Декартово дерево

Булгаков Илья, Гусев Илья

Московский физико-технический институт

Москва, 2021

Содержание

Деревья поиска

Виды самобалансирующихся деревьев

- AVL-дерево (рассматривали прошлый раз)
- Splay-дерево
- Красно-черное дерево

Другие деревья

- Декартово дерево (не является самобалансирующимся в обычном смысле)

Декартово дерево

Общее описание

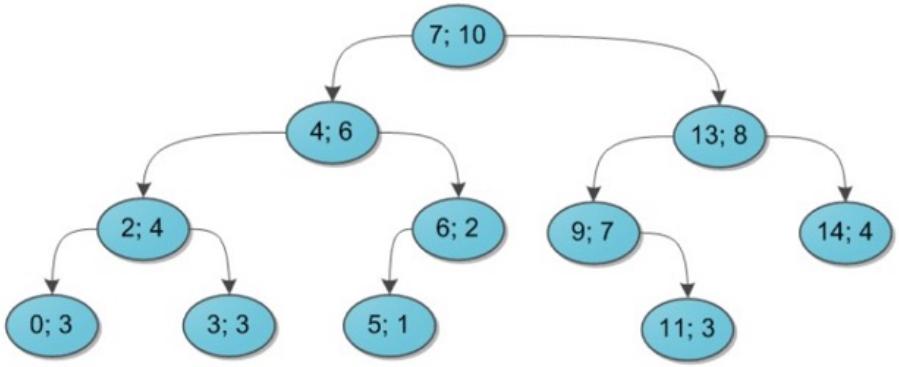
Структура данных, объединяющая в себе бинарное дерево поиска и бинарную кучу

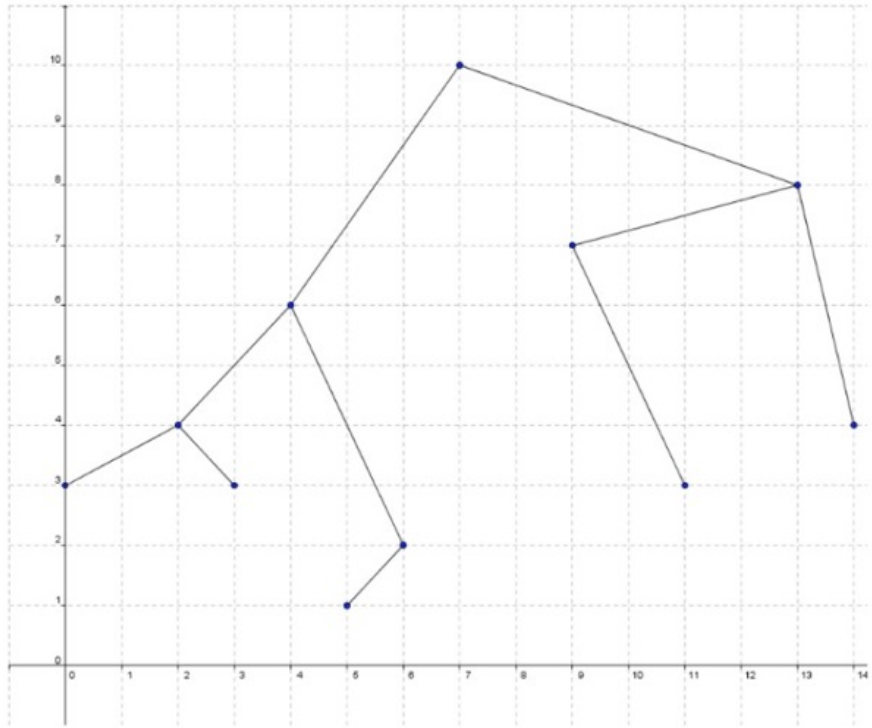
- Двоичное дерево поиска по ключу x
- Куча по приоритету y
- В одной вершине храним x и y
- Если приоритеты задаются случайно, то это может использоваться для балансировки

Другие названия:

1. treap (tree + heap)
2. дуча (дерево + куча)
3. дерамида (дерево + пирамида)
4. курево (куча + дерево)

Почему декартово?





По x - дерево поиска, по y - куча

Декартово дерево

Основные операции

Основные операции

- Вставка элемента: в среднем $O(log(N))$
- Удаление элемента: в среднем $O(log(N))$
- Поиск по ключу: в среднем $O(log(N))$
- Построение по отсортированному массиву за $O(n)$
- Поиск k-порядковой статистики: в среднем $O(log(N))$, нужно $O(n)$ доп. памяти
- Сумма, минимум, максимум на отрезке: в среднем $O(log(N))$, нужно $O(n)$ доп. памяти

Декартово дерево

Внутренние операции

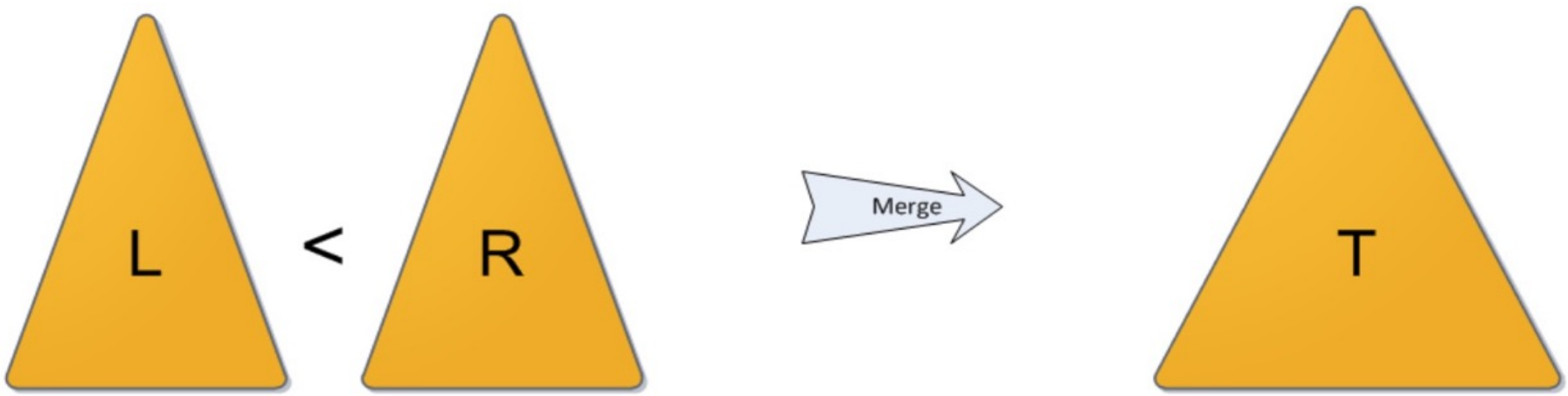
Для реализации используется 2 вспомогательные функции

- Merge - склейка 2 деревьев; все ключи одного меньше всех ключей другого: в среднем $O(log(N))$
- Split - разрезание по ключу на 2 дерева: в среднем $O(log(N))$

Декартово дерево

Merge

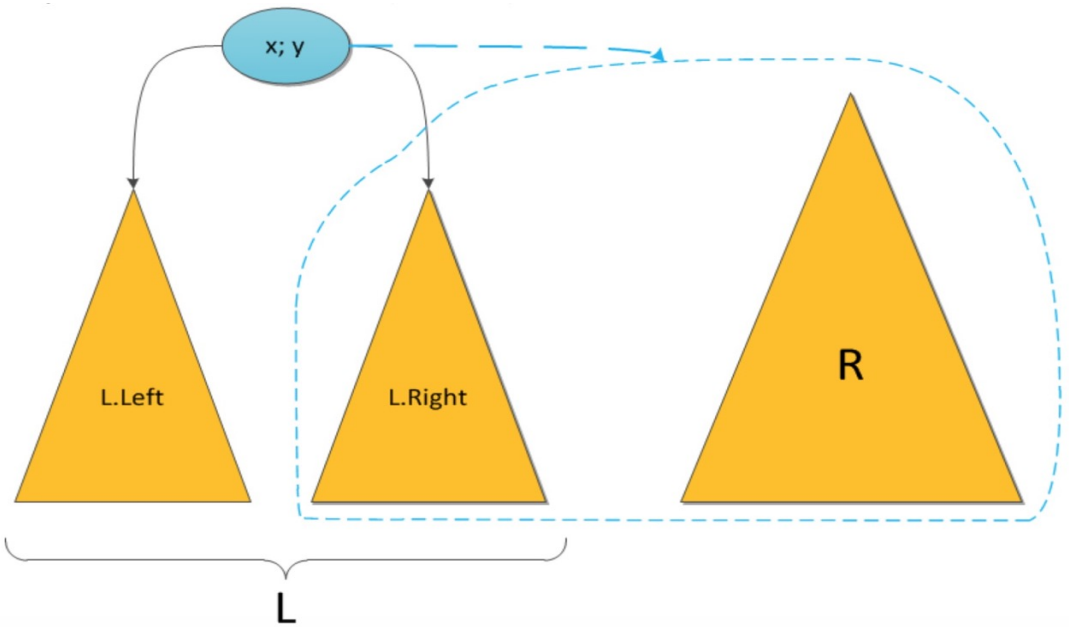
Операция Merge. На вход приходит два дерева. Цель - объединить в единое дерево.



Условие: все ключи (x) дерева L меньше ключей дерева R

Декартово дерево

Merge



Алгоритм функции Merge:

- Пусть приоритет (y) корня левого дерева больше приоритета корня правого дерева. Новый корень - корень левого дерева (без огр. общн.)
- Тогда R - точно в правом поддереве нового корня
- L.Left - точно левое поддерево нового корня
- Рекурсивно сливаем L.Right и R
- База рекурсии: хотя бы одно дерево пустое

Декартово дерево

Merge

Псевдокод функции Merge

```
Treap merge(t1: Treap, t2: Treap):
  if t2 == null
    return t1
  if t1 == null
    return t2
  else if t1.y > t2.y
    t1.right = merge(t1.right, t2)
    return t1
  else
    t2.left = merge(t1, t2.left)
    return t2
```

Декартово дерево

Merge

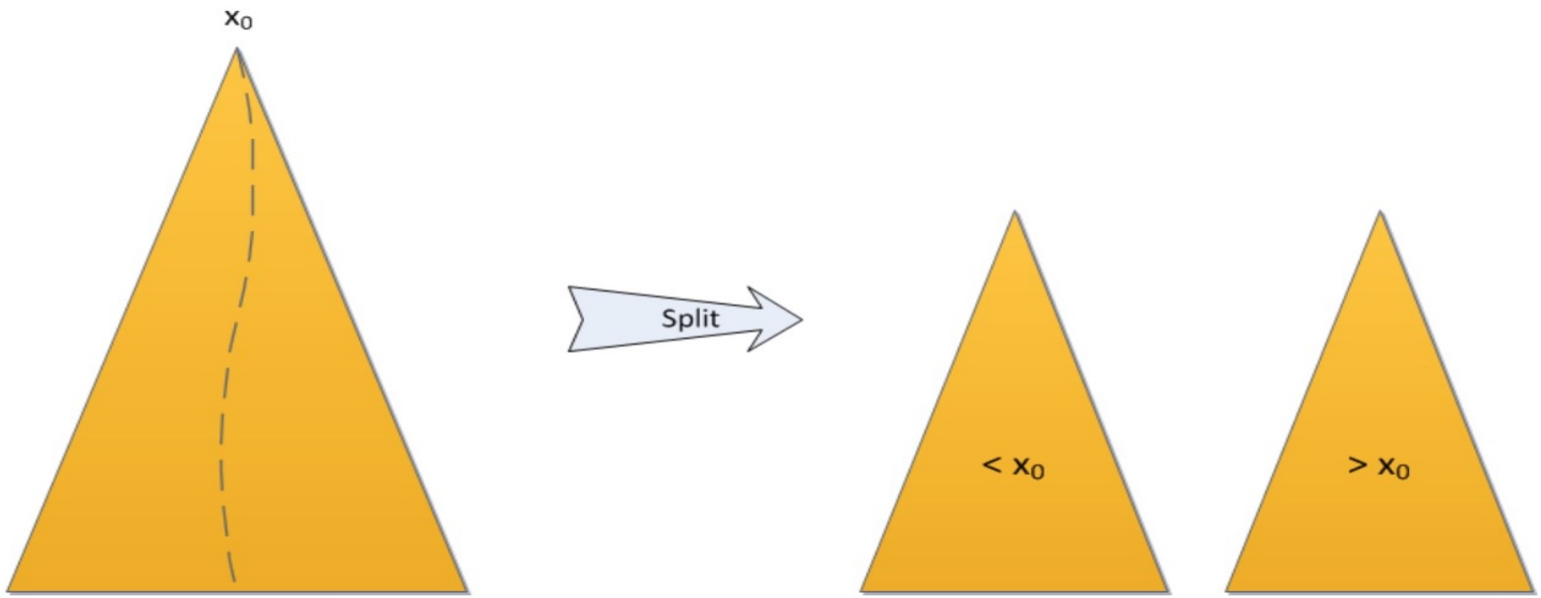
Оценка функции Merge

- Сложность: сумма высот деревьев, в среднем $O(log(n) + log(m))$

Декартово дерево

Split

Операция split (разрезать) позволяет разрезать исходное дерево T по ключу k. Возвращает пару таких деревьев T1,T2, что в дереве T1 ключи меньше k, а в дереве T2 все остальны.

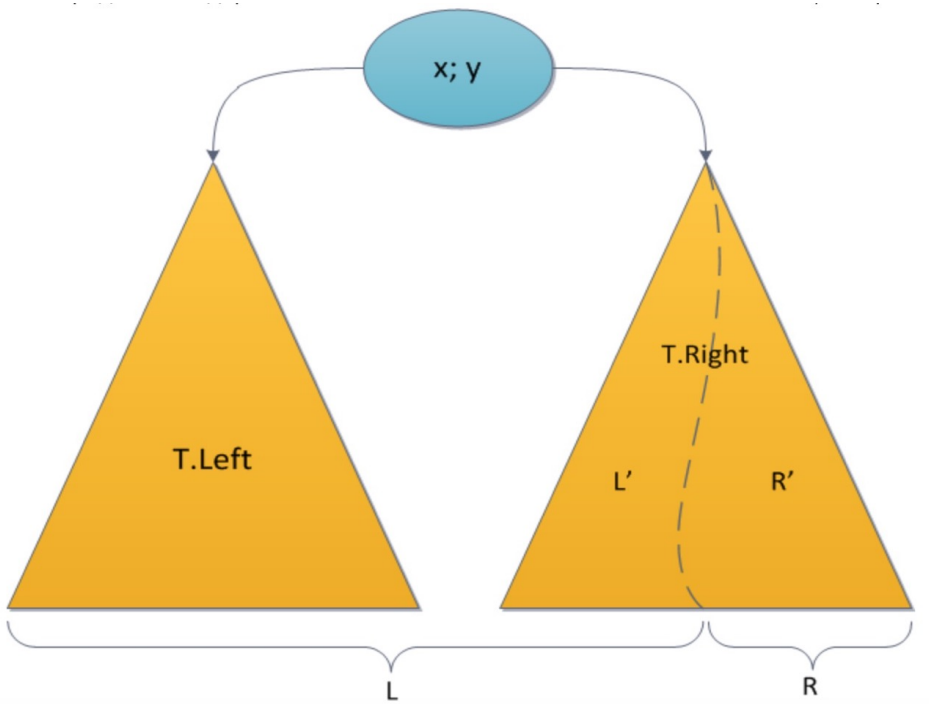


- Разделяем по ключу x_o

Декартово дерево

Split

Алгоритм функции Split



- Без огр.общности: ключ корня меньше x_0
- Рекурсивно делим правое поддерево корня на L' и R'
- L' - новое правое поддерево корня

Декартово дерево

Split

Псевдокод функции Split

```
(Treap, Treap) split(t: Treap, k: int):
  if t == null
    return (null, null)
  else if k > t.x
    (t1, t2) = split(t.right, k)
    t.right = t1
    return   t  ,  t 2
  else
    (t1, t2) = split(t.left, k)
    t.left = t2
    return (t1, t)
```

Декартово дерево

Split

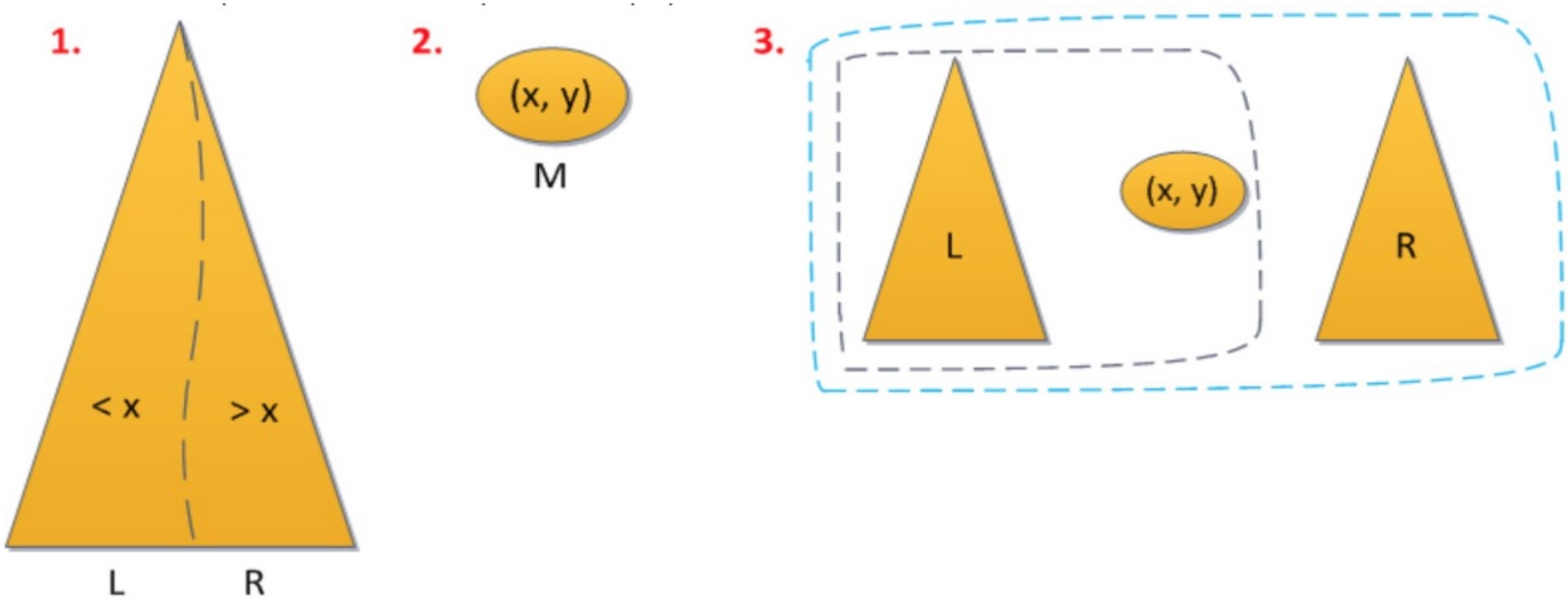
Оценка функции Split

- Сложность: высота изначального дерева, в среднем $O(log(n))$

Декартово дерево

Insert

Как реализовать функцию Insert?

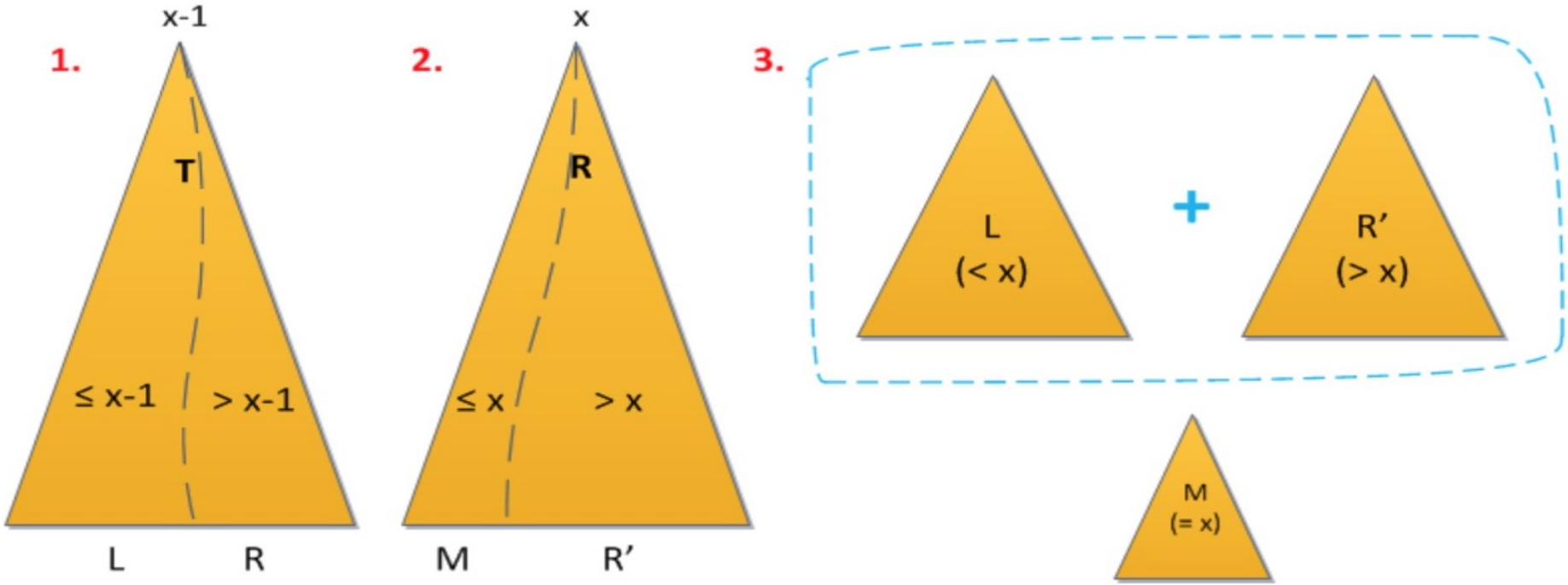


Вставка элемента (x, y)
1 Split + 2 Merge

Декартово дерево

Remove

Как реализовать функцию Remove?



Удаление элементов с ключом x

2 Split + 1 Merge

Декартово дерево

Build

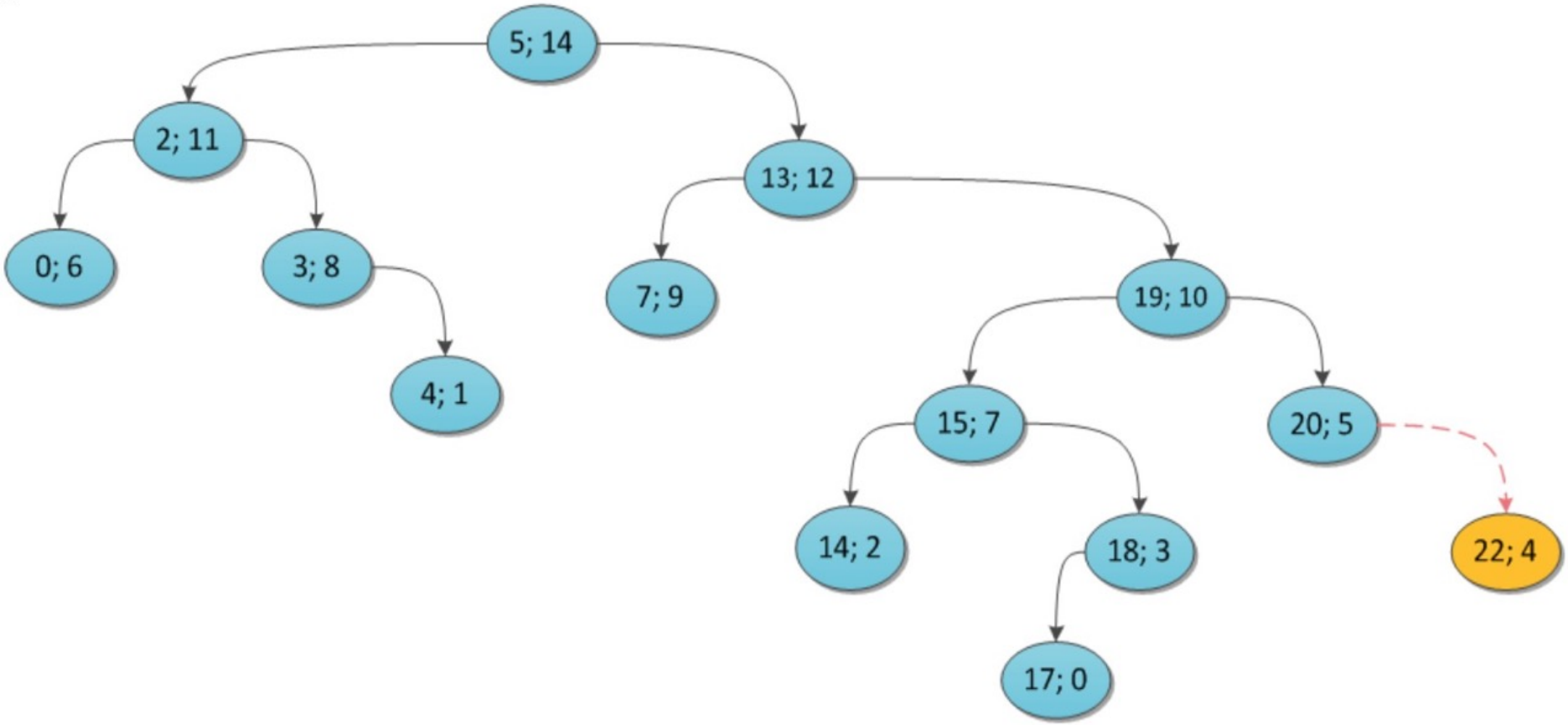
Как реализовать построение дерева?

- В случае неотсортированного массива - n вставок, $O(n \cdot log(n))$
- В случае отсортированного массива всегда рассматриваем самую правую ветку и вставляем в самую правую ветку $\rightarrow$ каждый элемент рассматривается не больше 2 раз $\rightarrow O(n)$
- Нужны ссылки на предков и на последнюю вставленную вершину

Декартово дерево

Build-1

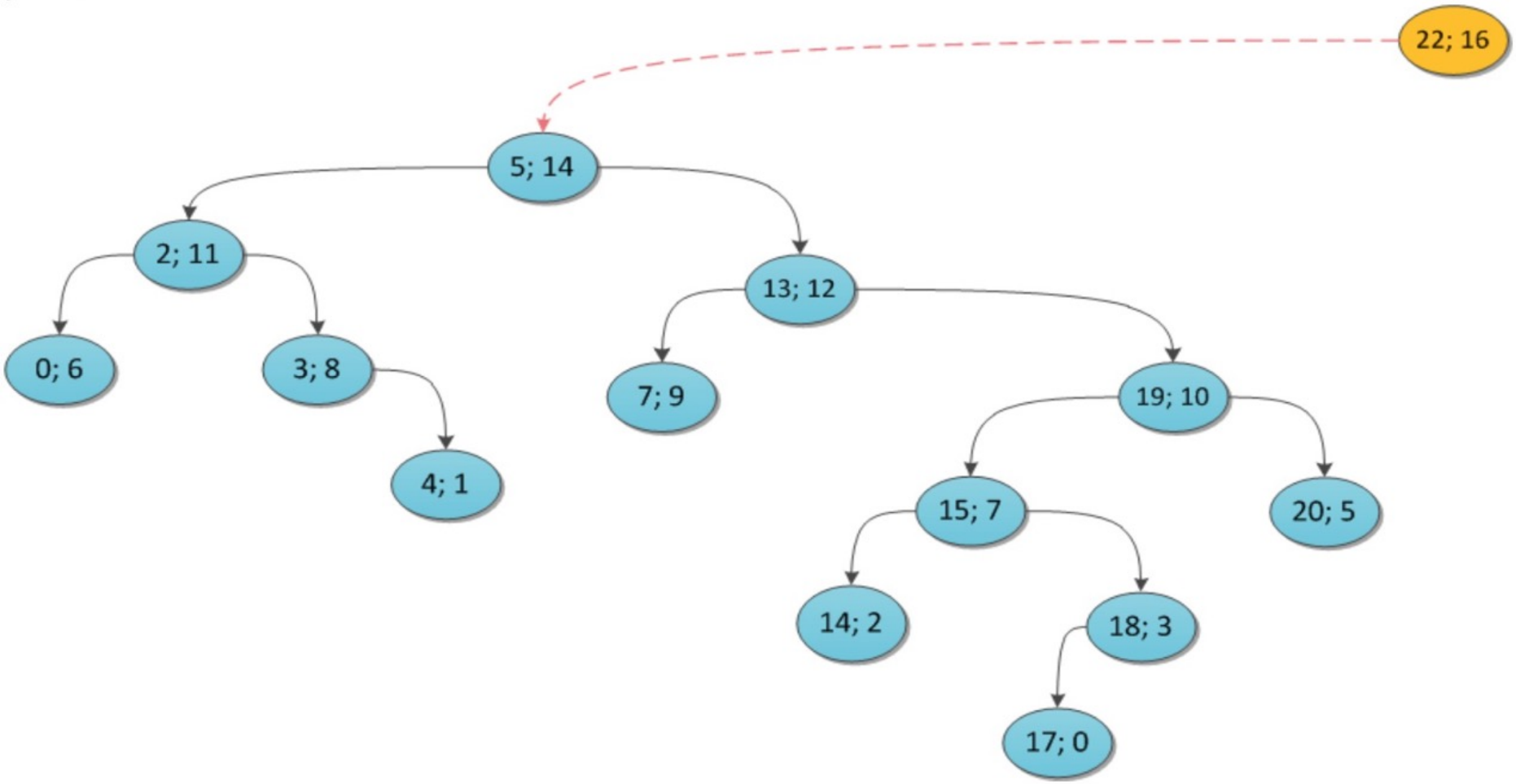
y = 4:



Декартово дерево

Build-2

y = 16:



Декартово дерево

Build-3