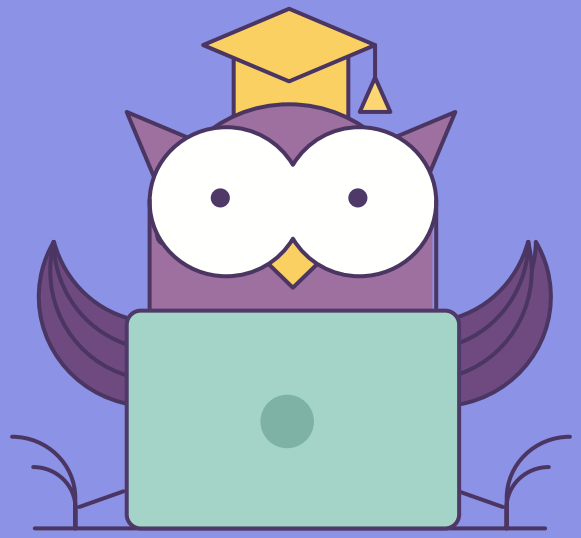
ОНЛАЙН-ОБРАЗОВАНИЕ

Не забыть включить запись!

Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Поехали!

OTUS

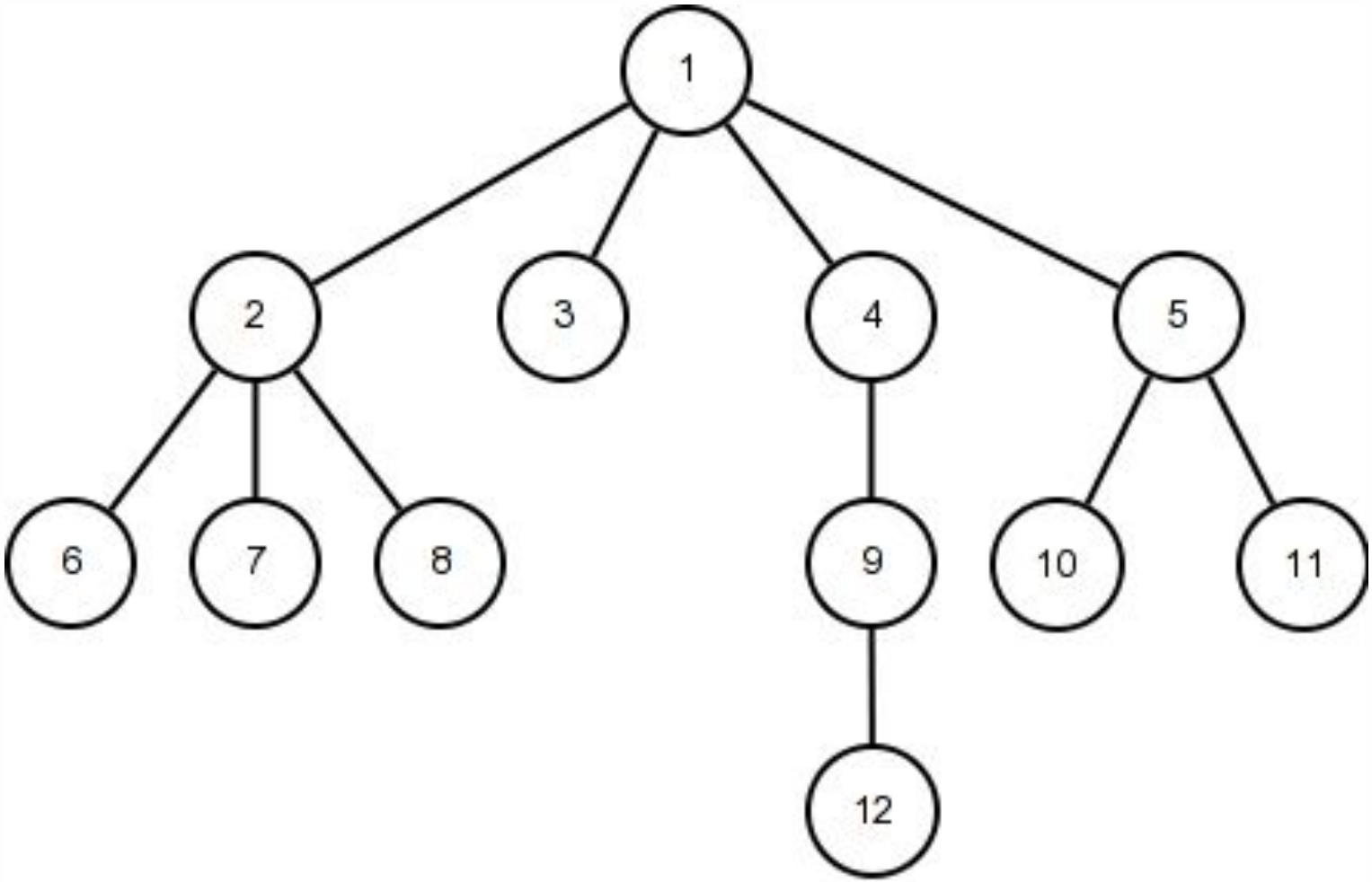
Двоичные деревья поиска, кучи, декартовы деревья

Мазнев Валерий

Содержание

- Двоичные деревья поиска
- Кучи
- Декартовы деревья

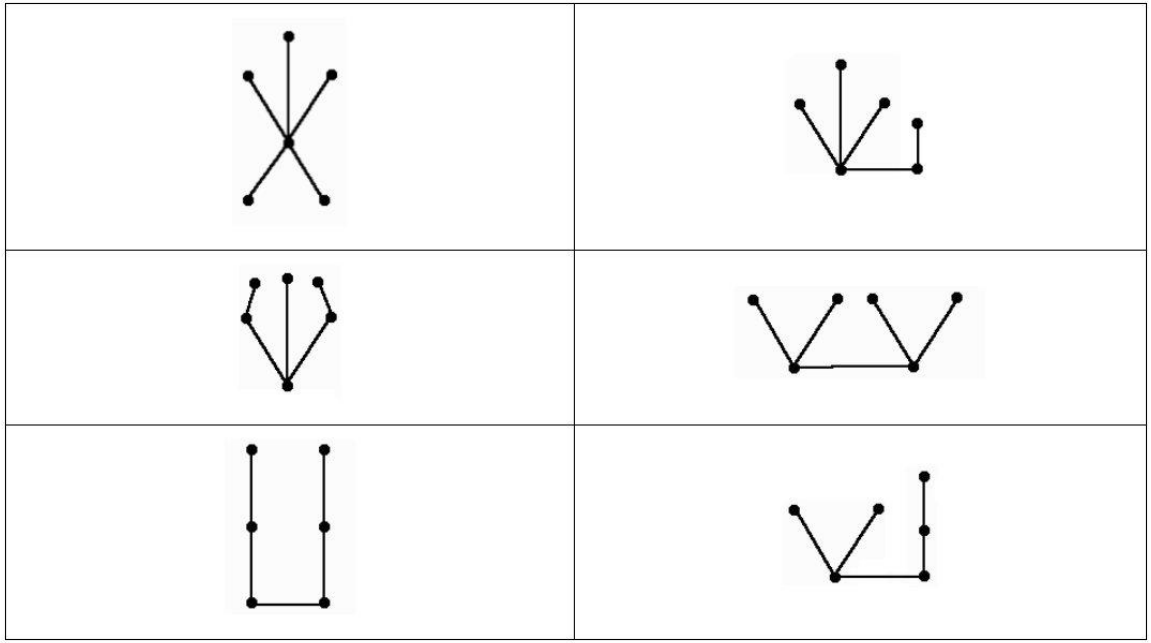
Связанный ацикличный граф.



Дерево

Свойства дерева

- Отсутствие циклов
- Между парами вершин имеется только один путь
- Ребра графа не ориентированные



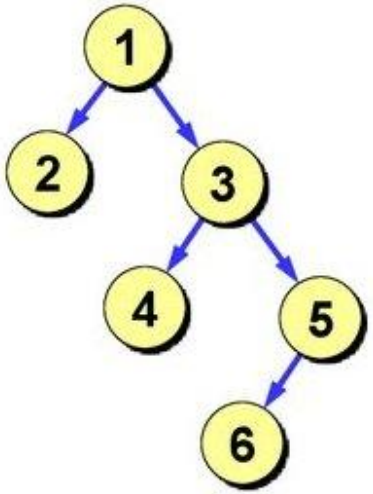
Вопрос

OTUS

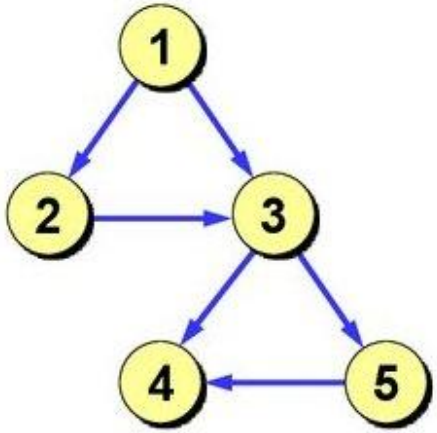
Что НЕ является деревом?

1

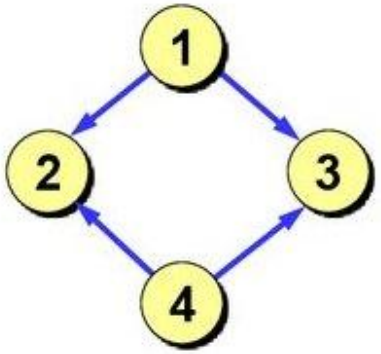
2



3

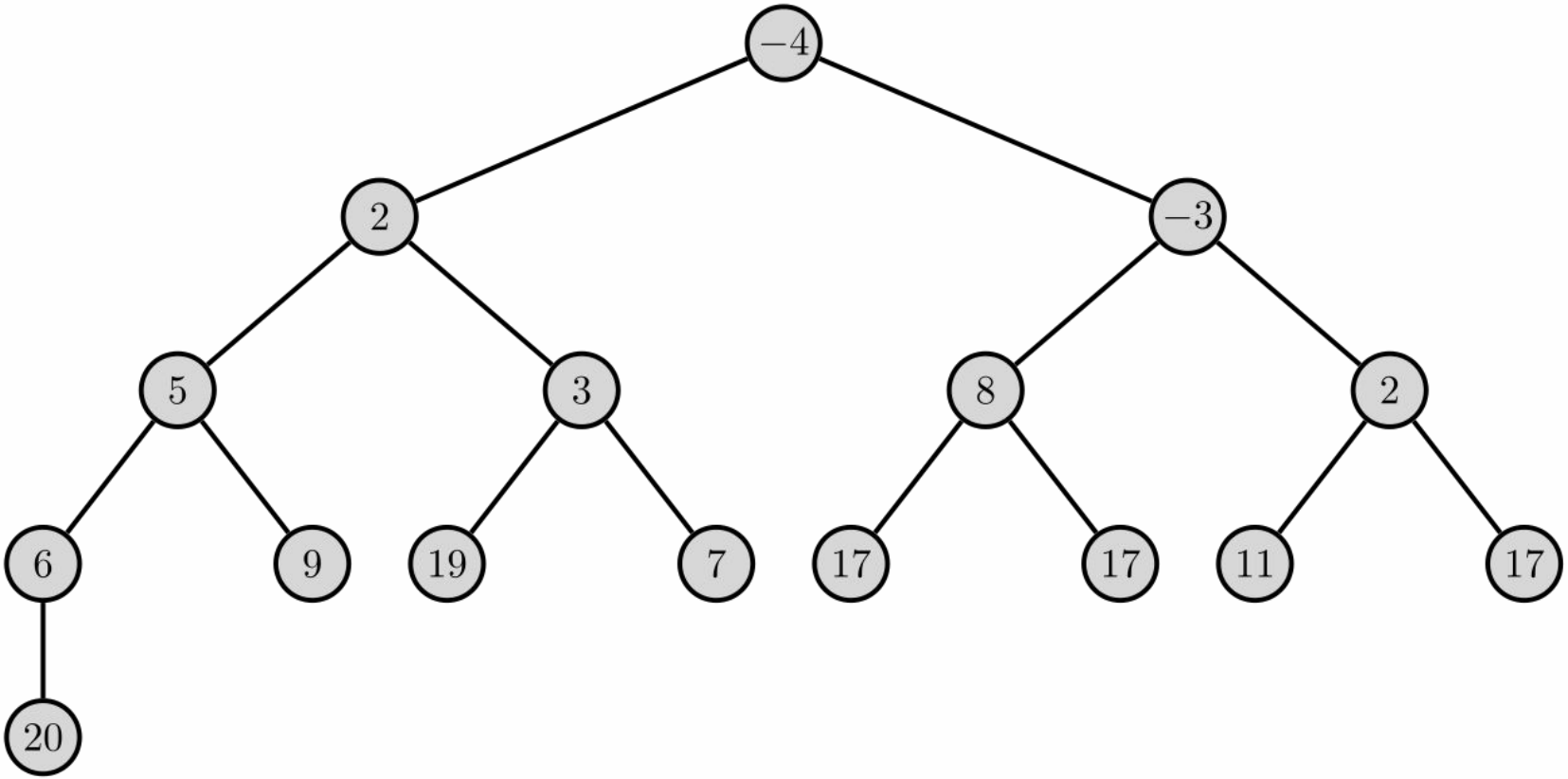


4



Термины

- Узел - данные
- Родитель - узел выше по иерархии
- Потомок - узел ниже по иерархии

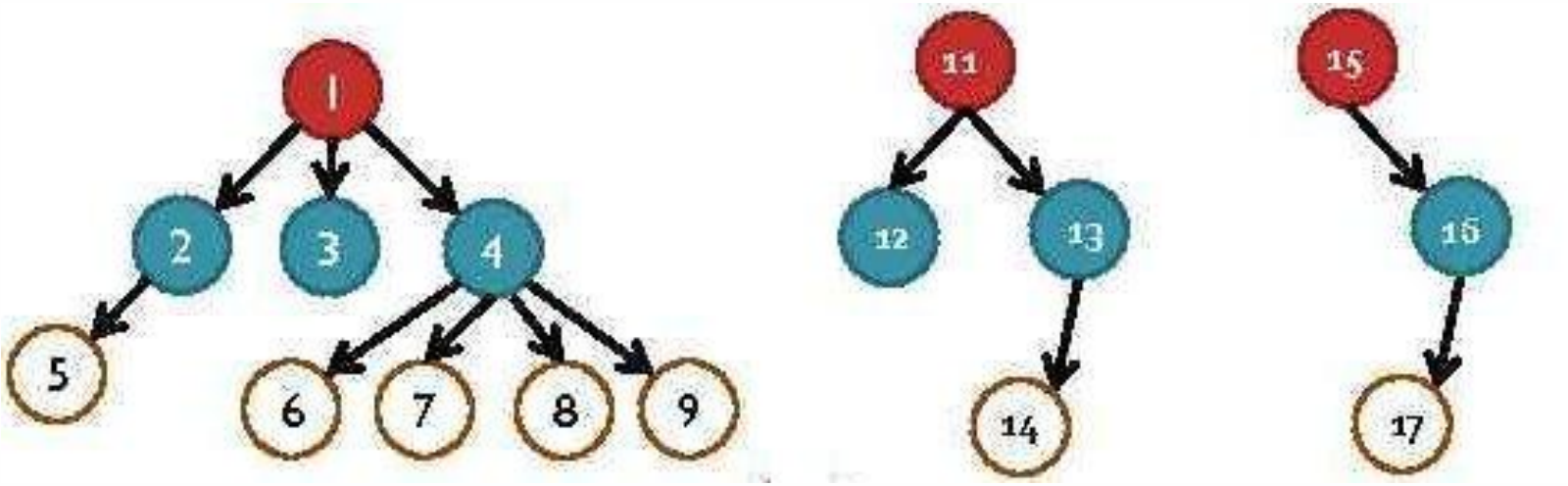


Термины

- Корневой узел — точка входа в дерево, узел не имеющий предков
- Лист, листовой или терминальный узел — узел, не имеющий дочерних элементов
- Внутренний узел — любой узел дерева, имеющий потомков, и таким образом, не являющийся листовым узлом

Термины

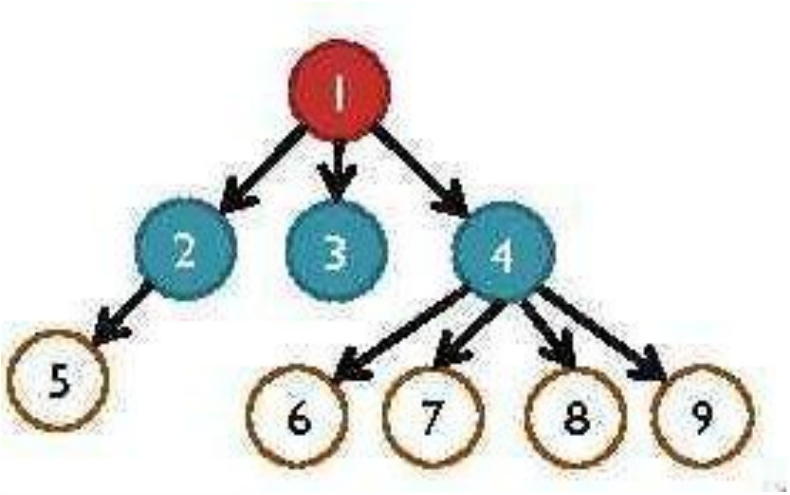
- Поддерево - часть дерева с какого либо узла
- Лес - совокупность не связанных деревьев



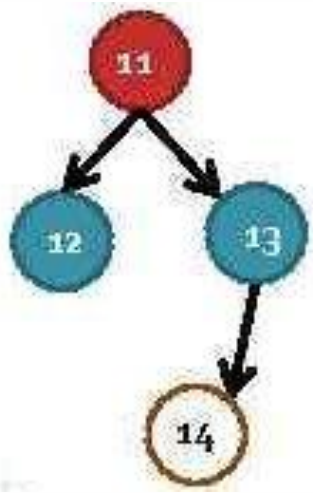
Термины

- N-арное дерево, N - макс допустимое количество потомков
- 1-арное дерево (список)
- Бинарное (двоичное) дерево

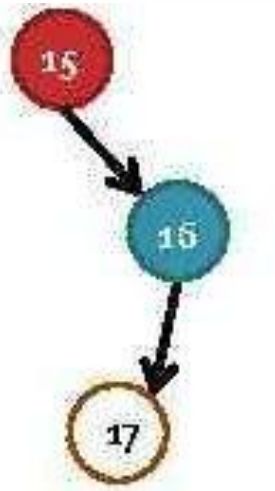
1



2



3

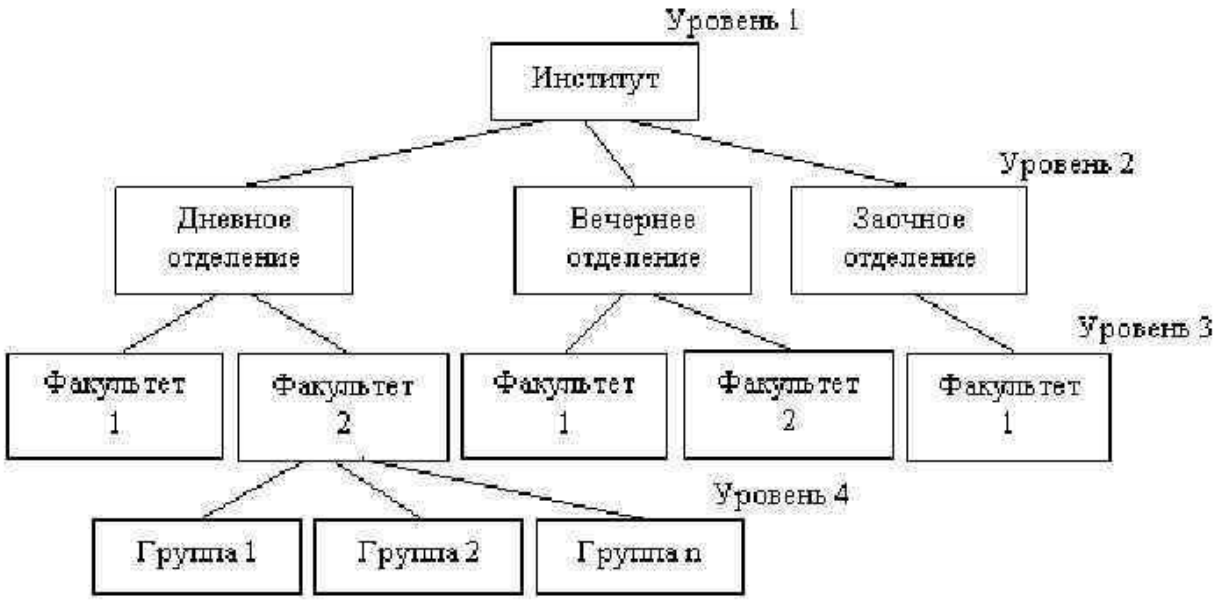


Структура данных

- Ссылочная структура в памяти
- Массив
- Записи в БД

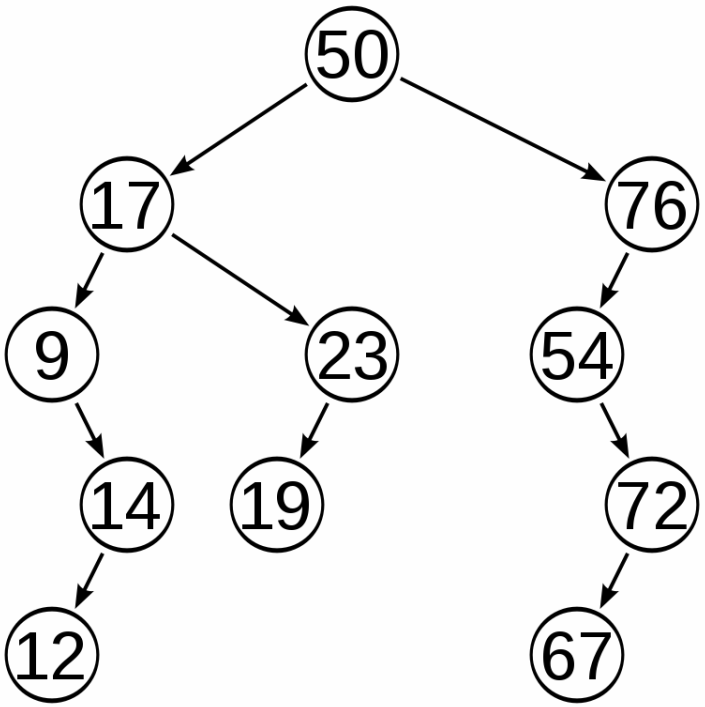
Структура в БД

- id
- parentId
- остальные данные



Пример иерархической структуры

Структура в БД



id, parentId

50, NULL

17, 50

76, 50

9, 16

23, 16

19, 23

14, 9

12, 14

54, 76

72, 54

67, 72

Такая структура дерева в БД

- Какие + ?
- Какие - ?

Двоичные деревья поиска

OTUS

- N-арность дерева = 2
- Наличие ключа key, >, <, =
- Левое и правое поддерово узла тоже двоичное дерево
- для левого key[left[X]] <= key[X]
- для правого key[right[X]] > key[X]

Алгоритм поиска

OTUS

Найти(ключ):

```
если поддерево пусто то
  вернуть <ничего не найдено>
если мойКлюч == ключ то
  вернуть моеЗначение
если мойКлюч > ключ то
  вернуть левоеПоддерево.Найти(ключ)
если мойКлюч < ключ то
  вернуть правоеПоддерево.Найти(ключ)
```

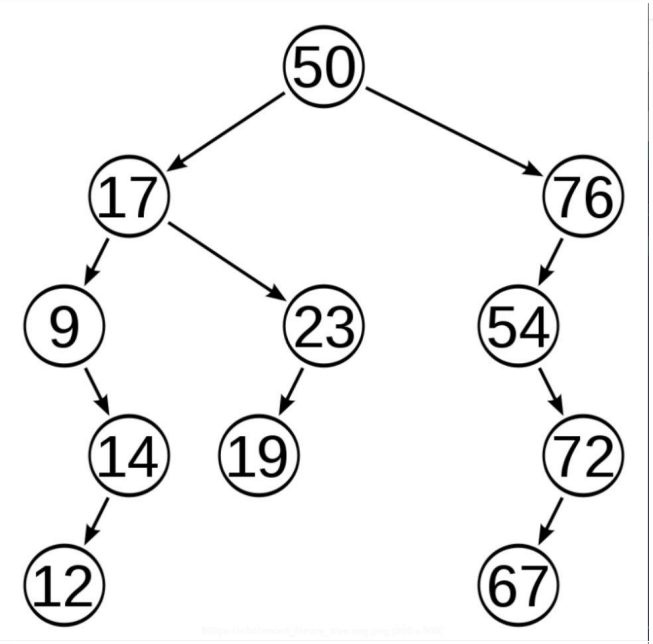
Алгоритм вставки

OTUS

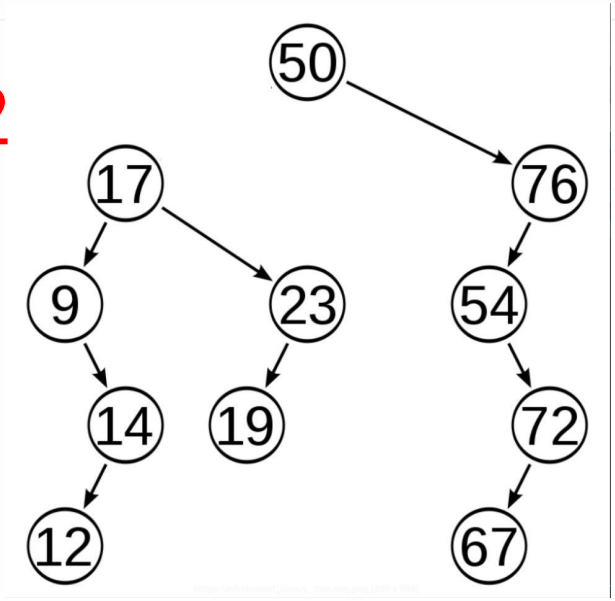
```
Вставить(ключ, значение):
если мойКлюч == ключ то
  моеЗначение = значение
если мойКлюч > ключ то
  если левоеПоддерево не пусто то
     левоеПоддерево.Вставка(ключ)
  иначе
     левоеПоддерево = новое Дерево(ключ, значение)
 иначе
     если правоеПоддерево не пусто то
       правоеПоддерево.Вставка(ключ)
     иначе
       правоеПоддерево = новое Дерево(ключ, значение)
```

Удаление узла 17

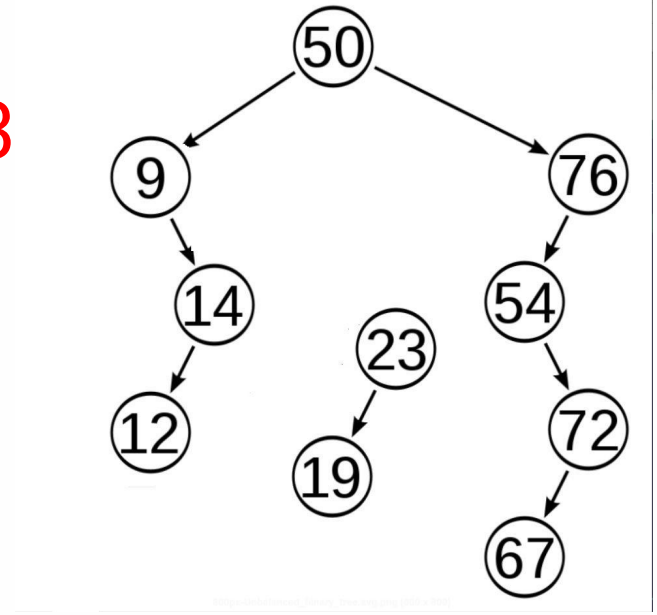
1



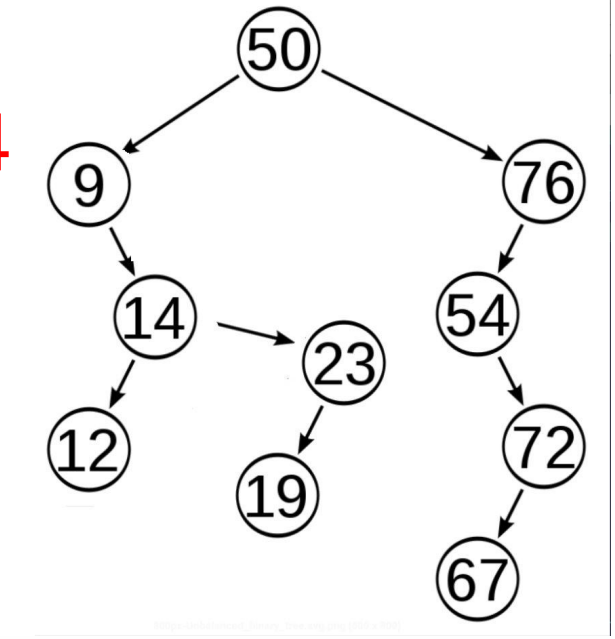
2



3



4



Удалить текущий узел

OTUS

```
УдалитьУзел():

если родитель не пусто то
        обнулить ссылку на себя у родителя
        если левоеПоддерево не пусто то
                родитель.Вставка(левоеПоддерево)
        если правоеПоддерево не пусто то
                родитель.Вставка(правоеПоддерево)
иначе
  если левоеПоддерево не пусто то
                корень = левоеПоддерево
        если правоеПоддерево не пусто то
                корень.Вставка(правоеПоддерево)
```

Алгоритм обхода прямой

OTUS

```
Обход(действие):
если левоеПоддерево не пустое то
  левоеПоддерево.Обход(действие)
Действие()
если правоеПоддерево не пустое то
  правоеПоддерево.Обход(действие)
```

Алгоритм обхода обратный

OTUS

```
Обход(действие):
если правоеПоддерево не пустое то
  правоеПоддерево.Обход(действие)
Действие()
если левоеПоддерево не пустое то
  левоеПоддерево.Обход(действие)
```

Структура данных

```
class TreeLeaf<Key, Value> {
    Key key;
    Value value;
    TreeLeaf parent;
    TreeLeaf left;
    TreeLeaf right;
}
```

Лист с неуникальными ключами

OTUS

Duplicates = true;

```
class TreeLeaf<Key, Value> {
    Key key;
    TreeLeaf parent;
    TreeLeaf left;
    TreeLeaf right;
    List values;
}
```

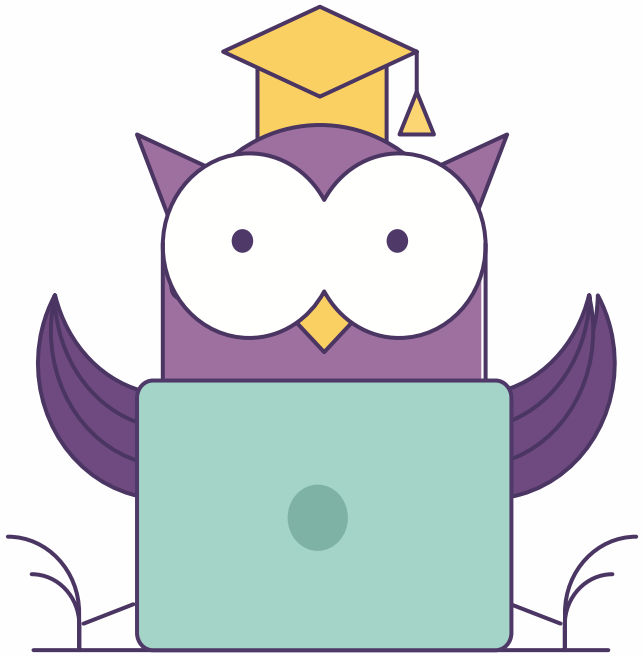
Какая сложность

- Поиск - ?
- Вставка - ?
- Удаление - ?

Операция	В среднем	В худшем случае
Поиск	O(log n)	O(n)
Вставка	O(log n)	O(n)
Удаление	O(log n)	O(n)

Ответить на вопросы

- По двоичному дереву есть вопросы?

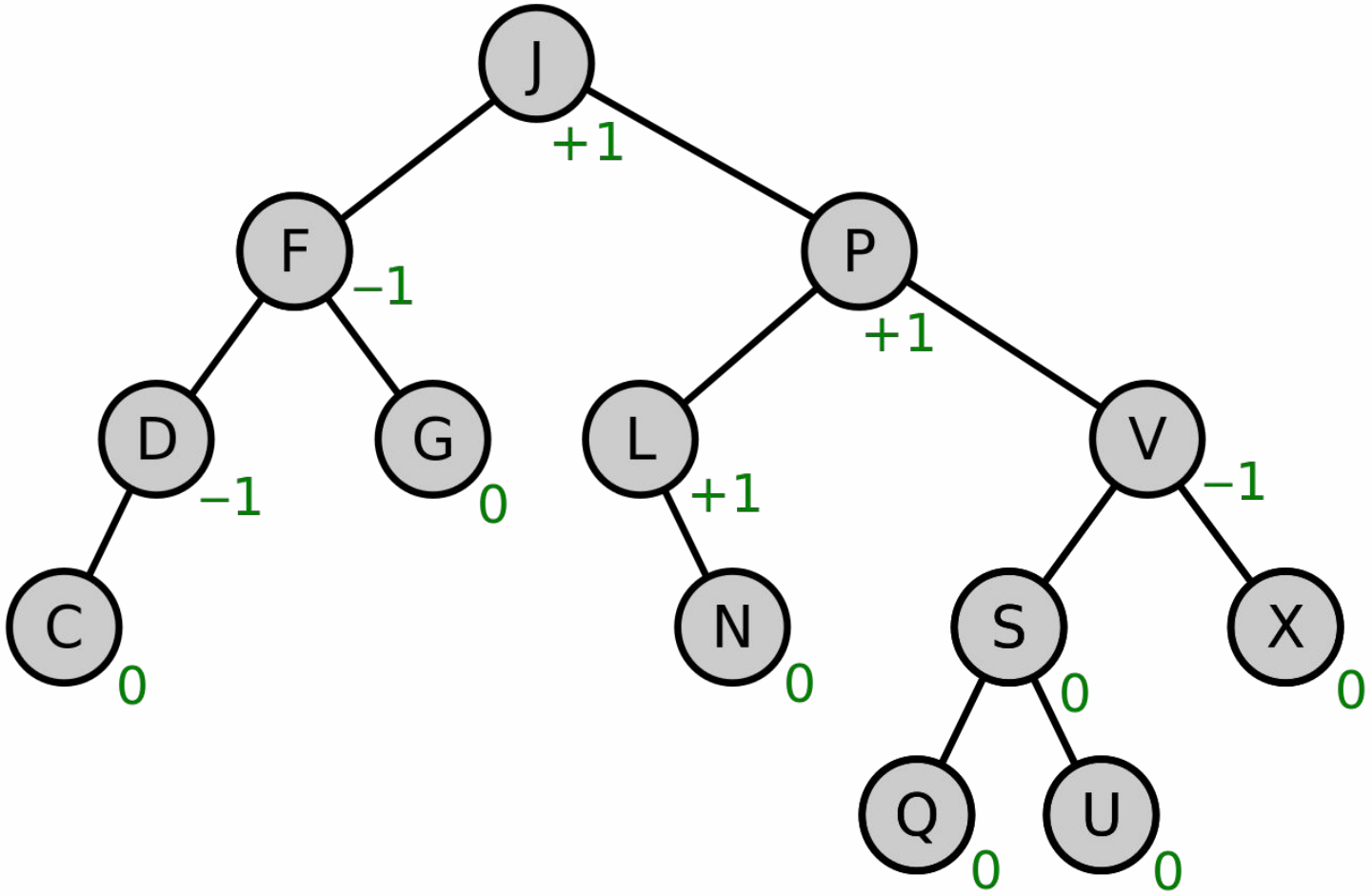


АВЛ-деревья

Сбалансированное двоичное дерево поиска

- 1962 год
- Адельсон-Вельский и Ландис

Каждая вершина хранит величину баланса



Страктура данных, базовые функции

OTUS

- храним **высоту**

- Высота(узел)
  вернуть узел пусто **?** 0 **:** узел.высота

- Баланс()
  вернуть Высота(левое) - Высота(правое)

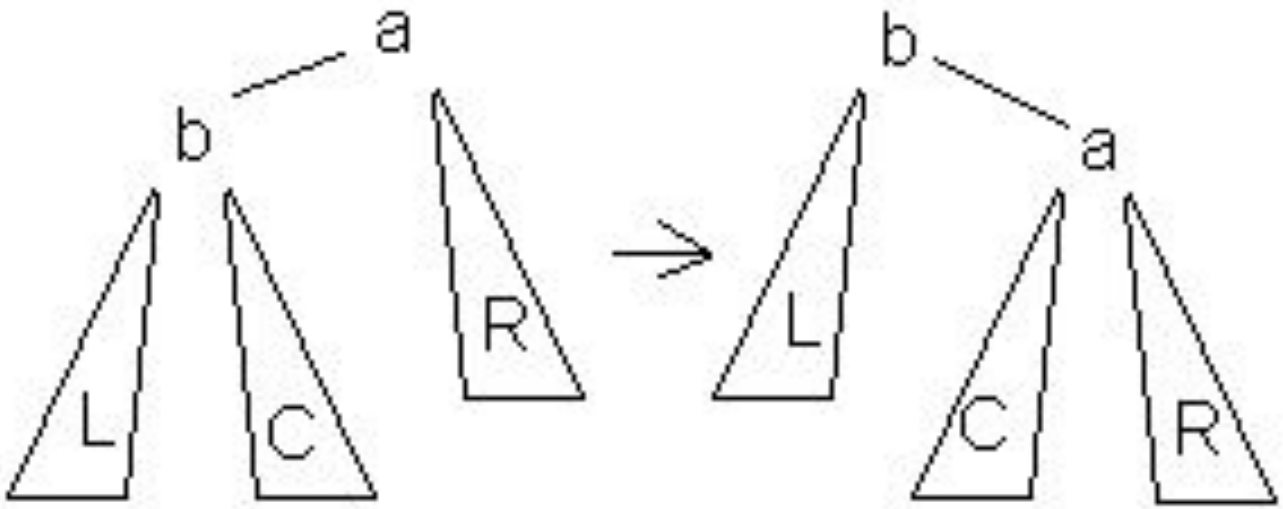
- ПересчитатьВысоту()
  высота = макс(Высота(левое), Высота(правое)) + 1;

Алгоритм вставки

```
Вставка()
    …
    ПересчитатьВысоту()
    Сбалансировать()
```

Малое правое вращение, LL

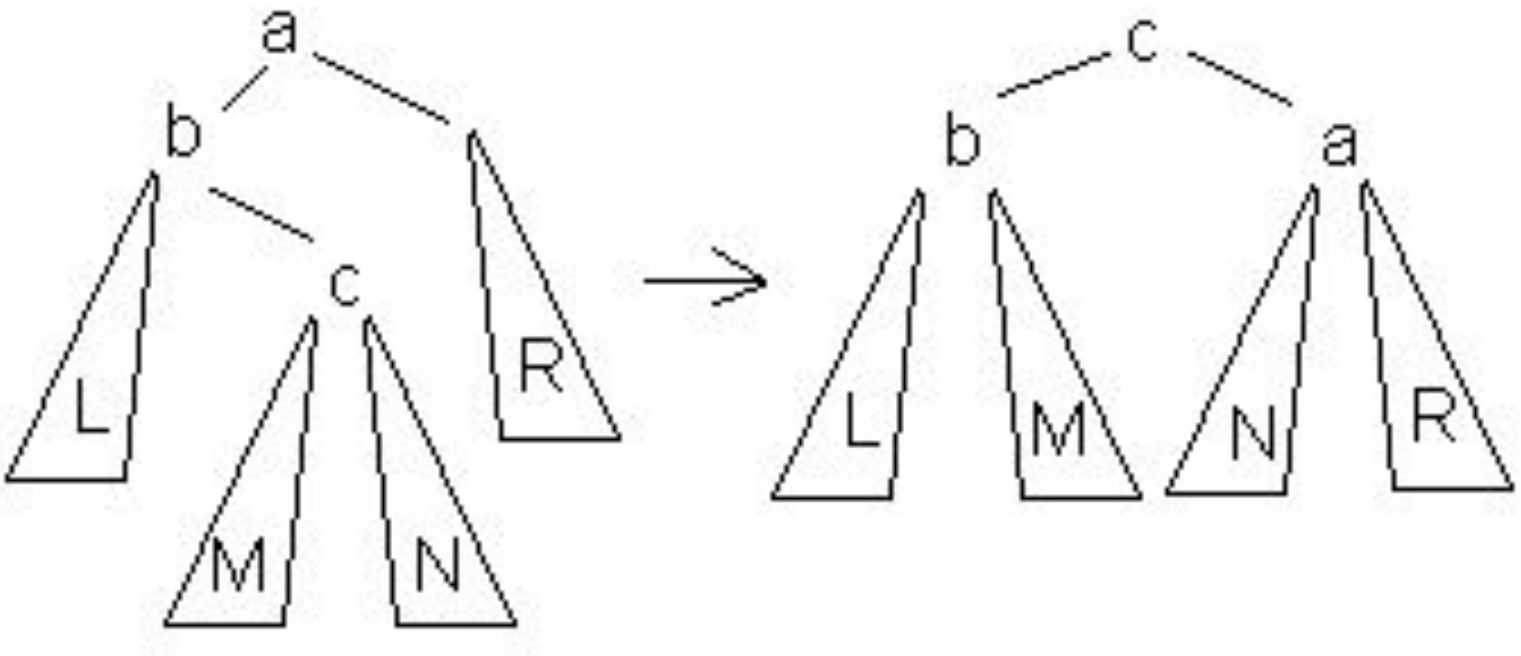
если (высота b-поддерева — высота R) = 2 **и** высота C <= высота L.



Алгоритм вставки

```
МалоеПравоеВращение()
      b = левое
      c = b.правое
      левое = c
      b.правое = текущий

      b.родитель = родитель
      родитель = b
      c.родитель = текущий
```

если (высота b-поддерева — высота R) = 2 **и** высота c-поддерева > высота L



Большое правое вращение, LR

```
БольшоеПравоеВращение()
      b = левое
       c = b.правое
      n = c.правое
      m = c.левое
      левое = n
      b.правое = m
      c.правое = текущий
      c.левое = b

       c.родитель = родитель
      a.родитель = c
      b.родитель = c
      n.родитель = текущий
      m.родитель = b
```

Balance

```
Сбалансировать()
    если баланс() >= 2 то
    ...
```

Delete

```
УдалитьУзел:
      если правое пусто то
        обнулить ссылку на себя у родителя
        если левоеПоддерево не пусто то
              родитель.Вставка(левоеПоддерево)
        если правоеПоддерево не пусто то
              родитель.Вставка(правоеПоддерево)
    иначе
     если левоеПоддерево не пусто то
              корень = левоеПоддерево
        если правоеПоддерево не пусто то
              корень.Вставка(правоеПоддерево)
```

Delete

```
Удалить():
    если левый не пусто или правый не пусто то
        если баланс > 0 то
            узел = левое.найтиМаксимальный()
        иначе
            узел = правое.найтиМинимальный()

    узел.правый = правый
    узел.левый = левый
    заменить себя у родителя
    Сбалансировать() // проверить необходимость
```

FindMin

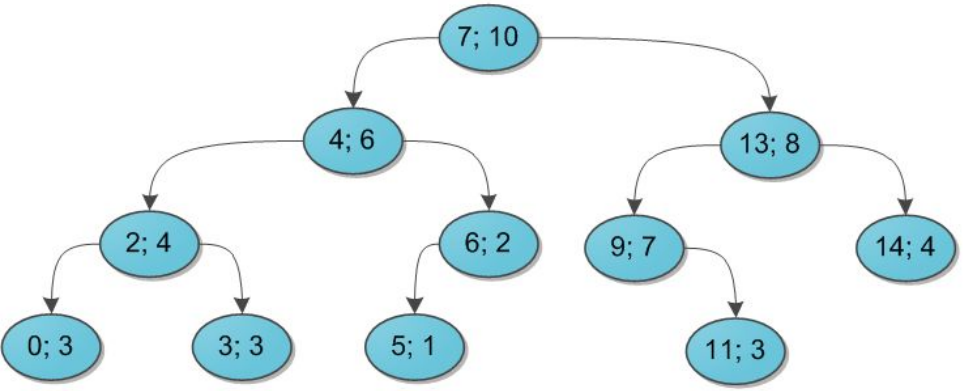
```
НайтиМинимальный():

    если левый не пусто то
        вернуть левый.НайтиМинимальный()

        родитель.левый = правый
        родитель.Сбалансировать()
        вернуть текущий
```

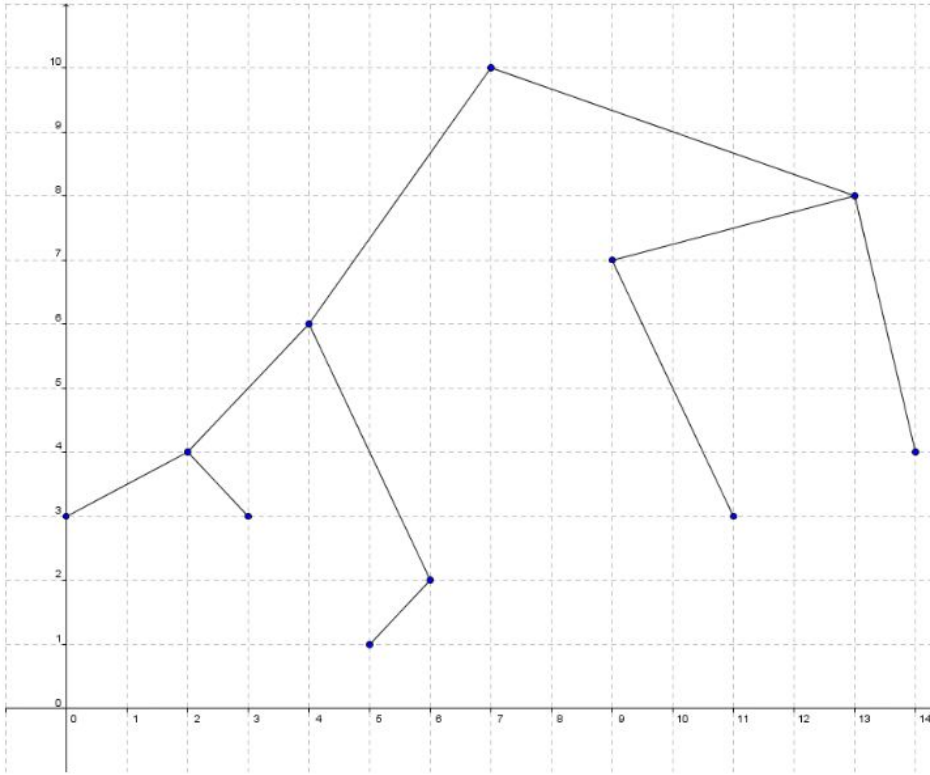
Декартово дерево

Структура данных, упорядоченная по 2-м ключам x, y, причем

- по оси x - дерево поиска
- по оси y - куча

Декартовы деревья





Термины

- treap (tree + heap)
- дуча (дерево + куча)
- дирамида (дерево + пирамида)
- курево (куча + дерево :)

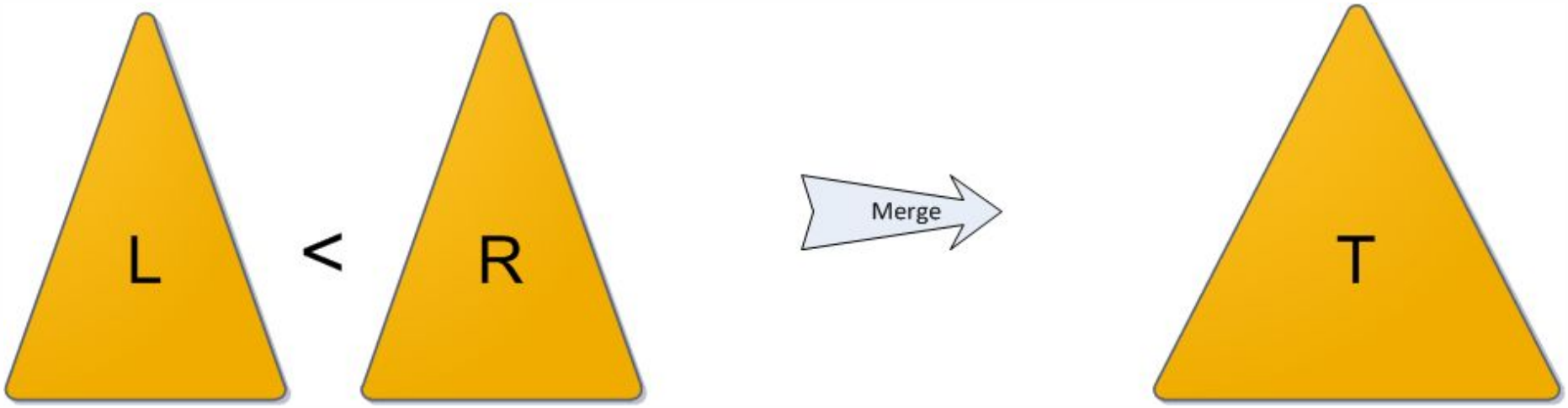
Свойства

- Пара значений x, y однозначно определяет структуру дерева

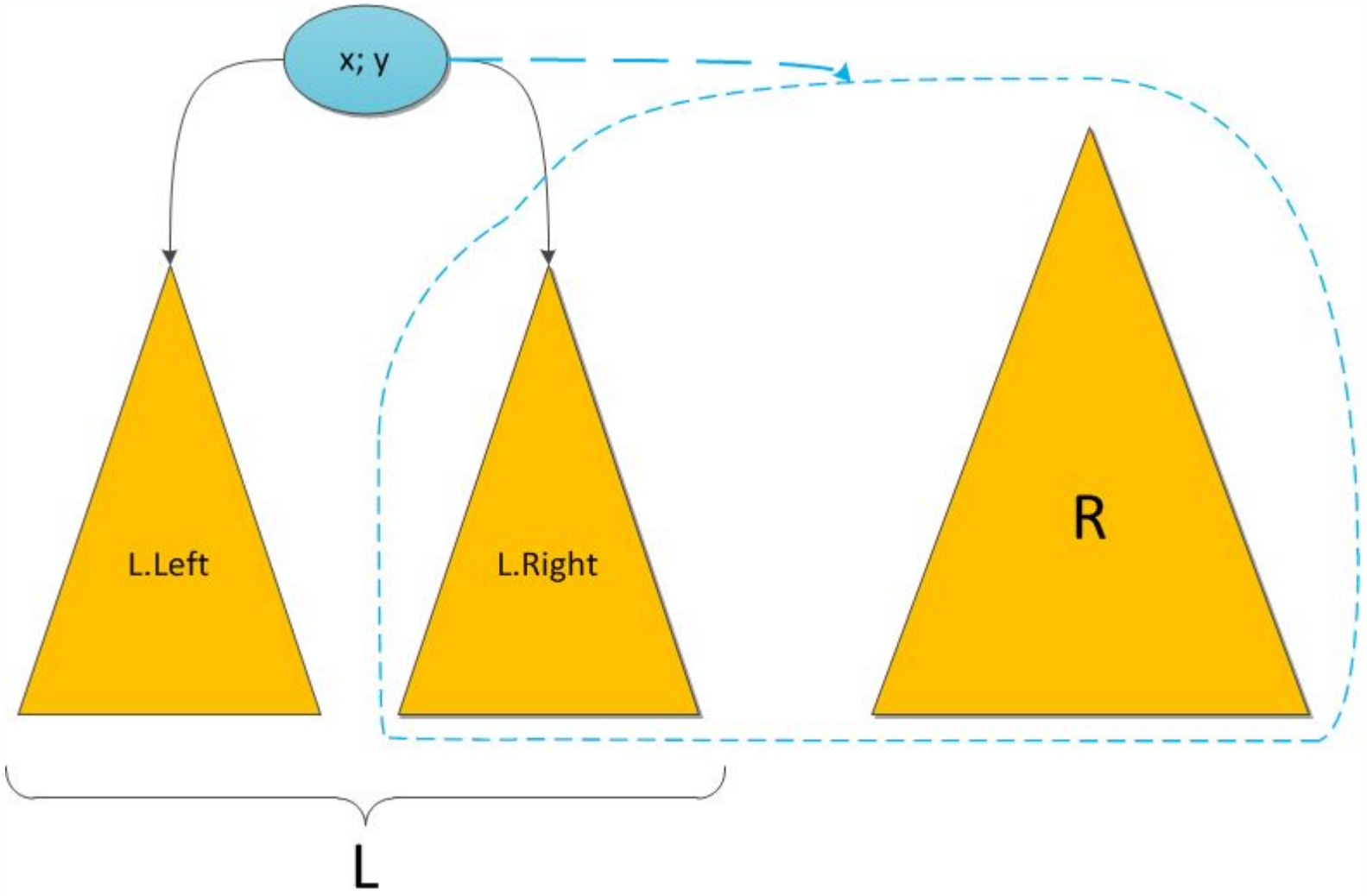
Структура данных

```
class Treap {

    int x, y;

    Treap Left, Right;

    Treap(int x, int y,
     Treap left = null,
     Treap right = null) { … }
}
```

Объединить



Объединить

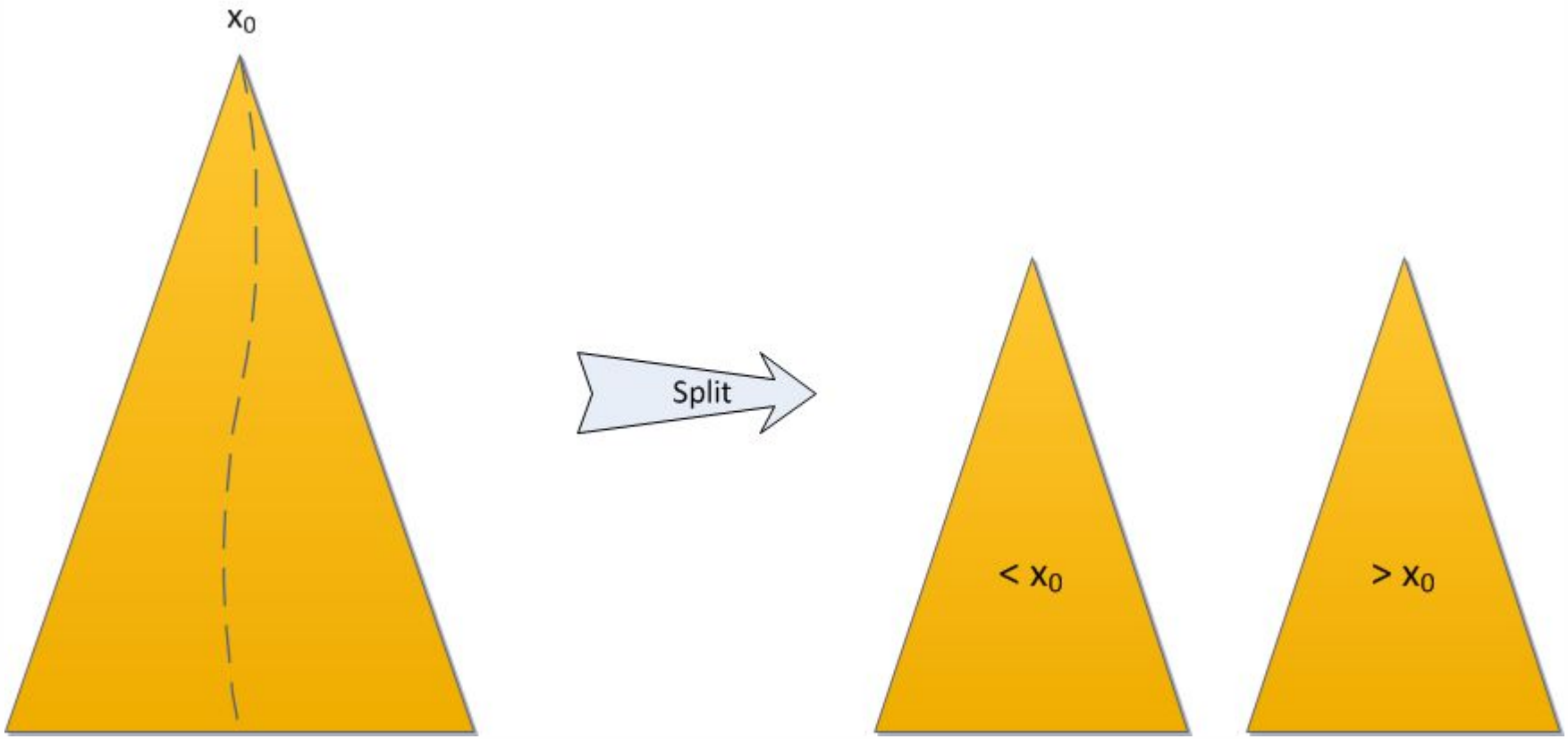


Merge

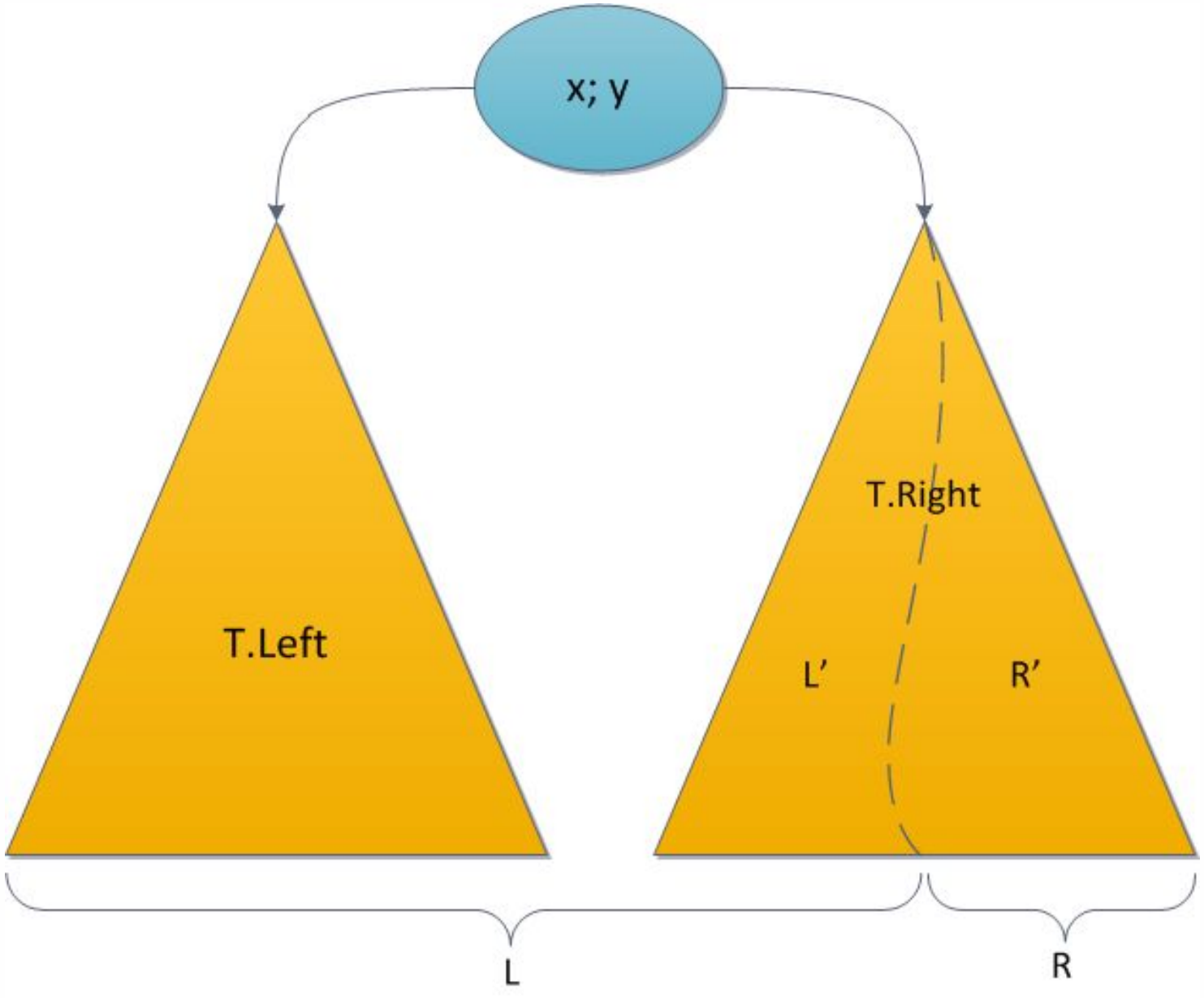
OTUS

```
public static Treap Merge(Treap L, Treap R)
{
    if (L == null) return R;
    if (R == null) return L;

    if (L.y > R.y)
    {
        var newR = Merge(L.Right, R);
        return new Treap(L.x, L.y, L.Left, newR);
    }
    else
    {
        var newL = Merge(L, R.Left);
        return new Treap(R.x, R.y, newL, R.Right);
    }
}
```

Разделить по ключу X0



Разделить по ключу X0

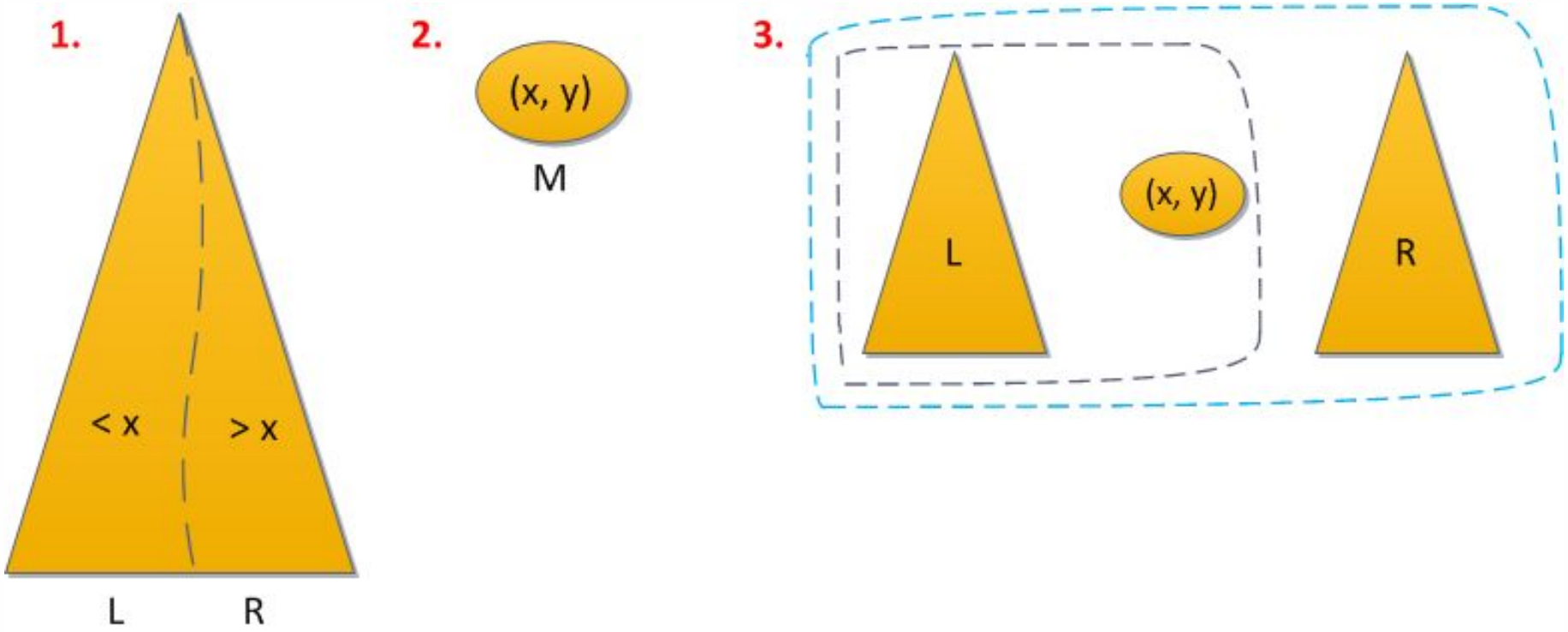


Split

OTUS

```
public void Split(int x, out Treap L, out Treap R)
{
    Treap newTree = null;
    if (this.x <= x)
    {
        if (Right == null)
            R = null;
        else
            Right.Split(x, out newTree, out R);
        L = new Treap(this.x, y, Left, newTree);
    }
    else
    {
        if (Left == null)
            L = null;
        else
            Left.Split(x, out L, out newTree);
        R = new Treap(this.x, y, newTree, Right);
    }
}
```

Добавить элемент

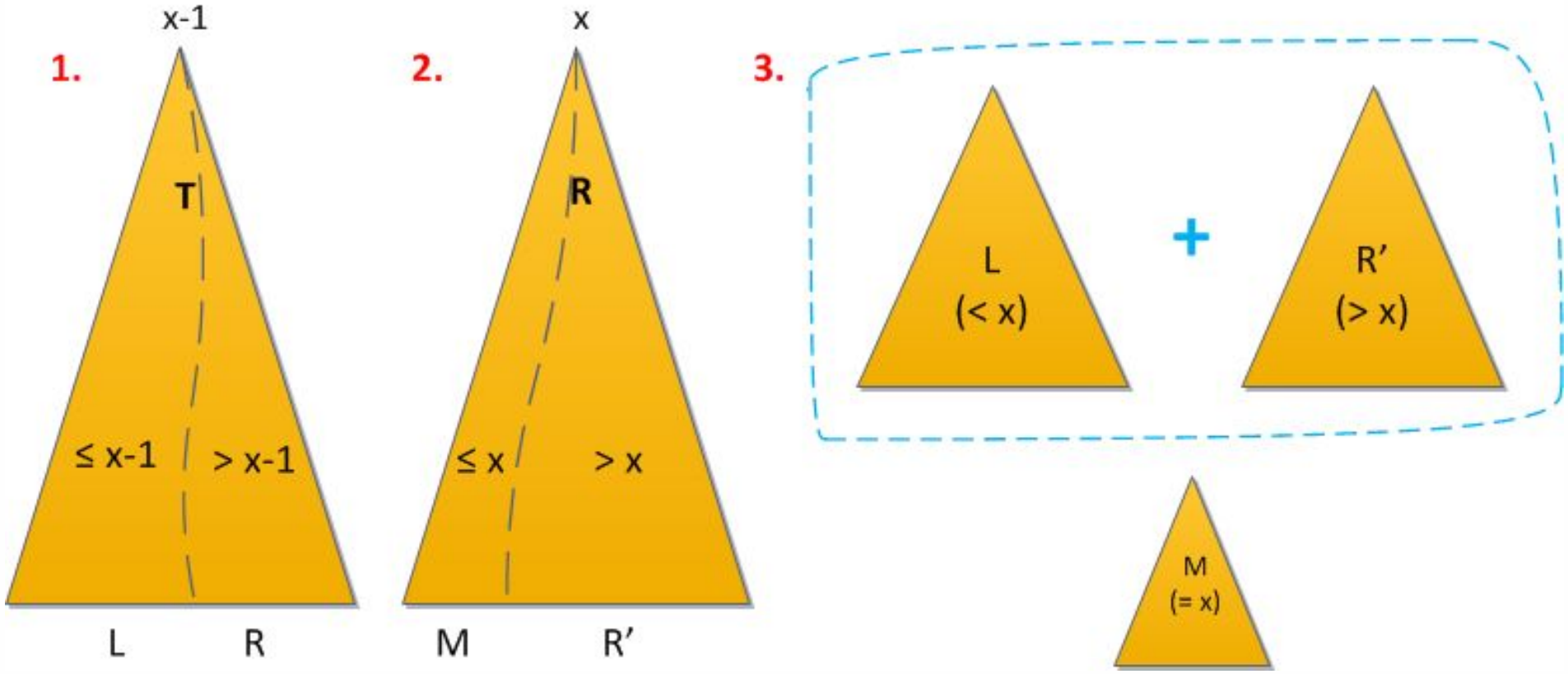


Add

OTUS

```
public Treap Add(int x)
{
    Treap l, r;
    Split(x, out l, out r);
    Treap m = new Treap(x, rand.Next());
    return Merge(Merge(l, m), r);
}
```

Удалить элемент



Remove

OTUS

```
public Treap Remove(int x)
{
    Treap l, m, r;
    Split(x - 1, out l, out r);
    r.Split(x, out m, out r);
    return Merge(l, r);
}
```

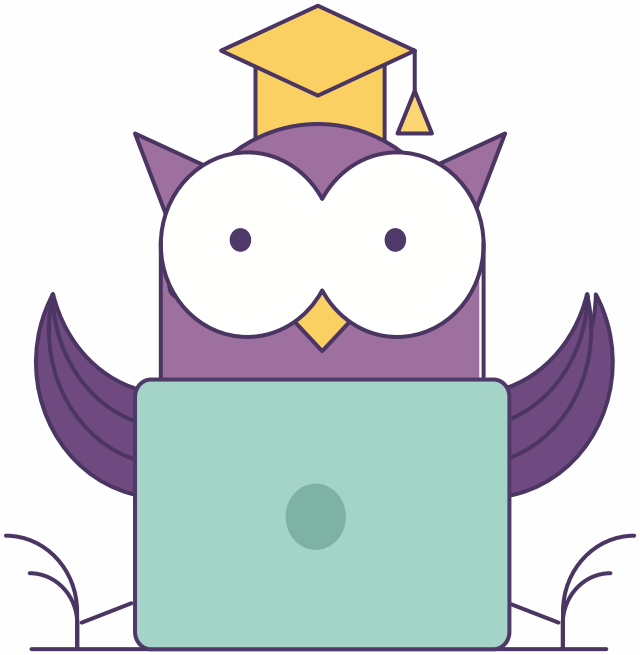
Реализовать кучу

Опционально:

Реализация декартова дерева

Ответить на вопросы

Что такое - дерево?
Что такое двоичное дерево поиска?
Что такое декартово дерево?
Что такое АВЛ-дерево?



Заполните, пожалуйста, опрос о занятии

Спасибо
за внимание!