# Декартово дерево
(cartesian tree)

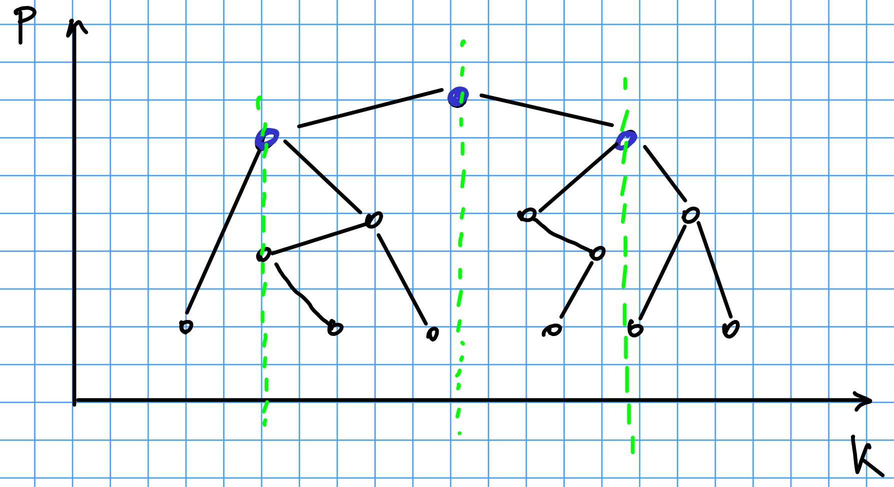
= бинарное дерево
в $\forall$ вершине хранится ключ $k$
и приоритет $p$.

Д. д. — дерево поиска по ключам
и куча по приоритетам

$(k_1, p_1), (k_2, p_2), \ldots, (k_n, p_n)$

## Лемма (Th. о сущ.)

Для $\forall (k_1, p_1) \ldots (k_n, p_n)$ декартово дерево $\exists$ и единственно (при условии, что все приоритеты разные)

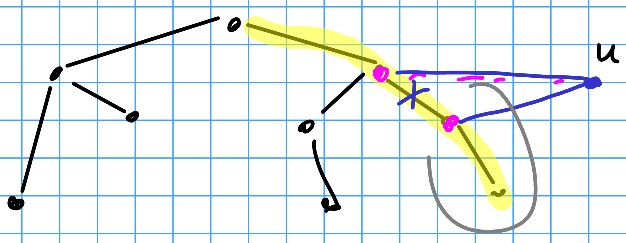


$O(n^2)$

## Эффективное построение

Шаг 1: сортировка по ключам $O(n\log n)$
Шаг 2: $(k_1, p_1) \ldots (k_n, p_n)$, $k_{i+1} \geq k_i \geq k_{i-1}$

Будем по очереди добавлять вершины в декартово дерево, т.е. всегда доб. правую вершину




Пусть $v_1 \dots v_k$ — это вершины самого правого пути.

На # шаге мы ищем $v_i$ и $v_{i+1}$:

$$pri(v_i) \geq pri(u) > pri(v_{i+1})$$

Если искать от корня $\Rightarrow O(n^2)$

Если искать с конца $\Rightarrow O(n)$

(нужно хранить указатель на самую правую в.)

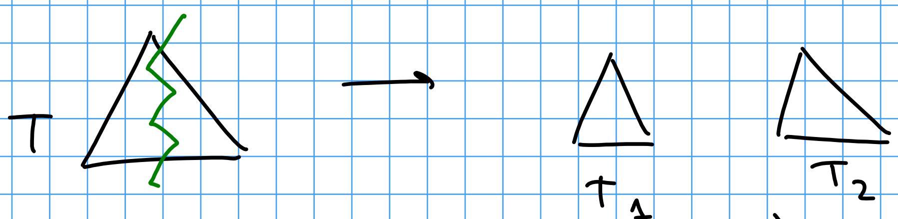
Наблюдение: # вершина просм. $\leq 2$ раз

Потенциал $\Phi$ = # вершин в правом пути

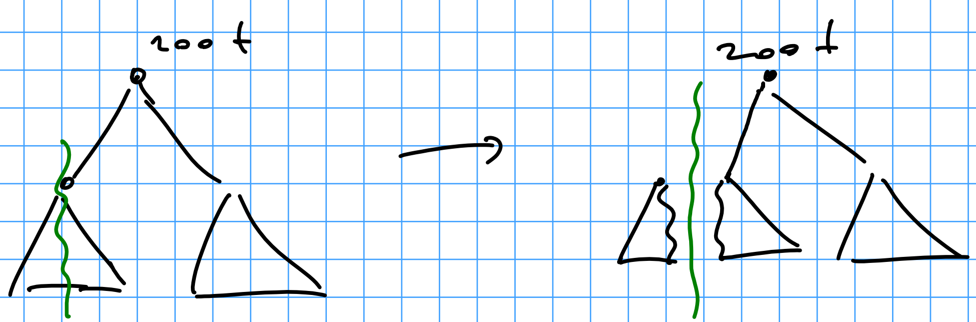
Если мы пройдём $t$ вершин, то

аморт. стоимость $\tilde{c} = \underbrace{(t+1)}_{c} \underbrace{- (t-1)}_{\Delta\Phi} = 2$

Split$(T, k)$:

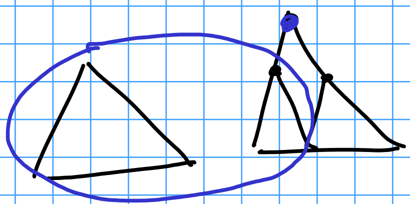



$$key(T_1) < k \leq key(T_2)$$



```
Split (T, k):
    if  k < key(T.root):                          O(h)
        T1, T2 = Split (T.left)
        T.left = T2
        return (T1, T)
    else
        ....
```

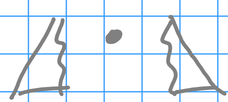
```
Merge (T1, T2): // key(T1) ≤ key(T2)
```



O(h)

```
    if pri (T2.root) > pri (T1.root):
        T2.left = Merge (T1, T2.left)
        return T2
    else
        T1.right = Merge (T1.right, T2)
        return T1
```

```
Insert (T, (k, p)):
    T1, T2 = Split (T, k)
```



```
    return Merge (T1, Merge((k,p), T2))
```

```
Remove (T, k):
    T1, T2 = Split (T, k)
    T3, T4 = Split (T2, k+1)
    return Merge (T1, T4)
```

Дуча (treap) / Курево // Дерамида

≡ Декартово дерево со случайными приоритетами

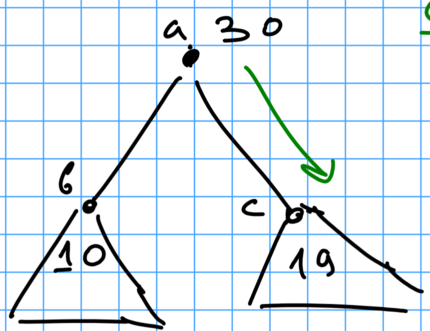
Лемма: матожидание высоты дучи $= O(\log n)$

См. глубину рекурсии Quick Sort.

## Декартово дерево с неявным ключом

Значение ключа требуется только в Split.

Давайте вместо ключей хранить размер поддерева в $\forall$ вершине




Split(a, 20)

Split(c, 20 - 10 - 1)

$\Rightarrow$ "Массив" с операциями за $\log n$ в среднем:

- Insert
- Delete
- Find
- Merge
- Split

....