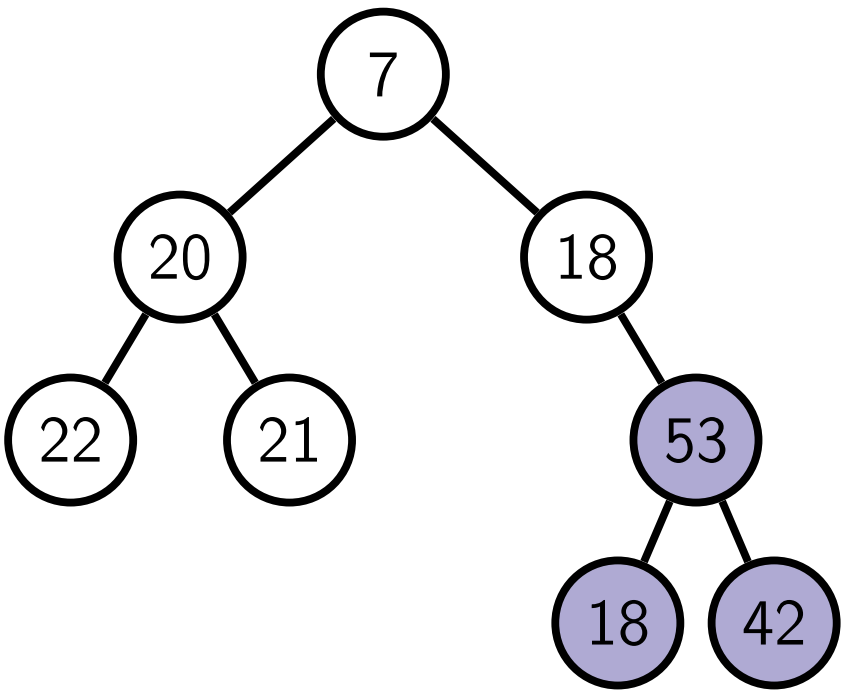
- заменим корень на любой лист
- обменяем проблемную вершину с меньшим из её детей, чтобы быть уверенными, что продолжать чинить кучу нужно только в одном из поддеревьев

Извлечение минимума и просеивание вниз



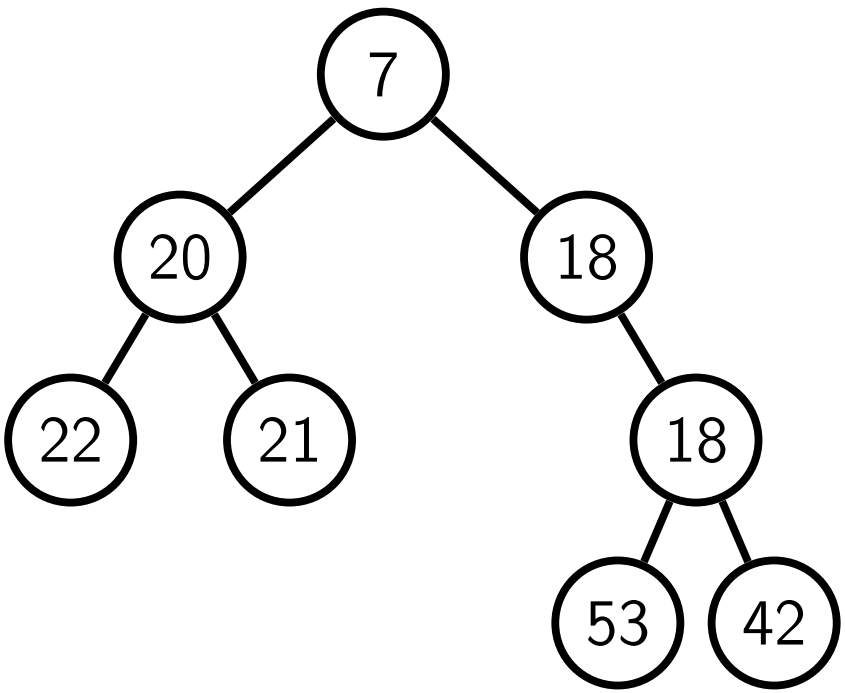
- заменим корень на любой лист
- обменяем проблемную вершину с меньшим из её детей, чтобы быть уверенными, что продолжать чинить кучу нужно только в одном из поддеревьев

Извлечение минимума и просеивание вниз



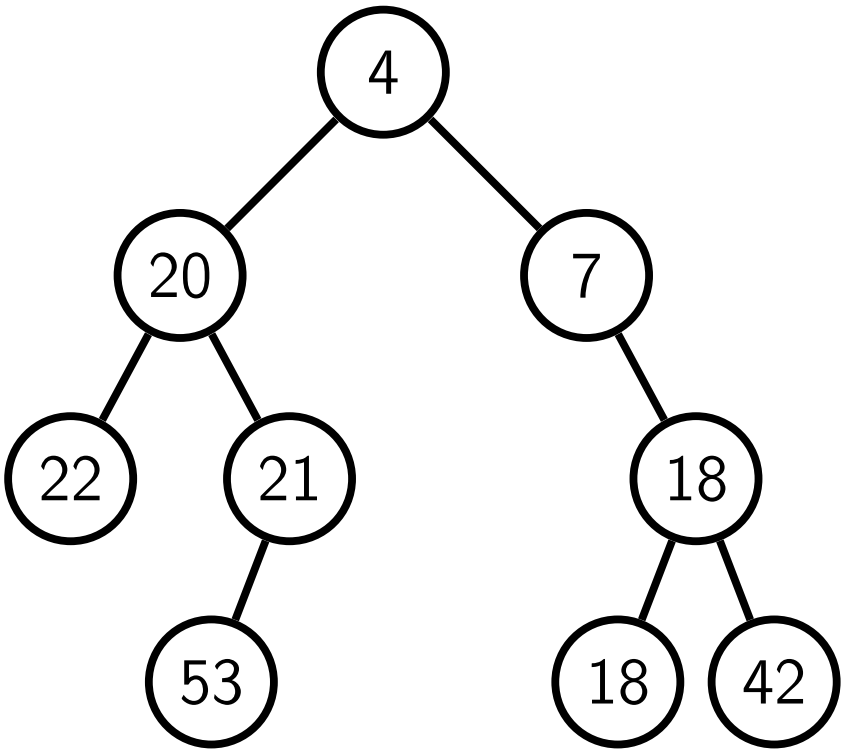
- заменим корень на любой лист
- обменяем проблемную вершину с меньшим из её детей, чтобы быть уверенными, что продолжать чинить кучу нужно только в одном из поддеревьев

Извлечение минимума и просеивание вниз

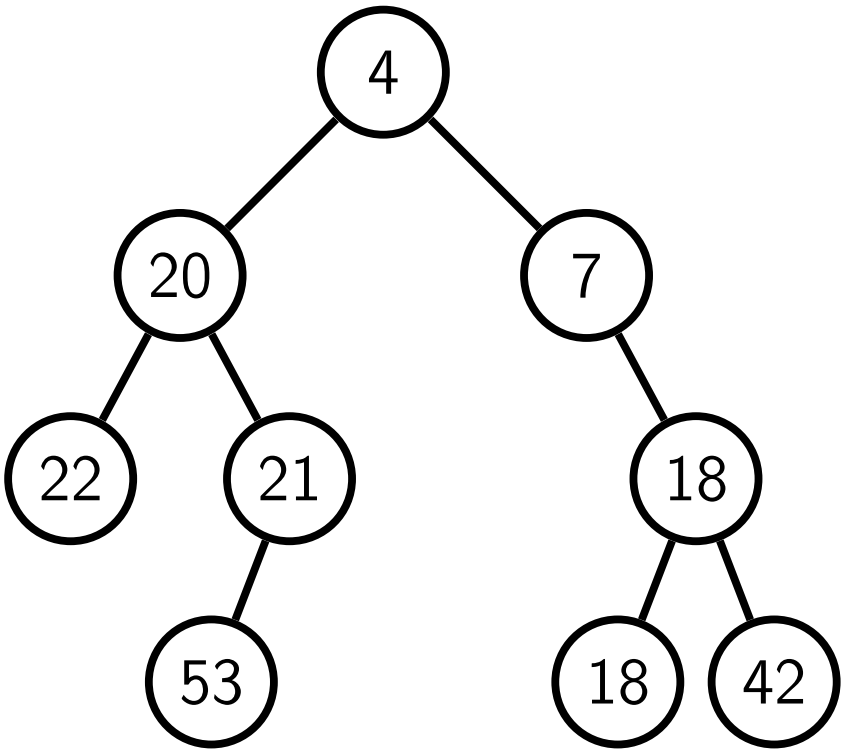


- заменим корень на любой лист
- обменяем проблемную вершину с меньшим из её детей, чтобы быть уверенными, что продолжать чинить кучу нужно только в одном из поддеревьев
- время работы: $O(h)$

Изменение приоритета

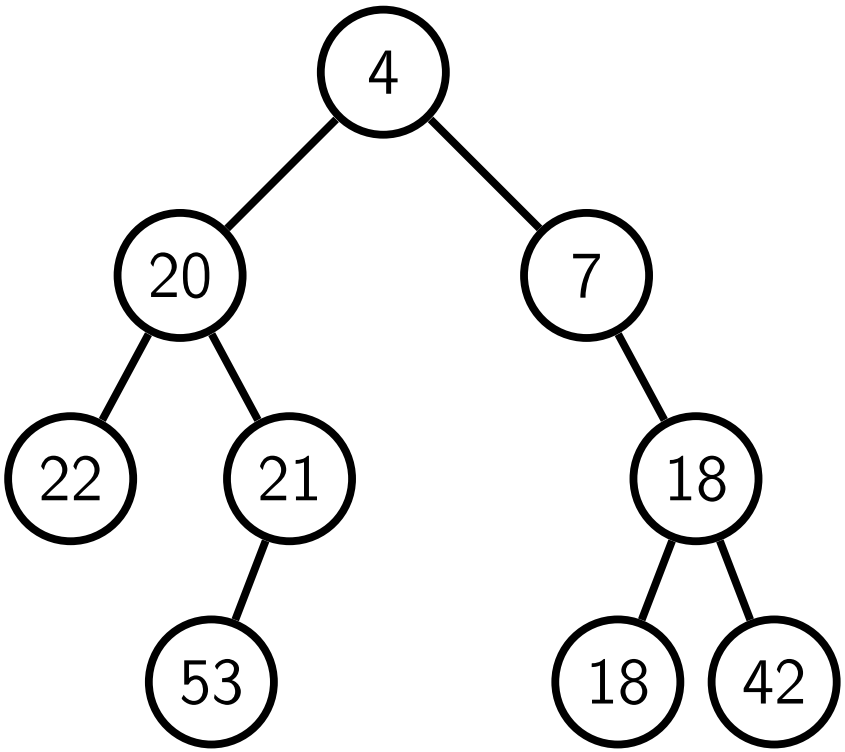


Изменение приоритета

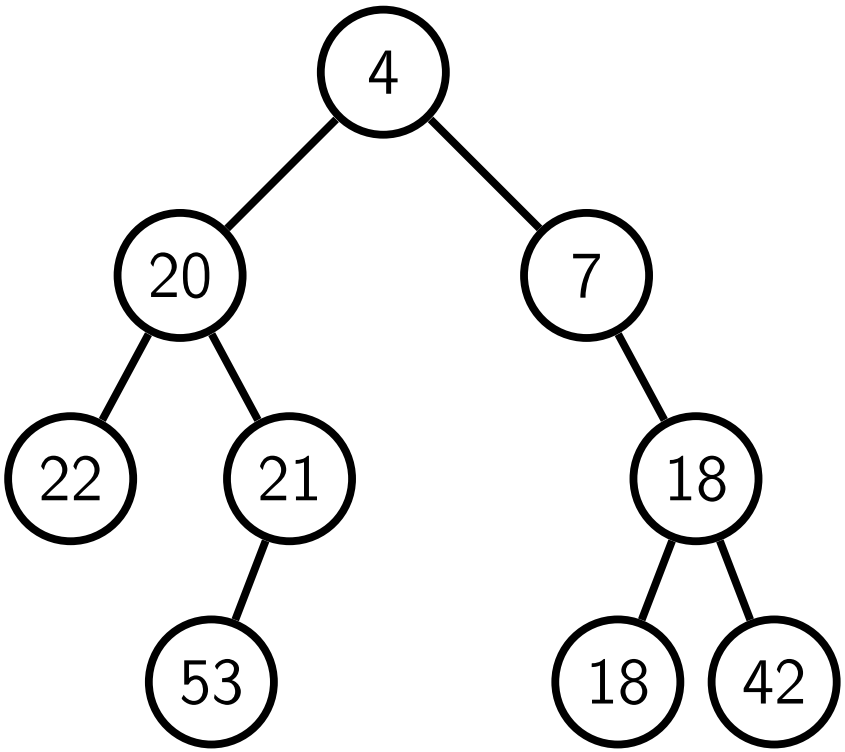


изменим приоритет и дадим элементу просеиться вниз или вверх

Удаление

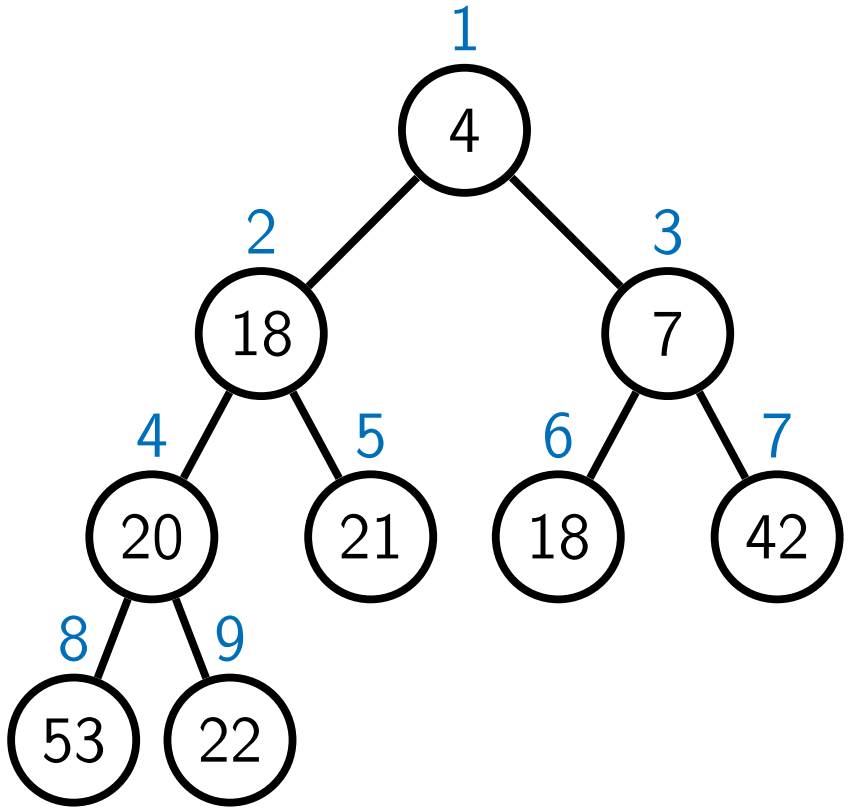


Удаление



изменим приоритет элемента на $-\infty$, просеим его вверх и извлечём минимум

Полное двоичное дерево и массив



1	2	3	4	5	6	7	8	9
4	18	7	20	21	18	42	53	22

Куча на массиве

- полное двоичное дерево: уровни заполняются слева направо; все уровни заполнены полностью, кроме, возможно, последнего

Куча на массиве

- полное двоичное дерево: уровни заполняются слева направо; все уровни заполнены полностью, кроме, возможно, последнего
- естественная нумерация вершин: сверху вниз, слева направо

Куча на массиве

- полное двоичное дерево: уровни заполняются слева направо; все уровни заполнены полностью, кроме, возможно, последнего
- естественная нумерация вершин: сверху вниз, слева направо
- при добавлении элемента подвешиваем лист на последний уровень; при удалении отрезаем самый последний лист

Куча на массиве

- полное двоичное дерево: уровни заполняются слева направо; все уровни заполнены полностью, кроме, возможно, последнего
- естественная нумерация вершин: сверху вниз, слева направо
- при добавлении элемента подвешиваем лист на последний уровень; при удалении отрезаем самый последний лист
- вершина i имеет родителя $\lfloor i/2 \rfloor$ и детей $2i$ и $2i + 1$ (при вычислении данных индексов нужно проверять, что они попадают в отрезок $[1, n]$)

Куча на массиве

- полное двоичное дерево: уровни заполняются слева направо; все уровни заполнены полностью, кроме, возможно, последнего
- естественная нумерация вершин: сверху вниз, слева направо
- при добавлении элемента подвешиваем лист на последний уровень; при удалении отрезаем самый последний лист
- вершина i имеет родителя $\lfloor i/2 \rfloor$ и детей $2i$ и $2i + 1$ (при вычислении данных индексов нужно проверять, что они попадают в отрезок $[1, n]$)
- не нужно хранить указатели на родителя и детей

Куча на массиве

- полное двоичное дерево: уровни заполняются слева направо; все уровни заполнены полностью, кроме, возможно, последнего
- естественная нумерация вершин: сверху вниз, слева направо
- при добавлении элемента подвешиваем лист на последний уровень; при удалении отрезаем самый последний лист
- вершина i имеет родителя $\lfloor i/2 \rfloor$ и детей $2i$ и $2i + 1$ (при вычислении данных индексов нужно проверять, что они попадают в отрезок $[1, n]$)
- не нужно хранить указатели на родителя и детей
- глубина кучи есть $O(\log n)$, поэтому все операции работают за время $O(\log n)$