Разработка и анализ алгоритмов

Лекция 7
Другие кучи

Сергей Леонидович Бабичев

Снова про кучи

- Бинарная куча:
  - проста в понимании;
  - проста в написании;
  - достаточно эффективна на *своих* операциях: $T_{getMin} = O(1)$, $T_{extractMin} = O(\log N)$, $T_{insert} = O(\log N)$, $T_{decreaseKey} = O(\log N)$.
  - А если требуется слияние двух бинарных куч?

Слияние бинарных куч A и B

- Наивный способ: извлекаем из кучи A вершину и вставляем её в B. Сложность $|A| \log |A| + |A| \log |A + B|$.

Слияние бинарных куч A и B

- Наивный способ: извлекаем из кучи A вершину и вставляем её в B. Сложность $|A| \log |A| + |A| \log |A + B|$.
- Более продвинутый способ: проходим по массиву в более короткой куче и вставляем элемент в более длинную. Сложность $|A| \log |A + B|$.

Слияние бинарных куч A и B

- Наивный способ: извлекаем из кучи A вершину и вставляем её в B. Сложность $|A| \log |A| + |A| \log |A + B|$.
- Более продвинутый способ: проходим по массиву в более короткой куче и вставляем элемент в более длинную. Сложность $|A| \log |A + B|$.
- Ещё более быстрый способ: формируем массив в $|A| + |B|$ элементов и создаём на нём кучу. Сложность $O(|A| + |B|)$.

Что осталось за кадром в прошлой лекции

- Мы использовали операцию `heapify` в пирамидальной сортировке в двух этапах:
  1. Построение бинарной кучи на массиве.

```
void sort_heap(T v[], size_t size) {
  for(ssize_t i = size/2-1; i >= 0; i--) {
    heapify(a, i, size);
  }
```

  2. Извлечение приоритетного элемента и отправка его на нужное место.

```
  while( size > 1 ) {
    size--;
    swap(a[0],a[size]);
    heapify(a, 0, size);
  }
 }
```

- Утверждается, что сложность первого этапа $O(N)$.

Сложность операции создания бинарной кучи из массива

- Почему цикл начинается с $size/2 - 1$?
- При адресации с 0 родитель с индексом l имеет детей с индексами $2l + 1$ и $2l + 2$.
- В куче из $size$ элементов, элемента с индексом $size/2 - 1$ имеет по крайней мере одного ребёнка. Элемент с индексом $size/2$ детей не имеет.
- $heapify$ — вариант $siftDown$.
- Пусть высота дерева равна h.
- Сначала ко всем элементам, расположенным над терминальными (множество S_1), применится $siftDown$. Элемент опустится не более, чем на 1 уровень вниз.
- Затем ко всем родительским элементам S_2, родительских для S_1, операция $siftDown$ опустит элементы не более, чем на 2 уровня вниз.

Сложность операции создания бинарной кучи из массива

- На шаге k элементы множества S_k опустятся не более, чем на k уровней.
- Общая сложность

$$T = \sum_{i=1}^{h} k|S_k| = \sum_{i=1}^{h} = k\left\lfloor\frac{N}{2^k}\right\rfloor = 1\left\lfloor\frac{N}{2^1}\right\rfloor + 2\left\lfloor\frac{N}{2^2}\right\rfloor + 3\left\lfloor\frac{N}{2^3}\right\rfloor + \cdots + h\left\lfloor\frac{N}{2^h}\right\rfloor$$

- Так как $k\left\lfloor\frac{N}{2^k}\right\rfloor \leqslant N\frac{h}{2^h}$, то

$$T \leqslant N \cdot \left(\frac{1}{2^1} + \frac{2}{2^2} + \cdots + \frac{h}{2^h}\right).$$

- Дифференцируем разложение геометрической прогрессии $1 + x + x^2 + \ldots$

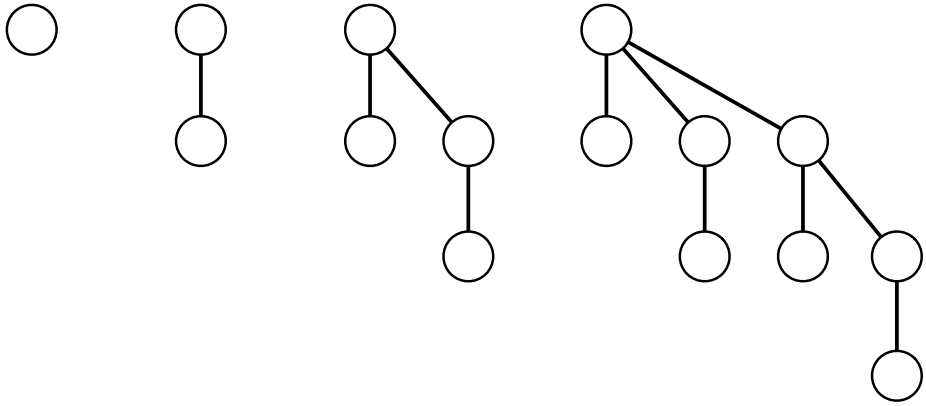
$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}.$$

- При $x = 1/2$ сумма есть 2.

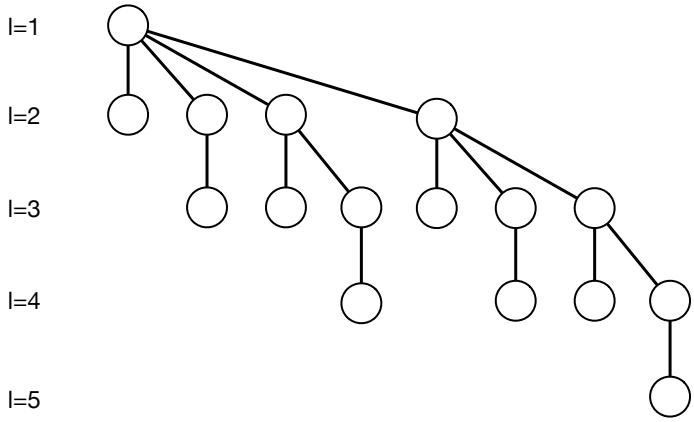
Биномиальные деревья и биномиальные кучи.

Definition (Биномиальное дерево)

Биномиальное дерево порядка 0 состоит из одного узла. Биномиальное дерево порядка n есть объединение двух биномиальных деревьев порядка $n-1$, при котором корень одного дерева есть ребёнок корня другого.



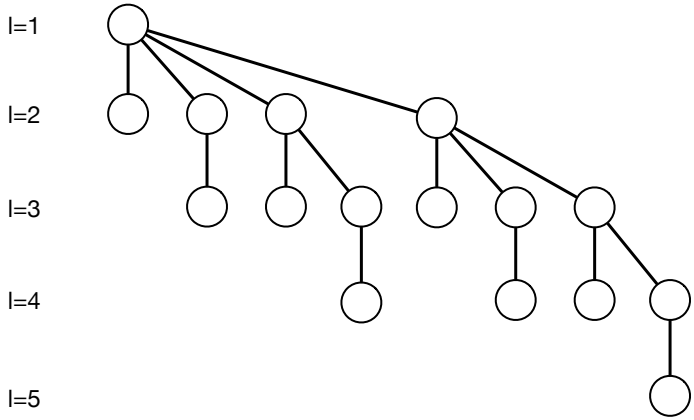
Свойства биномиального дерева



Lemma

Биномиальное дерево высоты K содержит ровно 2^K узлов.

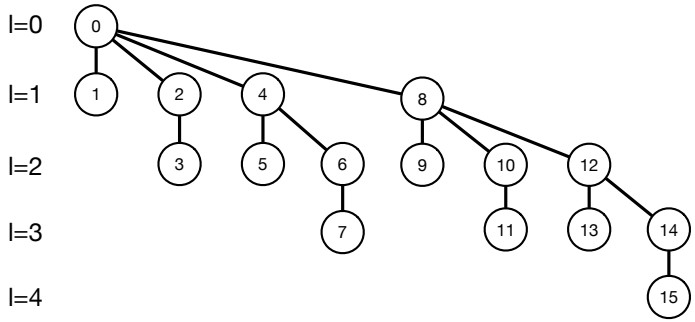
Свойства биномиального дерева



Lemma

В биномиальном дереве высоты K на уровне L располагается C_K^L узлов.

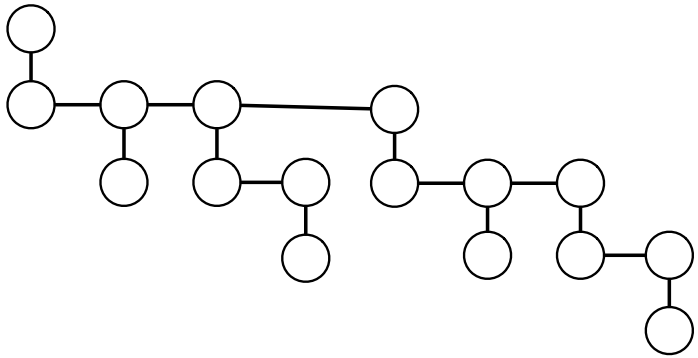
Хранение биномиального дерева степени K в виде массива



- Корень имеет K детей.
- Номера детей $2^0, 2^1, 2^2, \ldots$.
- Для узла с номером u дети имеют номера $u + 2^0, u + 2^1, \ldots$.
- Нет очевидного способа нахождения родителя.

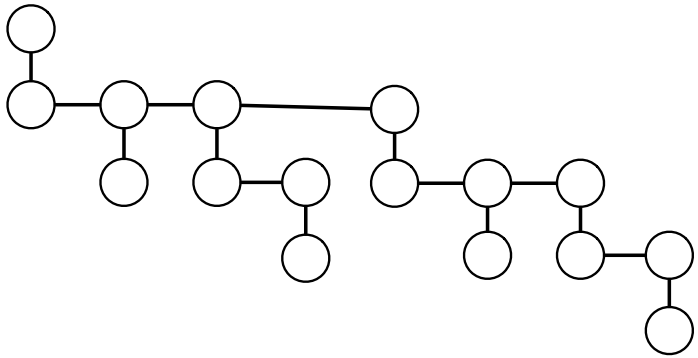
Хранение биномиального дерева с указателями

- Наивное хранение: переменное число указателей вниз.
- Более компактное хранение: хранится только родитель, один ребёнок, один брат.



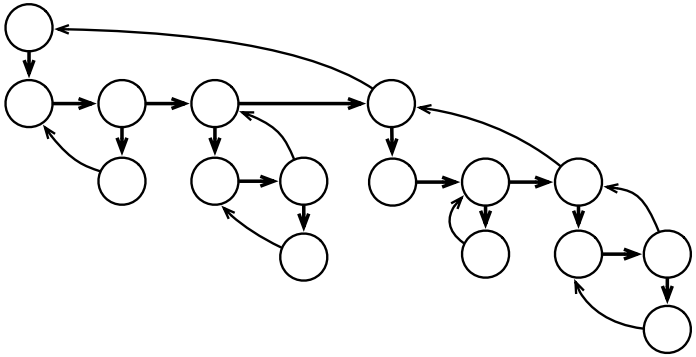
Хранение биномиального дерева с указателями

- Наивное хранение: переменное число указателей вниз.
- Более компактное хранение: хранится только родитель, один ребёнок, один брат.



Хранение биномиального дерева с указателями: компактная схема

- У самого правого ребёнка нет правого брата.
- Тогда в этом поле храним указатель на родителя.
- Обход всего дерева возможен.



Слияние биномиальных деревьев

- *Свойство кучи*: любой узел не меньше родителя.
- Количество элементов каждого биномиального дерева строго 2^K.
- Сливать можно только деревья одинакового размера.
- Для сохранения свойств кучи нужно корнем делать наименьший элемент.

Definition (Биномиальный бор)

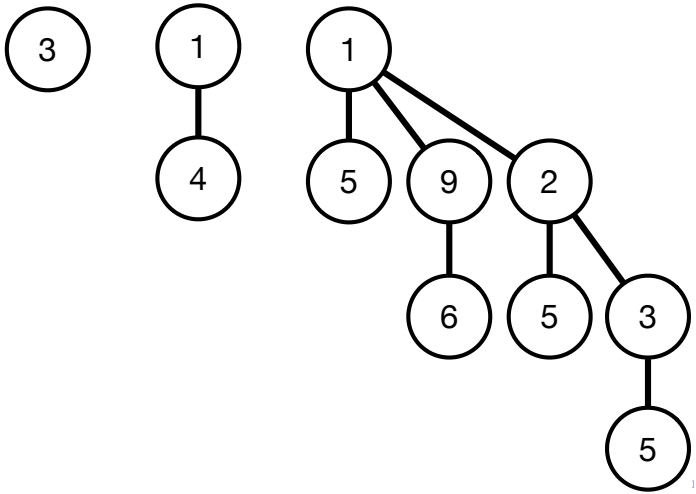
Биномиальный бор — множество биномиальных деревьев попарно различной высоты.

- Разбиение на деревья производится по разложению числа узлов бора по степеням двойки.
- Бор в 21 элемент представляется деревьями в 1, 4 и 16 элементов.
- $21 = 10101_2 = 2^0 + 2^2 + 2^4$.

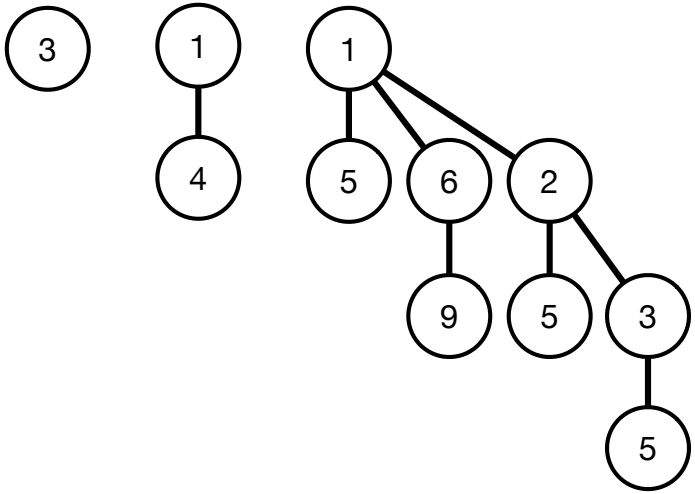
Definition (Биномиальная куча)

Биномиальная куча — биномиальный лес, удовлетворяющий *свойствам кучи*.

- Ключи $\{3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5\}$ разбиваются на три дерева $\{3\}$, $\{1, 4\}$, $\{1, 5, 9, 2, 6, 5, 3, 5\}$.



- Перестановкой элементов внутри дерева приводим к биномиальной куче.

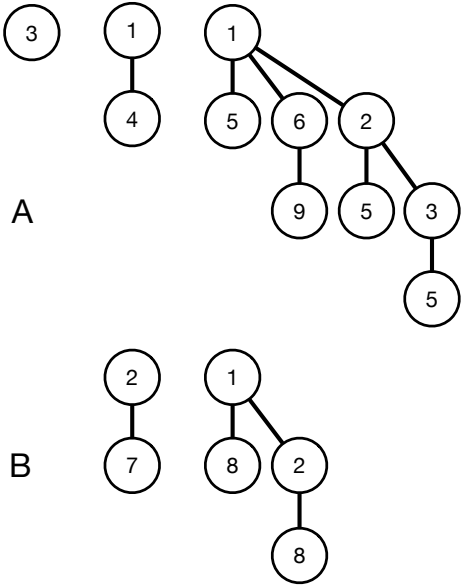


Операция *merge*

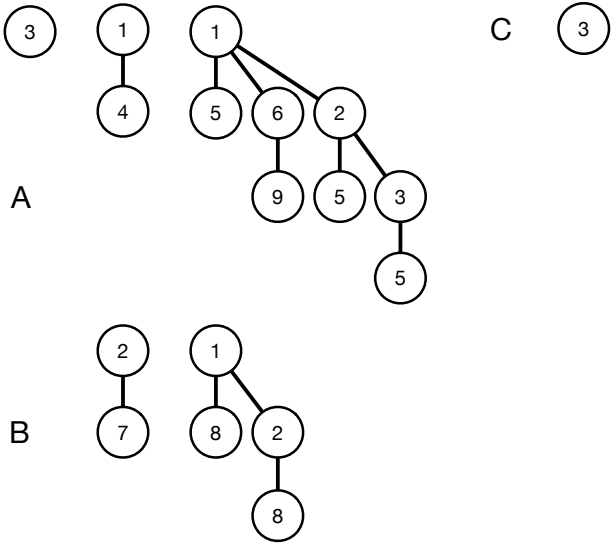
- Операция *merge* сливает кучи A и B разного размера в кучу C.

1. Установим текущий порядок дерева K в 0.
2. Если $K > K_A \wedge K > K_B$, то алгоритм закончен.
3. Если деревьев ранга K ни в A, ни в B не имеется, увеличим K на 1 и перейдём к п. 2.
4. Если имеется ровно одно дерево ранга K только в одном из A и B копируем его в C. Идём к п. 2.
5. . Если в A, B и C имеется ровно два дерева ранга K, то сливаем их в дерево ранга $K+1$ и переносим результат в C. Идём к п. 2.
6. . Если в A, B и C имеется ровно три дерева ранга K, сливаем B и C в одно ранга $K+1$ и переносим его в C. Идём к п. 2.

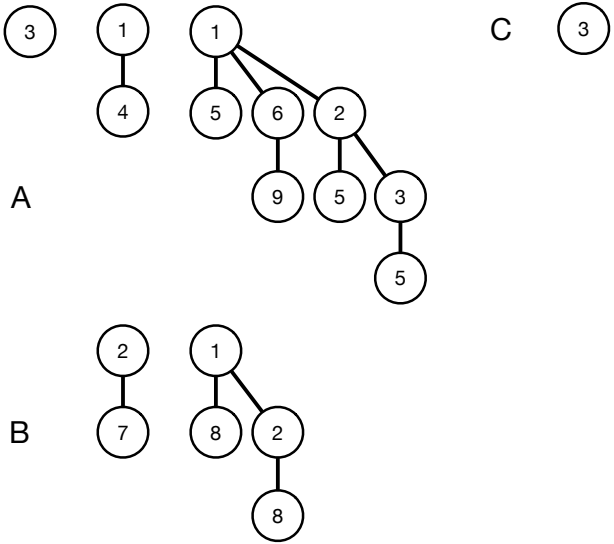
Слияние куч



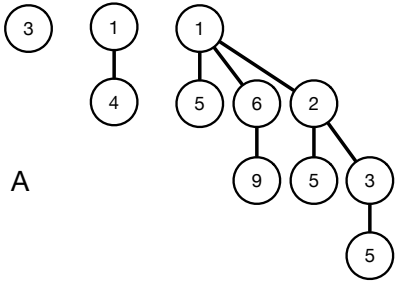
Слияние куч

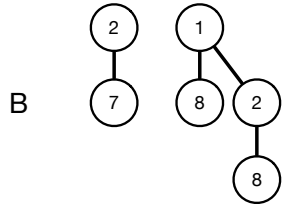


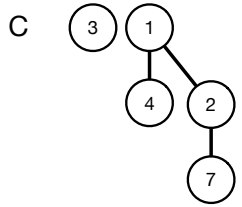
Слияние куч



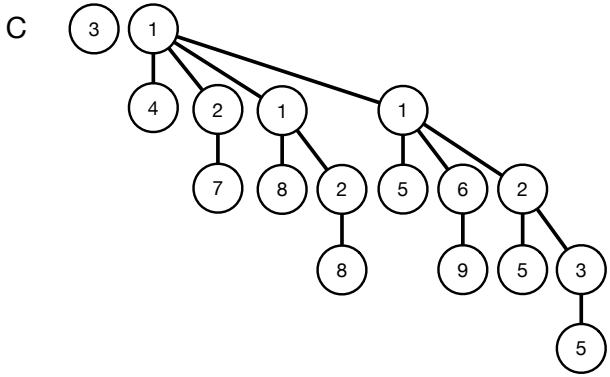
Слияние куч







Слияние куч



Операция *Слияние куч*

- Алгоритм и его сложность можно понять из рассмотрения двоичных представлений размеров куч.
- Складываем цифры справа налево, учитывая перенос.

$K_A = 1011_2$
$K_B = 110_2$

A		1	0	1	1
B			1	1	0
C	1	0	0	0	1

- $T = O(\log K_A + \log K_B) = O(\log N)$.

Операция *insert*

- Создаём кучу размером 1.
- Сливаем её с основной.
- $T = O(\log N)$

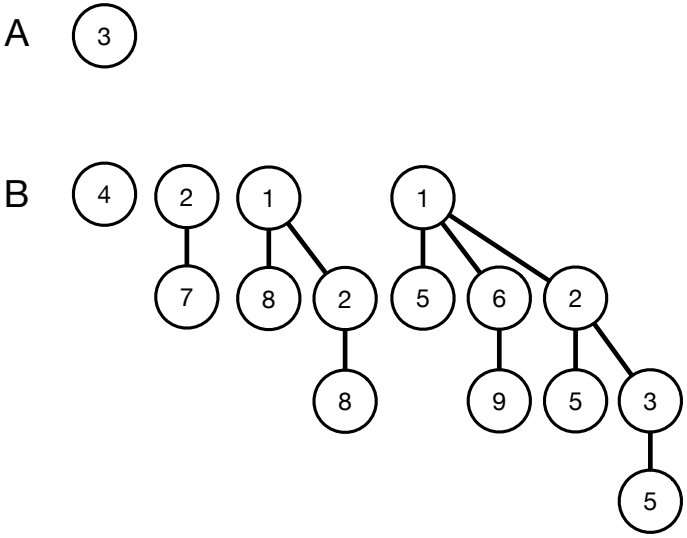
Операция *getMin*

- Поиск производится в корневых вершинах.
- $T = O(\log N)$.

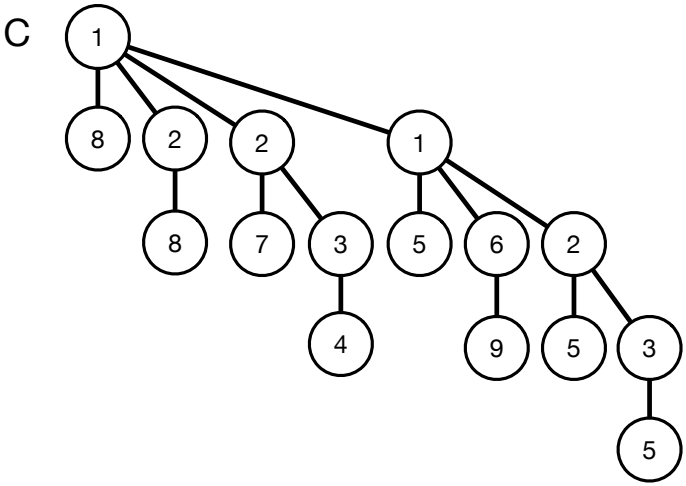
Операция *extractMin* для кучи A

1. Находим нужный корень по $getMin$.
2. Убираем это значение в списке корней кучи A.
3. Множество детей удалённой вершины образует биномиальную кучу B.
4. Сливаем кучу B с кучей A.

Операция *extractMin*: превращение дерева в кучу



Операция *extractMin*: превращение дерева в кучу: итог



Левацкая куча

Definition (Левацкое дерево)

Левацкое дерево — двоичное дерево, у которого ранг левого ребёнка не больше ранга правого ребёнка.

Definition (Ранг вершины левацкого дерева)

Ранг R вершины u левацкого дерева T определяется рекурсивно:

$$R(u) = \begin{cases} 0, \text{ если } u \equiv \emptyset; \\ \min(R(u_l), R(u_r)) + 1, \text{ иначе.} \end{cases}$$

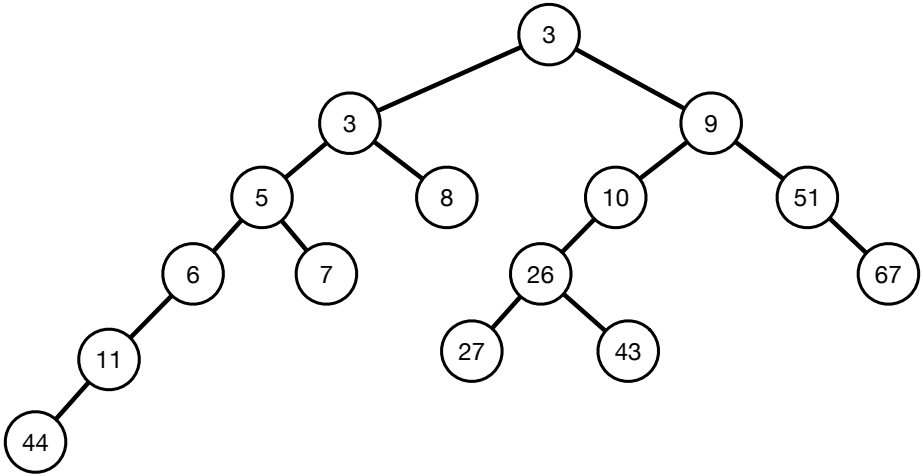
Definition (Левацкая куча)

Левацкая куча H — левацкое дерево, в котором выполнено свойство кучи, то есть родитель не превосходит детей.

Definition (Ранг левацкой кучи)

Ранг левацкой кучи есть ранг её корня.

Пример левацкой кучи



Левацкая куча

Lemma

Для любого узла $u \in H$ верно $R(u) = 1 + R(u_r)$.

Lemma

Количество узлов в левацкой куче ранга K не меньше 2^K.

Слияние левацких куч

Definition

Обозначим через $u_{A,B}$ поддерево левацкой кучи T с корнем в вершине u с левым ребёнком A и правым ребёнком B.

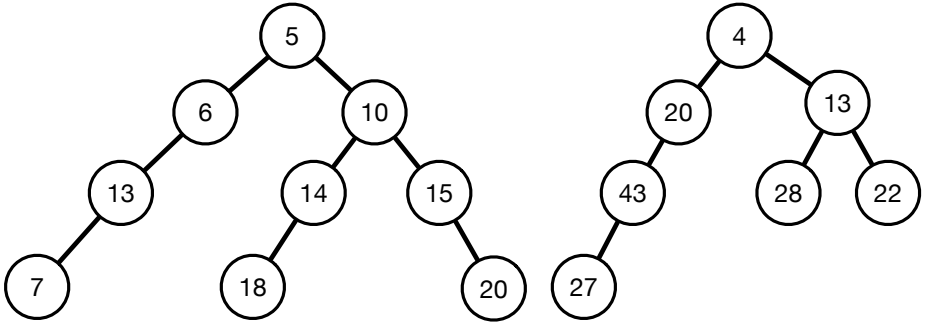
Lemma (Операция слияния двух левацких куч)

Операция $merge$ двух левацких куч может быть выражена как

$$merge(u_{A,B}, v_{C,D}) = \begin{cases} u_{A,B}, & \text{если } v_{C,D} \equiv \emptyset; \\ v_{C,D}, & \text{если } u_{A,B} \equiv \emptyset; \\ u_{A,merge(B,v_{C,D})}, & \text{если } u_{A,B} \leqslant v_{C,D}; \\ v_{C,merge(D,u_{A,B})}, & \text{если } u_{A,B} > v_{C,D}; \end{cases}$$

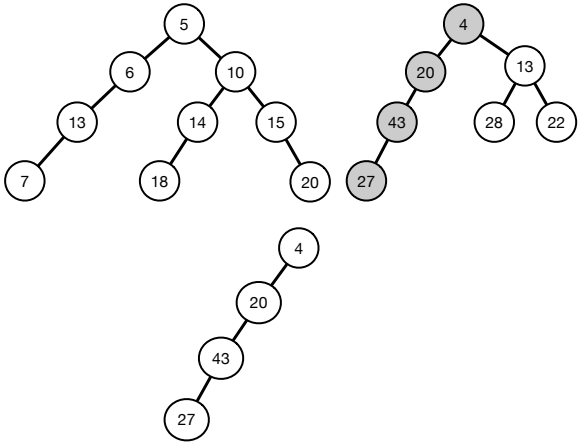
У полученного дерева свойства кучи сохраняется, свойство левацкости может быть нарушено.

Слияние левацких куч: пример



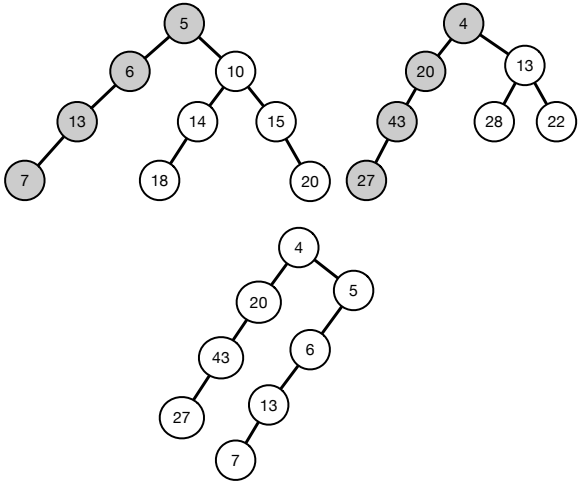
Слияние левацких куч: пример

Выбрана правая куча ($4 < 5$). Всё левое поддерево отправляется в результат. Рассмотренная часть помечается серым цветом.

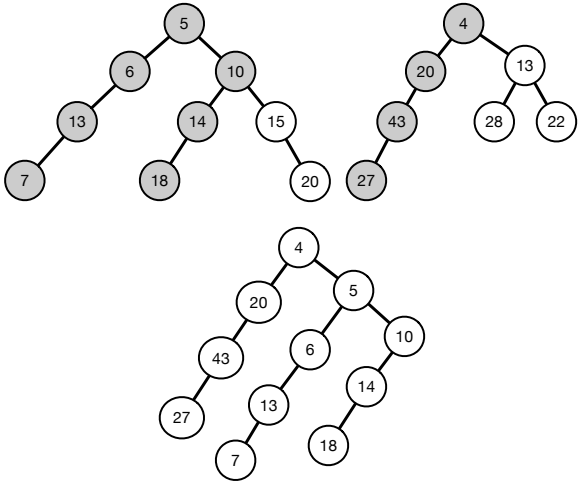


Слияние левацких куч: пример

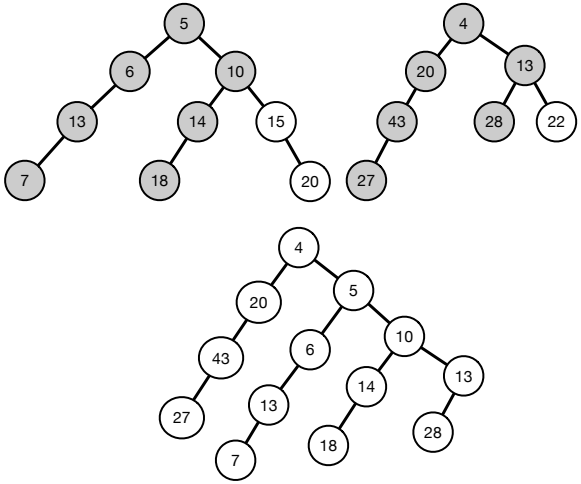
В рекурсии остались белые вершины на картинке. Выбрана левая куча с корнем 5 и она отправилась направо.



Слияние левацких куч: пример

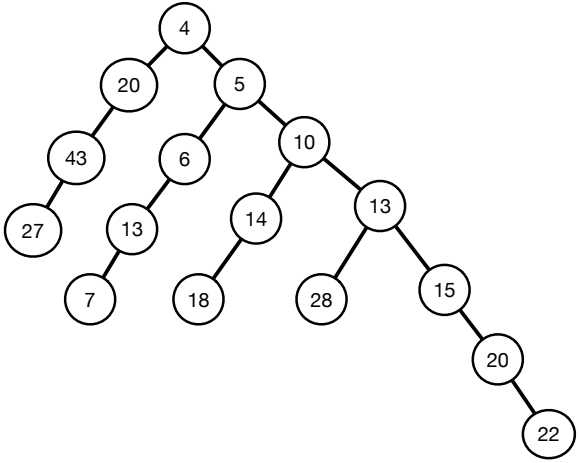


Слияние левацких куч: пример



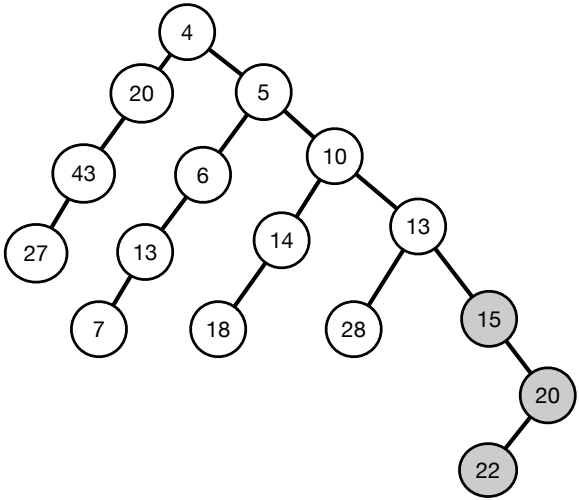
Слияние левацких куч: пример

Итоговый результат слияния: куча не левацкая. Возвращаемся по рекурсии по правой ветке для исправления левацкости. Узел 22 в порядке, поднимаемся на 20.



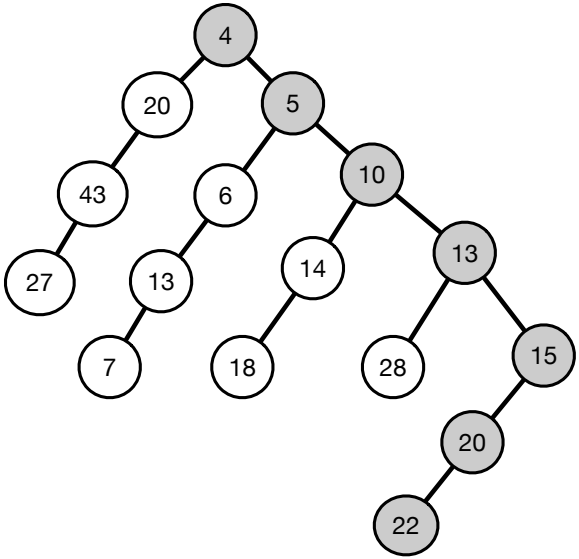
Слияние левацких куч: пример

Узел 20 не в порядке — меняем детей местами.



Слияние левацких куч: пример

Итог.



Сложности операций над левацкой кучей

- **Merge(A,B)** — $O(K_A + K_B)$.
- **Insert** — $O(K_A)$.
- **getKey** — $O(1)$.
- **extractKey** — какая?