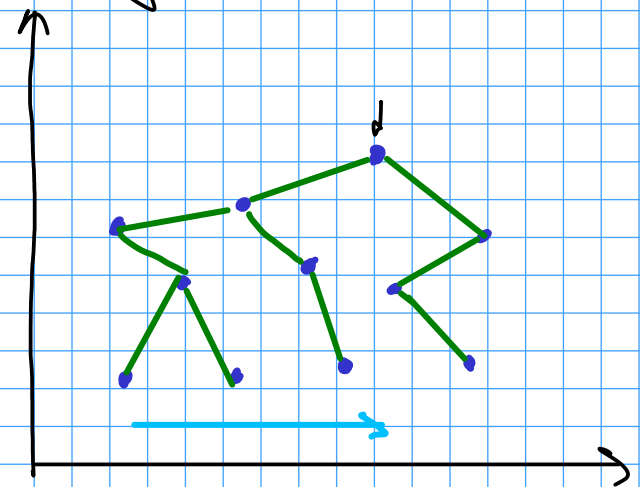
Декартово дерево

(cartesian tree)

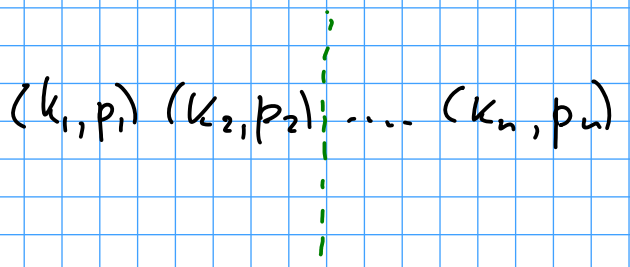
Вход: $(k_1, p_1), (k_2, p_2) \dots$



1. Дерево поиска отн k_i
2. Куча отн. p_i

Усл: Декартово дерево $\exists$ для любых входн. данных.

$\exists$ $k_i \le k_{i+1}$ $\forall i$

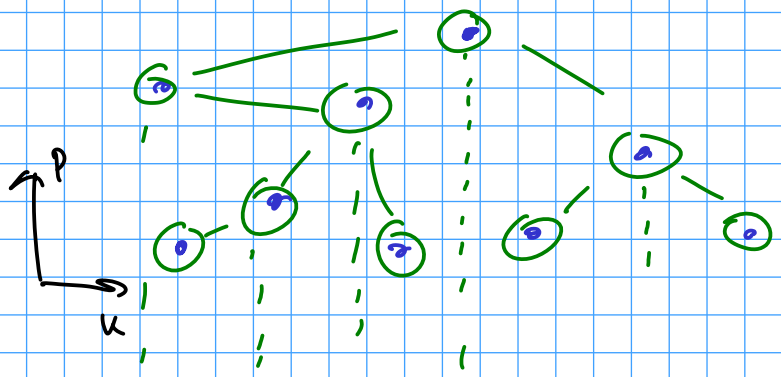


Construct CT $((k_1,p_1)\dots(k_n,p_n))$
return Construct $(1, n)$

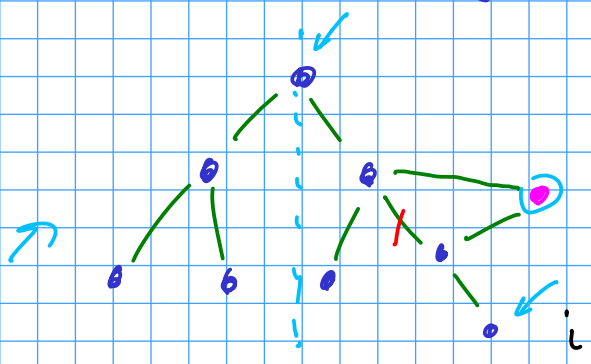
```
Construct (i, j):
    k ← argmax p_ℓ
        i≤ℓ≤j
    n ← node(k_k, p_k)
    n.left = Construct(i, k-1)
    n.right = Construct(k+1, j)
    return n
```

$O(n^2)$

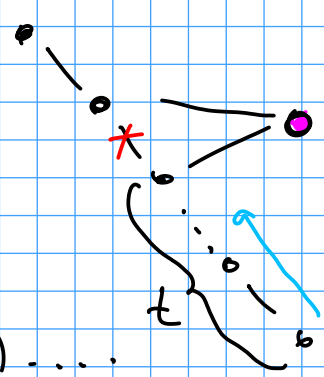
$\Omega(n \log n)$



Построение за $O(n)$



$k_i \ge k_{i-1} \ge \dots \ge k_1$



$(1,1), (2,2), \dots$

$(1,n), (2,n-1), (3,n-2) \dots$

$\Phi(T_i)$ = длина правого пути.

$$c'(i) = c(i) + \Phi(i) - \Phi(i-1) = O(1)$$

$t+1$ $\quad F-t+1 \quad F$

1

$$\sum_i c'(i) = \sum c(i) + \Phi(n) - \Phi(0)$$

$\Phi(0) = 0$, $\Phi(n) \geq 0$

$$\sum c(i) = \sum c'(i) - \Phi(n)$$

$$\sum c(i) \leq \sum c'(i) = O(n)$$

$k_1, k_2, \ldots, k_n \longrightarrow (k_1, z_1), (k_2, z_2), \ldots, (k_n, z_n)$

$z_i \leftarrow U$

↓

Декартово дерево

Treap (Tree + heap)

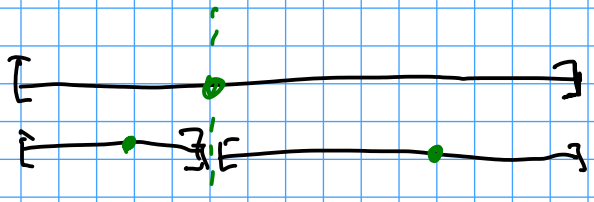
Дуга (Дерево + куча)

Почему сбалансировано?

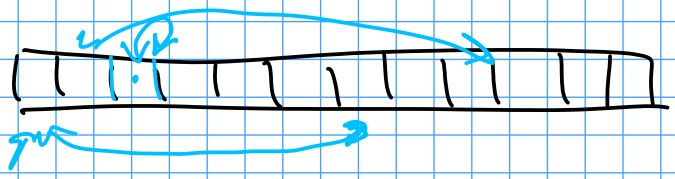
Похоже на QuickSort.

Для QuickSort $E(h(T)) = O(\log n) \Rightarrow$

$\Rightarrow$ для дуги $E(h(T)) = O(\log n)$



NB: Декартово дерево с разными приоритетами определяется однозначно.



Rand(0, n) → U[0,n] ....

Shuffle:

$\frac{1}{n!}$ $\qquad$ $\frac{1}{n} \cdot \frac{1}{n-1} \cdot \ldots \cdot \frac{1}{1} = \frac{1}{n!}$

Search - как и для ∀ дерева поиска.

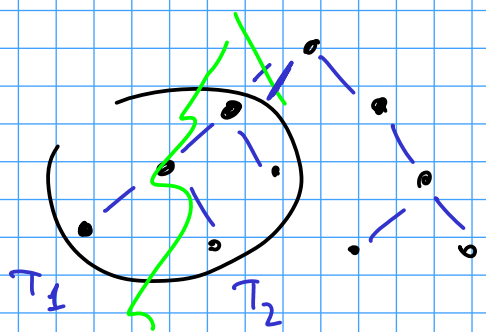
Insert

Remove

Split (T, x)

$T \to T_1, T_2$

$< x \qquad \geq x$



Split (T, x):

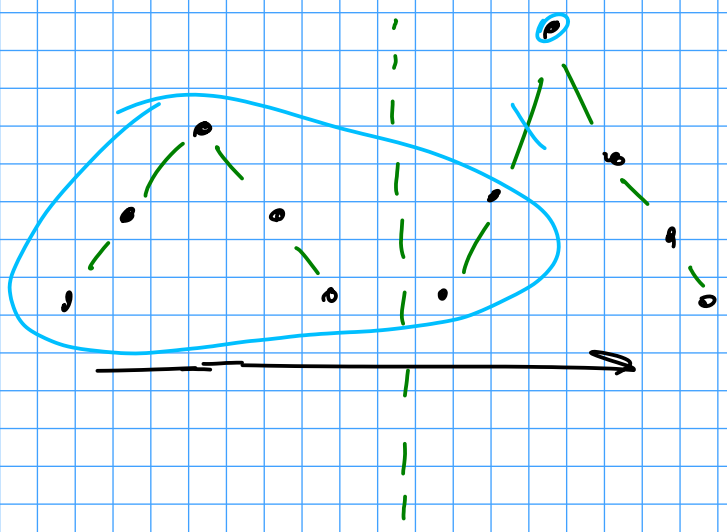
if (x < key(root(T)))

$T_1, T_2 \leftarrow$ Split (left(T), x)

T.left = T_2

return (T_1, T)

Merge (T_1, T_2)



Merge (T_1, T_2)

if (pri(T_1) < pri(T_2))

T_2.left =

Merge(T_1, T_2.left)

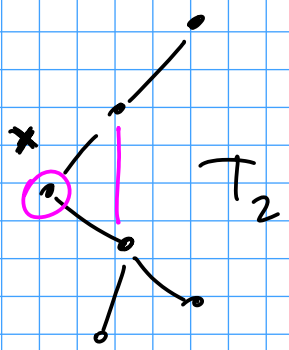
return T_2

else ...

$O(h(T))$ $O(h(T_1)+h(T_2))$

Remove (T, x)

T_1, T_2 = Split (T, x)
T_2 = RemoveLeft (T_2)
return Merge (T_1, T_2)



Insert (T, x)

T_1, T_2 = Split (T, x)
T_3 = node $(x$, rand $(\ldots))$
return Merge $(T_1$, Merge $(T_3, T_2))$