Левосторонняя куча

Что такое куча?

- Куча-структура данных реализующая интерфейс очереди с приоритетом (Priority Queue).

- Приоритетная очередь — это абстрактная структура данных наподобие стека или очереди, где у каждого элемента есть приоритет. Элемент с более высоким приоритетом находится перед элементом с более низким приоритетом. Если у элементов одинаковые приоритеты, они располагаются в зависимости от своей позиции в очереди.

Применение

- Алгоритмы на графах : алгоритм Дейкстры, алгоритм Прима, поиск по первому наилучшему совпадению.
- Алгоритмы сжатия: алгоритм Хаффмана
- Поиск k-ого по порядку элемента

Сравнение куч

Операции	Двоичная	Биномиальная	**Левосторонняя**	Фибоначчиева	Бродала-Окасаки
find-min	O(1)	O(log n)	O(1)	O(1)	O(1)
delete-min	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)
insert	O(log n)	O(log n)	O(log n)	O(1)	O(1)
merge	O(n)	O(log n)	O(log n)	O(1)	O(1)

Проблема

Часто необходимо слить две кучи в одну:

- merge (A, B): вернуть новую кучу с ключами из A и B, уничтожив кучи A и B.
- В бинарной куче операция merge достаточно дорогая.

Как можно ускорить слияние?

- Использовать несбалансированное дерево.

Определения

- Левостороннее дерево — бинарное дерево, где ранг левого потомка всегда не меньше ранга правого.

  Это дерево было изобретено Кларком Алланом Крейном.

- Левосторонняя куча (англ. leftist heap) — левостороннее дерево с соблюдением порядка кучи.

Левосторонняя куча

Идея ускорения слияния:

- Сосредоточить все работы по изменению кучи в одной небольшой части кучи

Основные идеи левосторонней кучи:

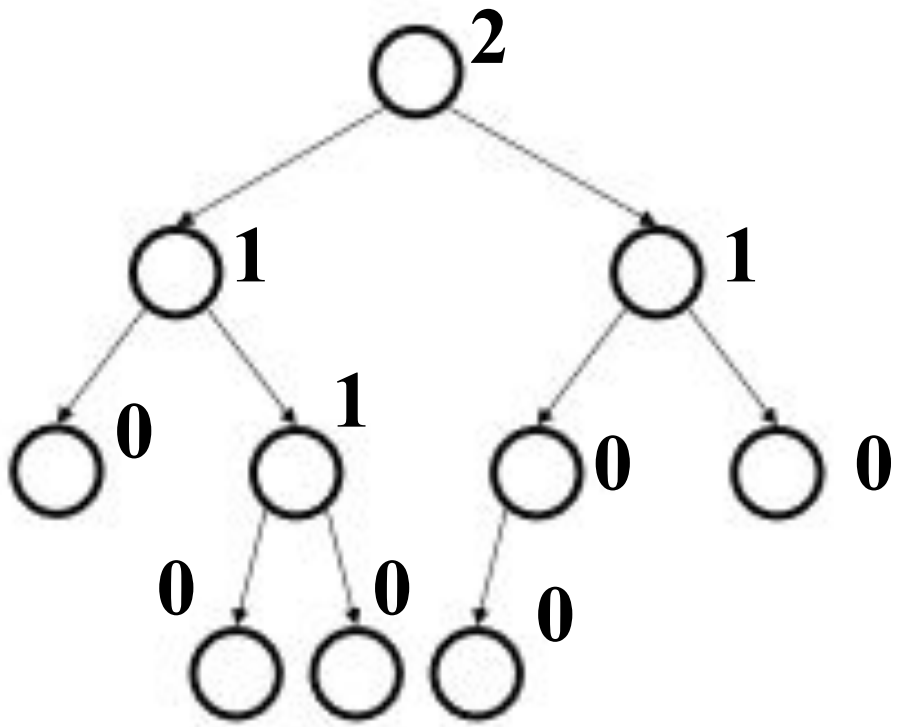
- Бинарные деревья
- Большинство узлов находятся слева
- Вся работа по слиянию происходит справа

Определение - Ранг узла

- Неполный узел — узел, у которого < 2 непосредственных потомков.

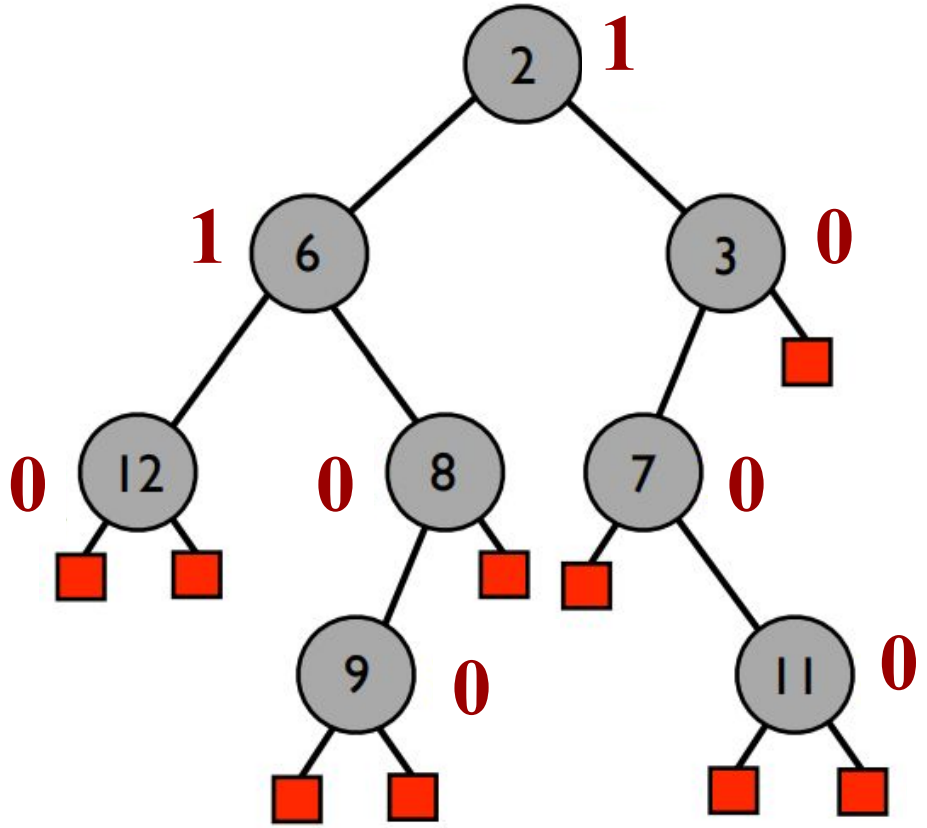
- Ранг узла(*null path length - npl*) — расстояние (число ребер) от него до ближайшего неполного потомка (+1)

  - *npl(null)=-1*
  - *npl(leaf)=0*
  - *npl(single-child node)=0*



Определение - Ранг узла

$$\mathrm{npl}(u) = \begin{cases} 0 & \text{Если u — неполный узел} \\ 1 + \min\{\mathrm{npl}(\mathrm{left}(u)), \mathrm{npl}(\mathrm{right}(u))\} \end{cases}$$



Свойства левосторонней кучи

- Требование к каждой вершине

➢ Значения в узлах-родителях ≤ значений в узлах-детях (для min heap)

➢ Зачем? Минимальный элемент всегда в корне (как в бинарной куче)

- Требование к структуре

➢ Для каждого узла x: $npl(left(x)) \geq npl(right(x))$

➢ Зачем? Левое поддерево всегда будет больше

Короче ли?

Утверждение: правый путь (путь от корня до самого правого неполного узла) такой же короткий, как и любой другой в дереве.

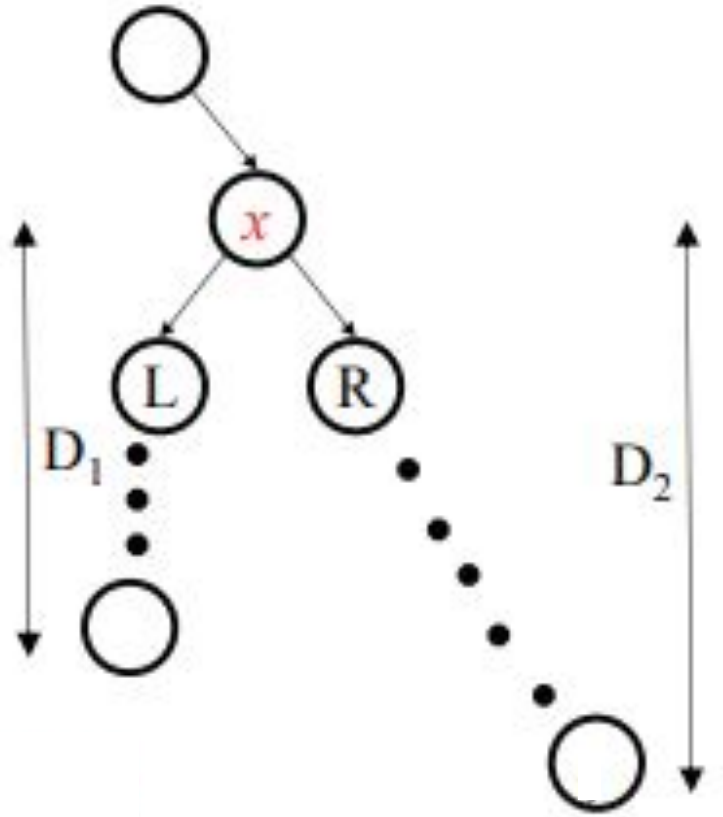
Доказательство: (Противоречие)

Выберем более короткий путь: **D1 <D2**

Скажем, что **D1** отклоняется от правого пути **D2** в узле **x**:

npl (L) ≤ D1-1 из-за пути длины **D1-1** к неполному узлу

npl (R) ≥ D2-1, потому что каждый узел на правом пути является левым ребенком.



Свойство левосторонней кучи для узла x нарушено!

Короче ли?

Утверждение: если правильный путь имеет **r** узлов, то дерево имеет как минимум 2^r-1 узлов.

Доказательство: (по индукции)
Базовый случай: r = 1. Дерево имеет как минимум $2^1-1 = 1$ узел
Индуктивный шаг: предположим,что утверждение верно для **r-1**.
Докажем для дерева с правым путем хотя бы **r**:
1. Правое поддерево: правильный путь узлов **r-1**
$\Rightarrow 2^{r-1}-1$ правых поддеревьев узлов (по индукции)
2. Левое поддерево: также правый путь длиной не менее **r-1** (пред. слайд)
$\Rightarrow 2^{r-1}-1$ левых узлов поддерева (по индукции)

$\Rightarrow$ **Общий размер дерева:** $(2^{r-1}-1) + (2^{r-1}-1) + 1 = 2^{r-1}$

Зачем все это?

Данные требования гарантируют, что:

- Правое поддерево действительно короче
- Левостороннее дерево из **N** узлов имеет правый путь не более $\log_2(N+1)$ узлов

Вспомним основную идею: Сосредоточить все работы по изменению кучи в одной небольшой части кучи (т.е в правом поддереве)

Слияние двух куч (операция merge)

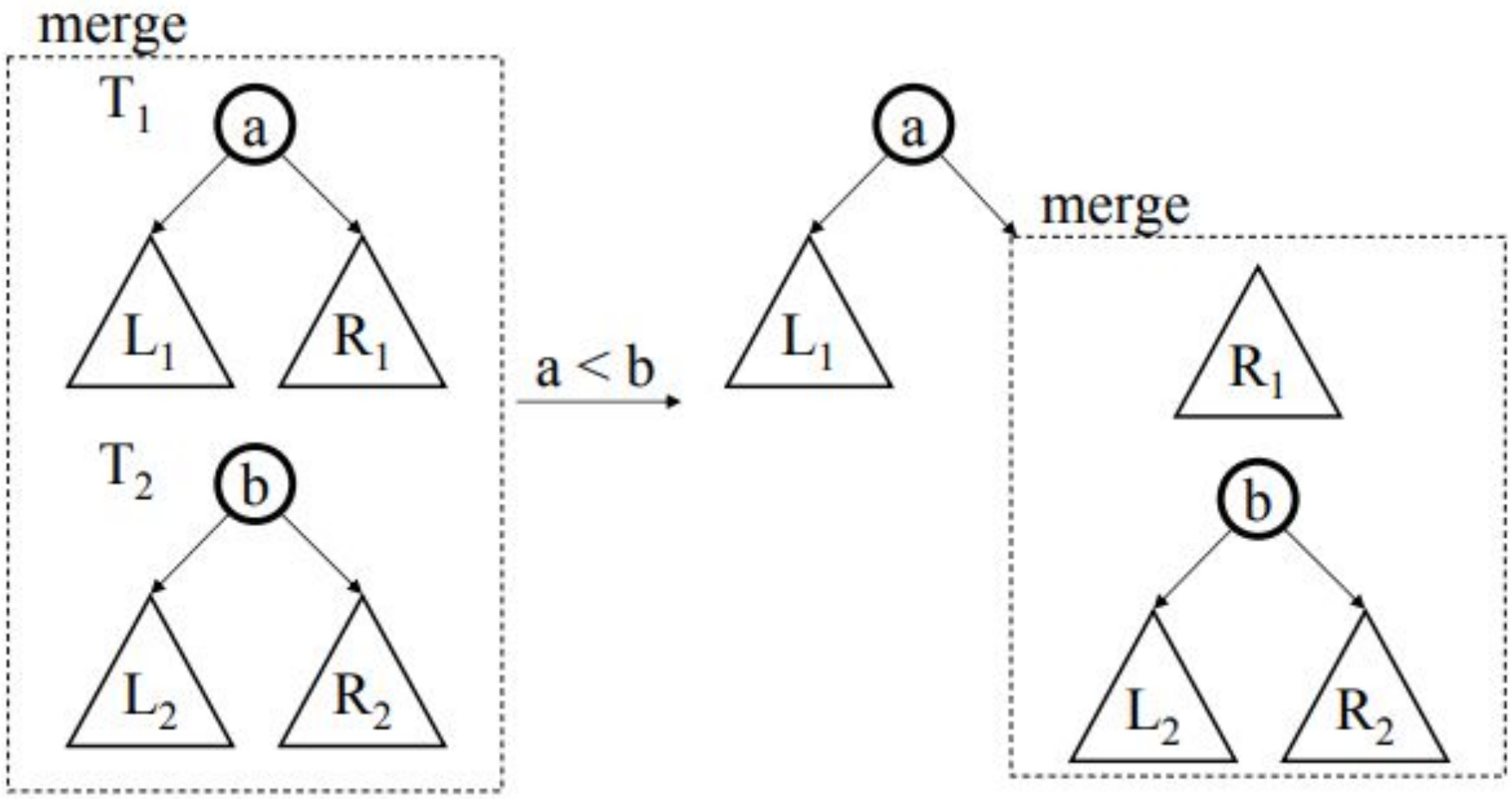
Основная идея:

- Сделать корень с меньшим значением корнем новой кучи
- *Левое поддерево нового корня оставить, а правое рекурсивно* слить с правым деревом
- Перед возвращением из рекурсии:
  - Обновить *npl* нового корня.
  - При необходимости поменять местами левое и правое поддеревья этого корня, чтобы сохранить левостороннее свойство результирующего дерева.

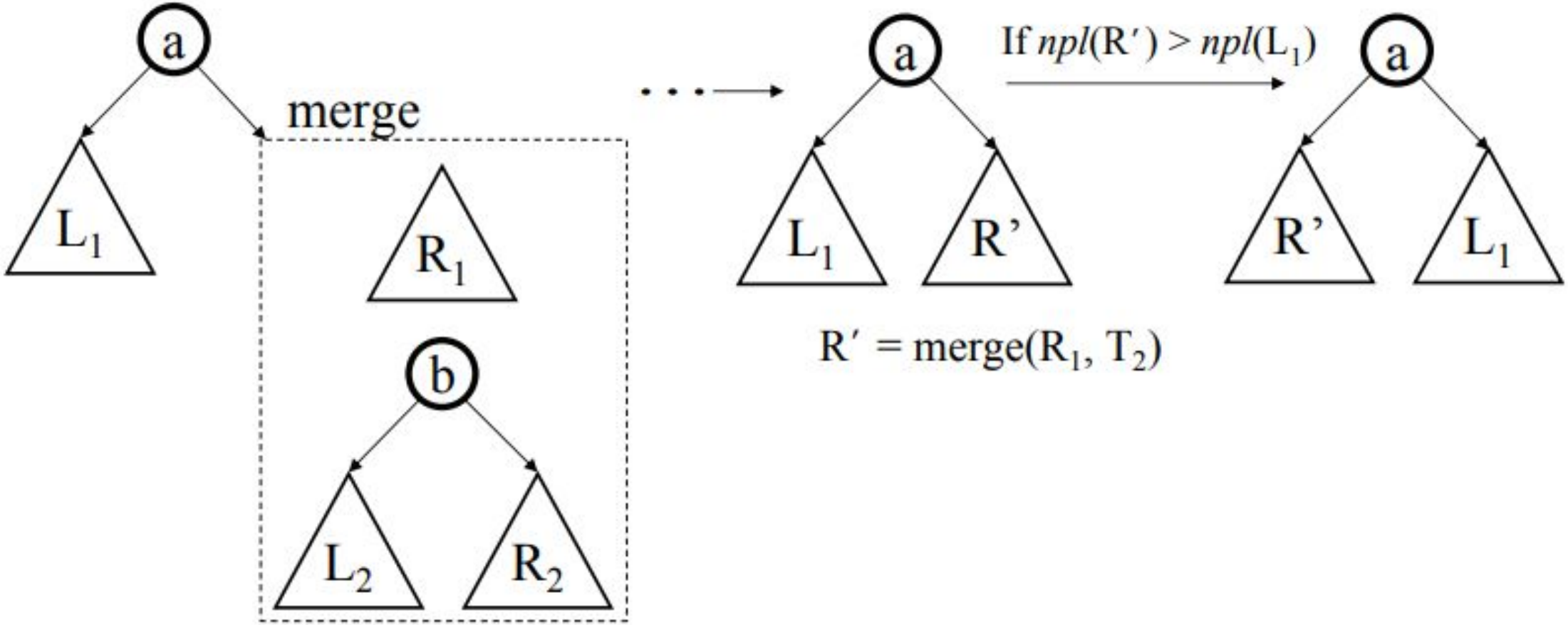
Специальный случай: merge(T,null) = merge(null,T) = T

Merge

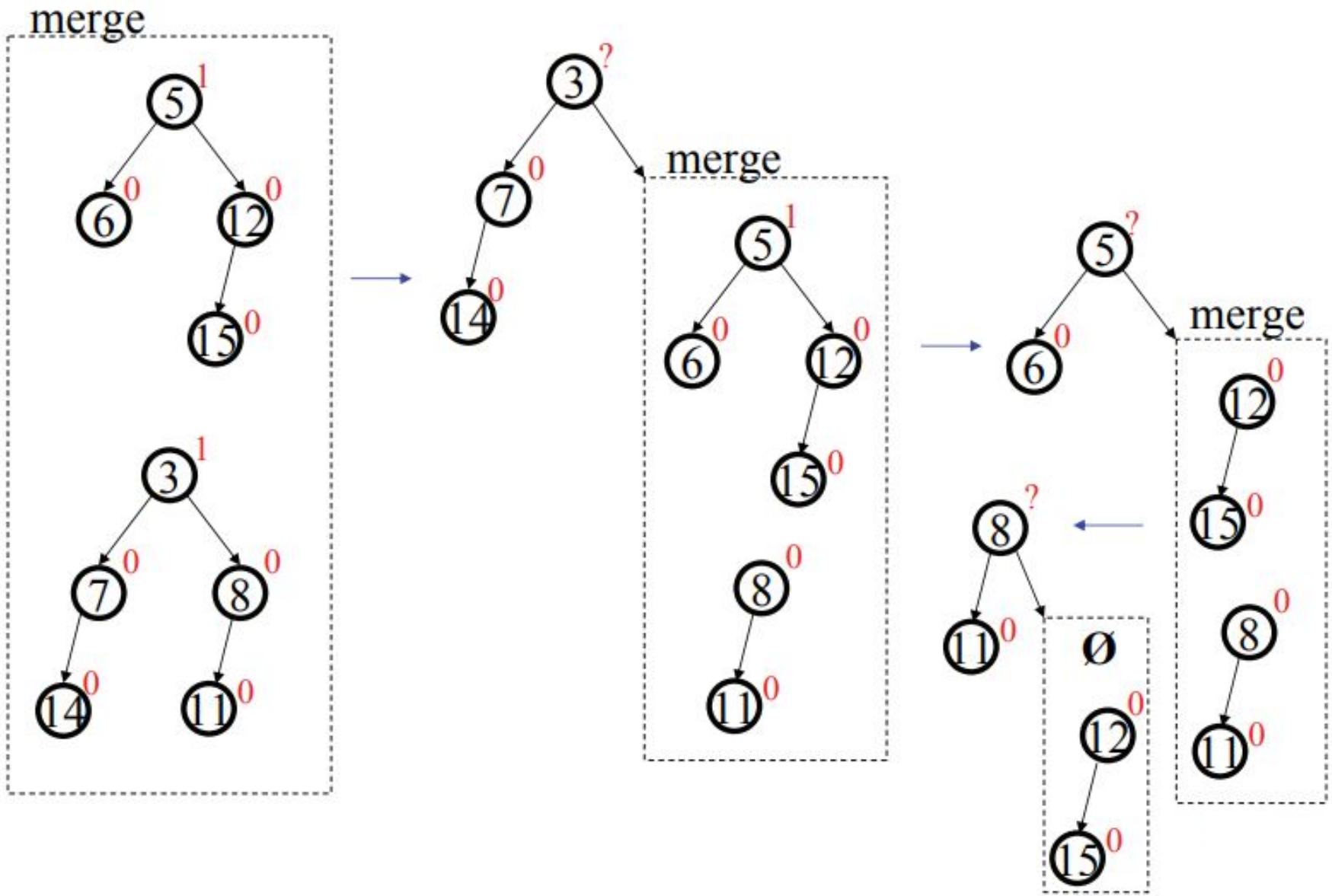
Рекурсивные вызовы merge(T_1, T_2) возвращают новую левостороннюю кучу, содержащую все элементы двух куч T_1 и T_2



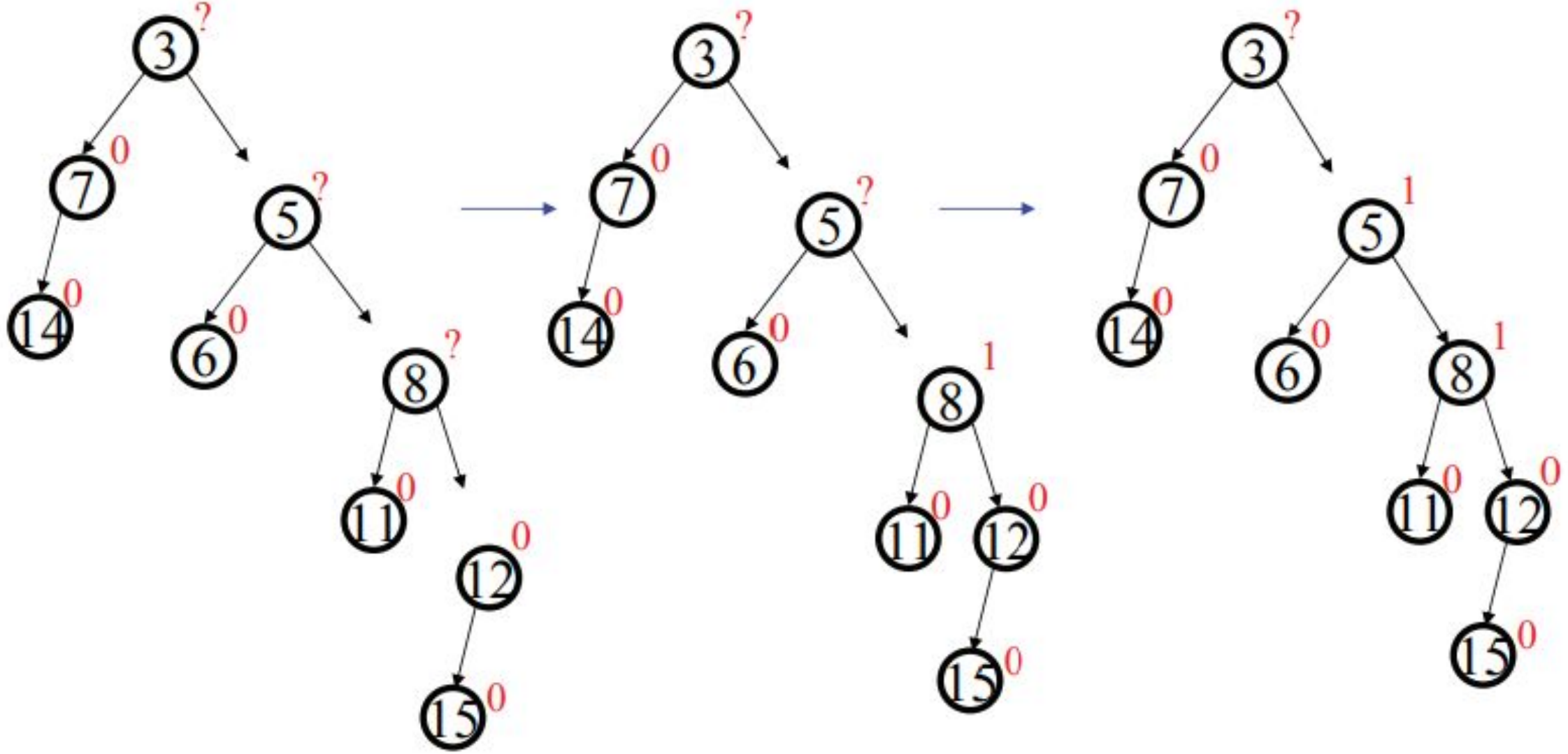
Merge-продолжение



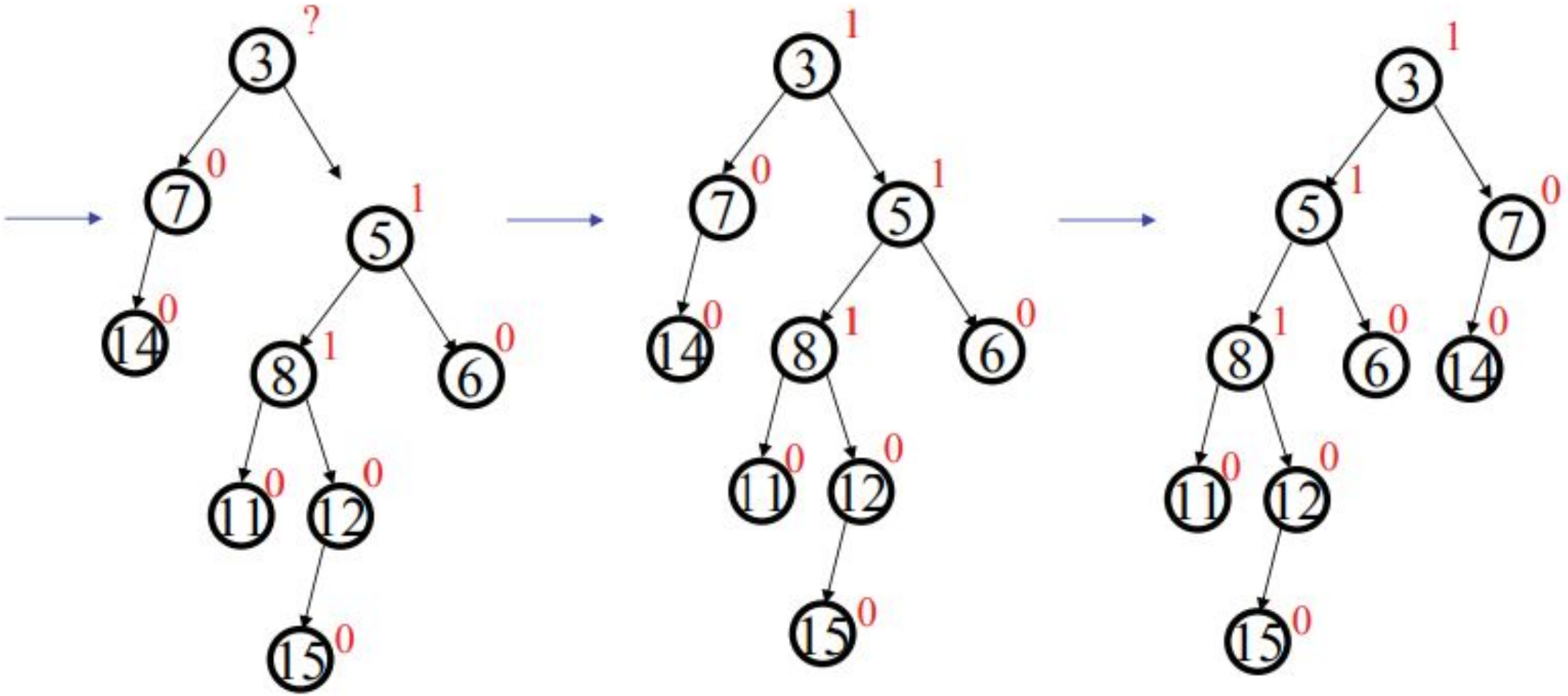
Merge - пример



Merge - восстановление свойств



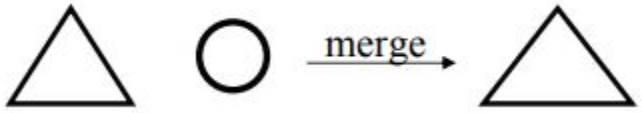
Merge - восстановление свойств



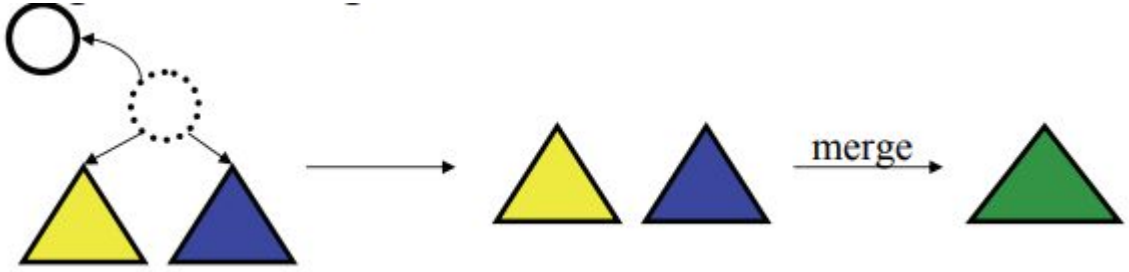
Операции левосторонней кучи:

- **merge**: объединить два дерева, общий размер которых n.
  - Асимптотика: *O(log n)* (Каждый раз идем только в правое поддерево. То есть не более логарифма вызовов.)

- **insert:** добавить новый элемент в кучу размера n.
  - притворный узел - левосторонняя куча размера 1
  - вставить, объединив оригинальную кучу с кучей из одного элемента
  - Асимптотика: *O(log n)* (Т.к на основе **merge**)



- **extractMin:** извлечь минимальный элемент из кучи размера n.
  - удалить и вернуть корень
  - объединить левое и правое поддеревья
  - Асимптотика: *O(log n)* (Т.к на основе **merge**)



Построение кучи за O(n)

- Храним список левосторонних куч
- Пока их количество больше 1:

  1) из начала списка достаём две кучи

  2) сливаем эти кучи

  3) и результат кладём в конец списка

Доказательство асимптотики

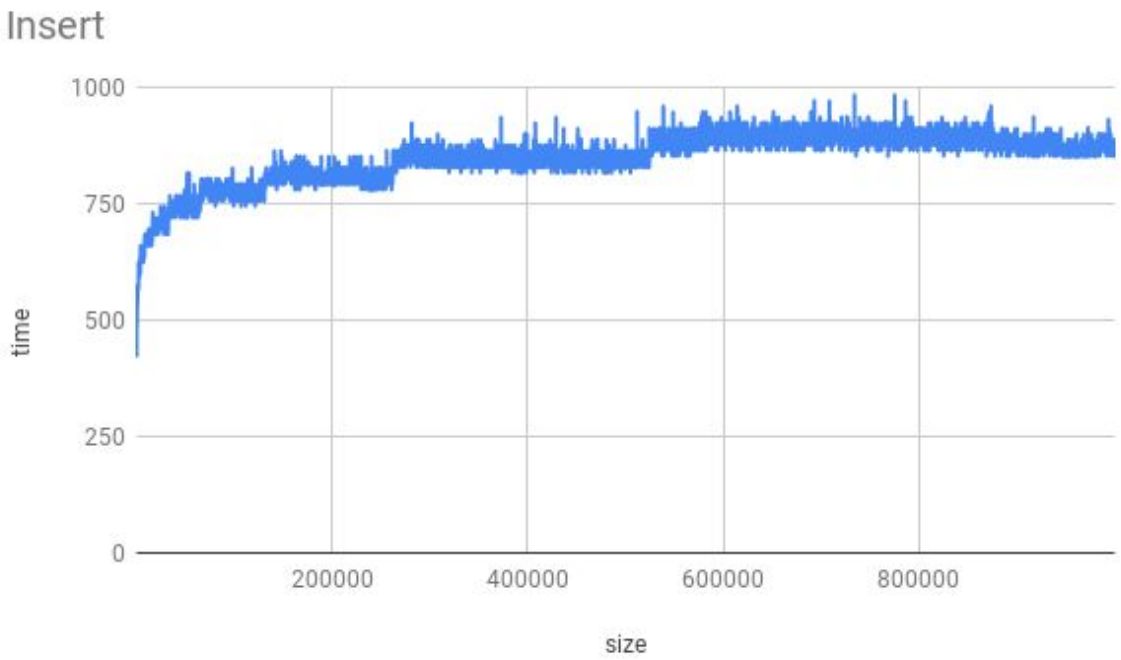
На нулевом шаге — **n** куч из одного элемента.
На каждом шаге количество куч уменьшается вдвое ,а число вершин в куче увеличивается вдвое.
На i-ом шаге в списке остались кучи размера $\mathbf{2^i}$.

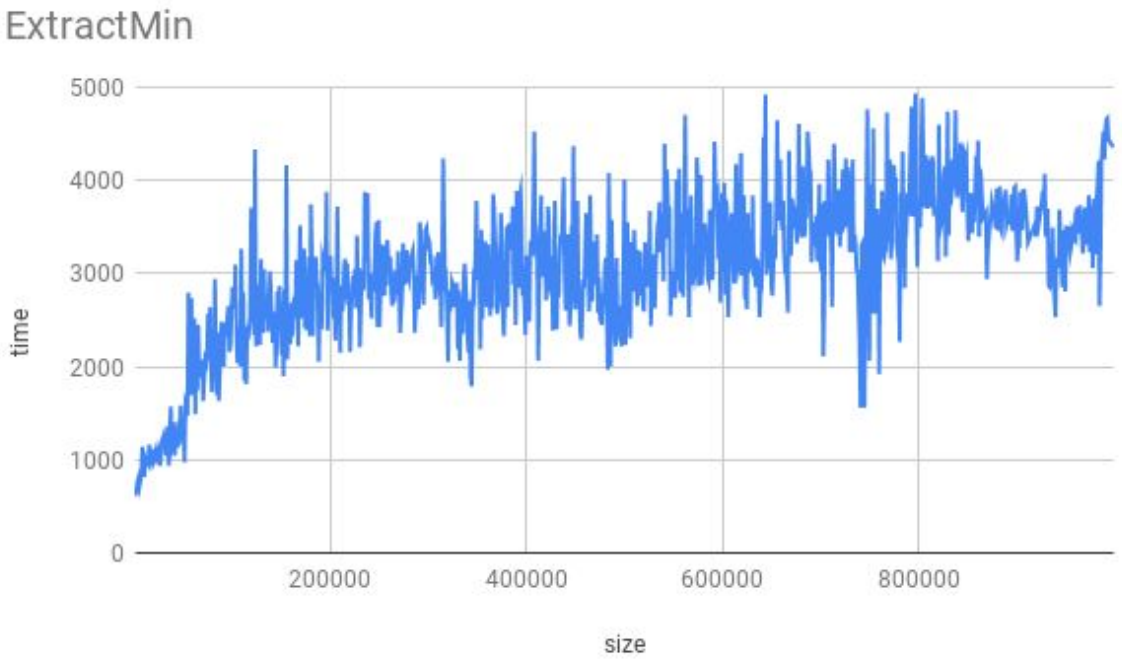
Слияние двух куч из $\mathbf{n_i}$ элементов — это $\mathbf{O(\log n_i)}$
Поэтому построение будет выполняться за:

$$\sum_{i=1}^{\lceil \log n \rceil} \frac{n \cdot \log n_i}{2^i} = n \cdot \sum_{i=1}^{\lceil \log n \rceil} \frac{\log 2^i}{2^i} = n \cdot \sum_{i=1}^{\lceil \log n \rceil} \frac{i}{2^i}$$

Сумма ряда равна **2**.
Поэтому построение кучи произойдёт за **O(n)**.

Замеры времени работы





Резюме

Плюсы:

- Основная операция merge
- Merge за O(log n)
- Простая реализация
- Нигде не делается уничтожающих присваиваний. Не создается новых узлов в merge. А значит, кучу можно легко сделать персистентной.

Минусы:

- Не в виде массива
- Дополнительное поле npl
- Можно быстрее (:

Источники

- Левосторонняя куча - викиконспекты ИТМО
- Приоритетные очереди - викиконспекты ИТМО
- Применение куч - GeekForGeeks
- CSC 378: Data Structures and Algorithm Analysis
- Functional Heap - Leftist Tree
- MAW Chapter 6 summary
- Нормально анимированная визуализация (:

Спасибо за внимание

Вопросы?