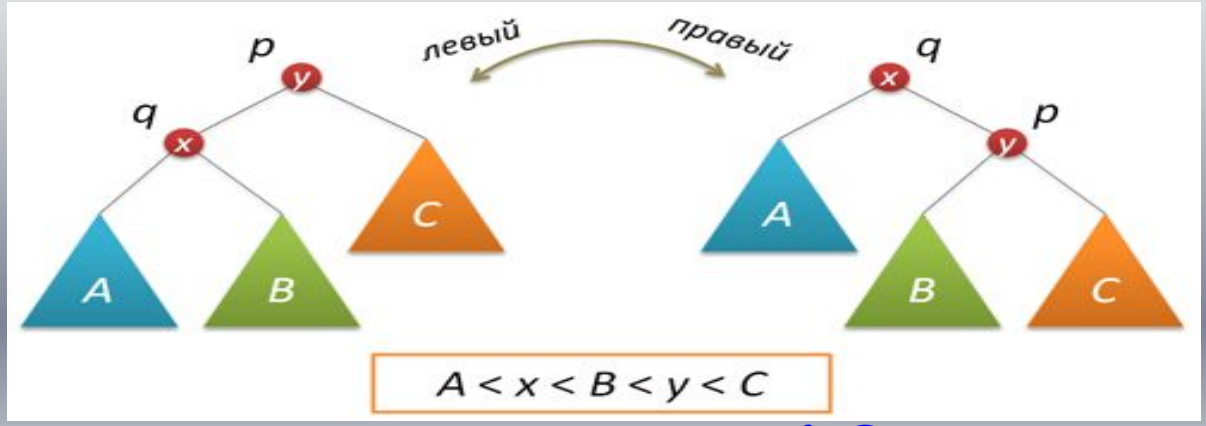
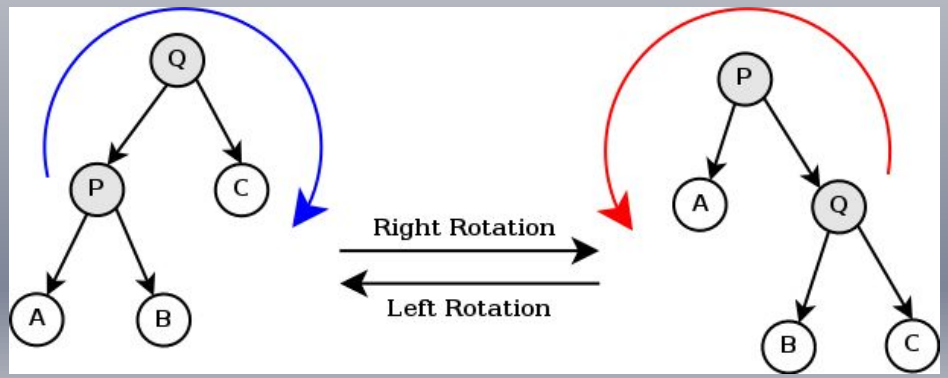
Алгоритмы и структуры данных



Лекция 10

Часть 1.

Рандомизированные пирамиды поиска



Рандомизированные пирамиды поиска

Пирамиды

Бинарное дерево обладает свойством *пирамидальности* или, выражаясь короче, является *пирамидой* (Heap),

если **у каждого узла дерева нет сыновей с ключами, большими чем ключ в этом узле**.

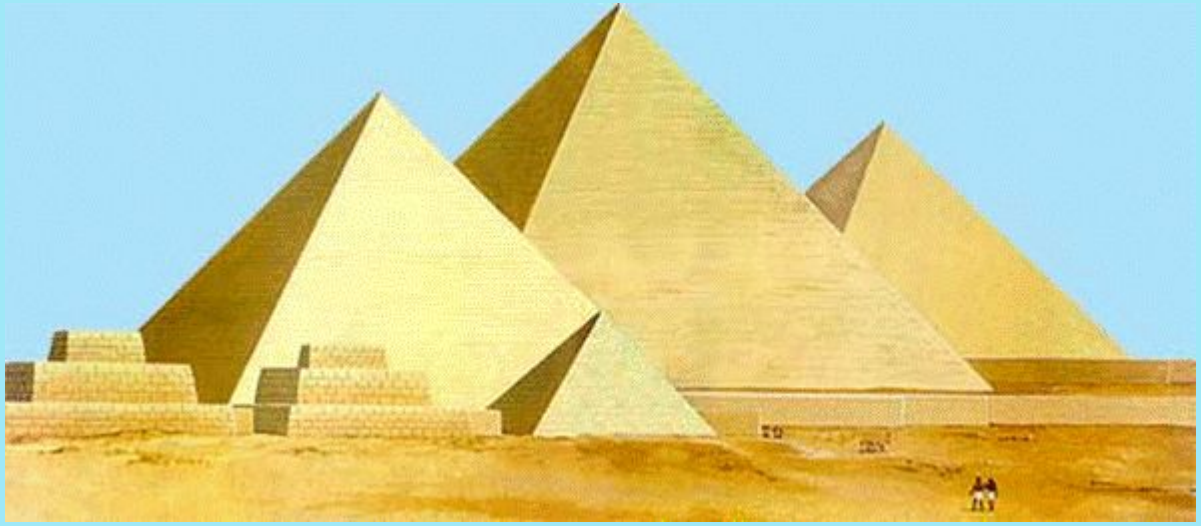
Пирамиды

Инвариант пирамиды H:

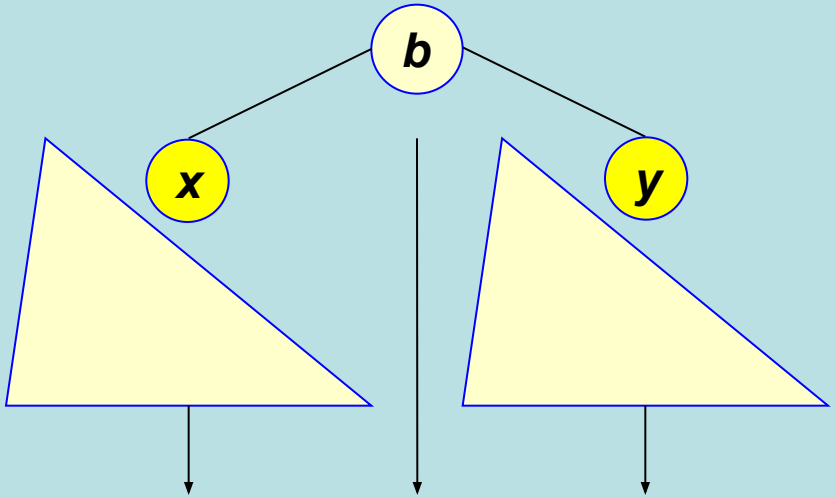
$(\forall\ b \in H$:

$((\textbf{not}\ \text{Null}(\text{Left}(b)) \rightarrow (\text{Root}(\text{Left}(b)).\text{key} \leq \text{Root}(b).\text{key}\ )\ \textbf{\&}$

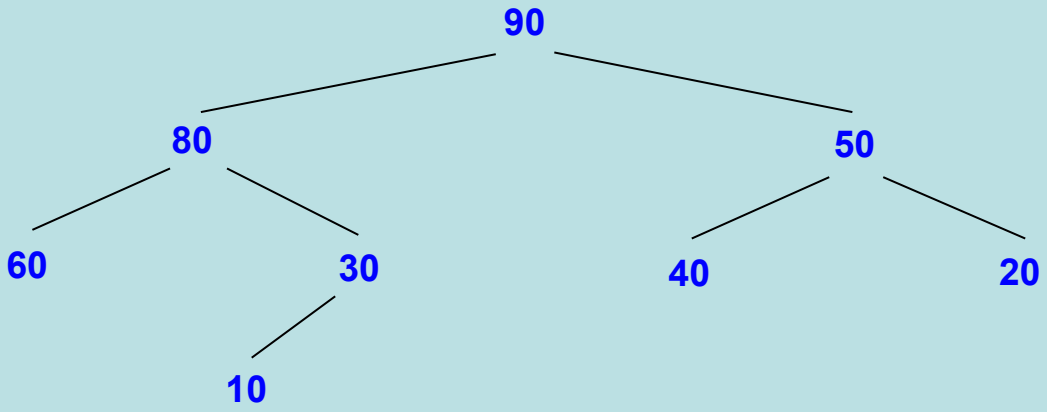
$((\textbf{not}\ \text{Null}(\text{Right}(b)) \rightarrow (\text{Root}(\text{Right}(b)).\text{key} \leq \text{Root}(b).\text{key}\ ))$



Пирамида



$k(b) \geq \max(k(x), k(y))$



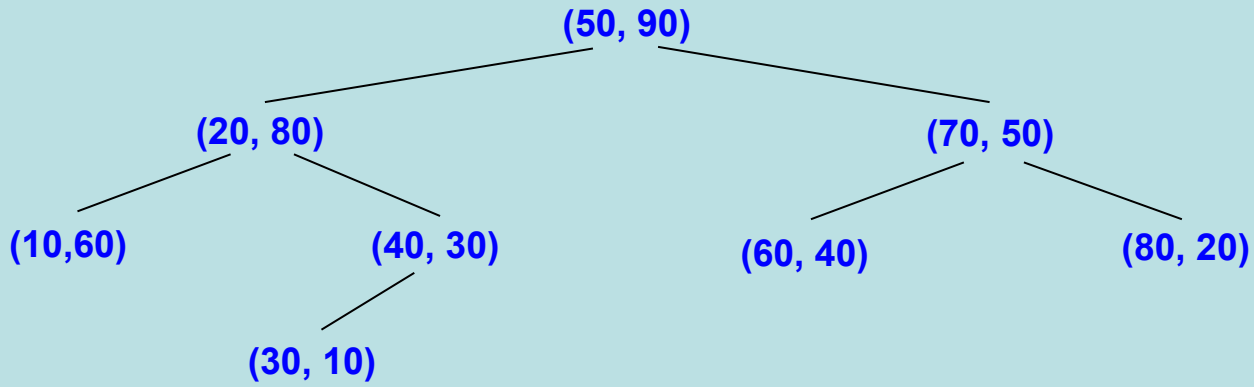
Бинарное дерево, обладающее свойством пирамидальности

1

Пирамиды поиска (Treaps - *Дерамиды*)

Treap = Tree + Heap

Определение. Пусть каждому узлу бинарного дерева T приписана пара (k_i, p_i), где k_i – *ключ*, а p_i – *приоритет* (ключи и приоритеты *порядкового* типа). **Пирамида поиска** – это бинарное дерево, которое является *деревом поиска по ключам и пирамидой по приоритетам.*



Пирамида поиска

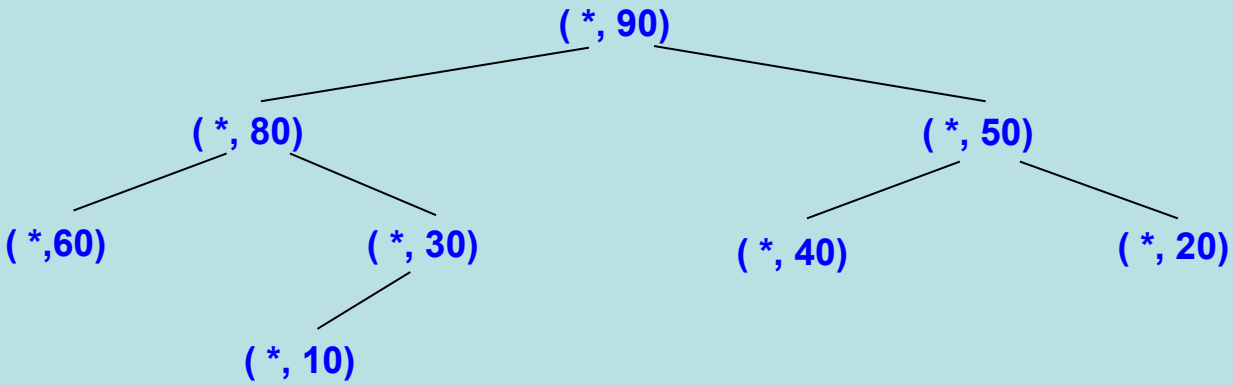
S = {(10, 60), (20, 80), (30, 10), (40, 30), (50, 90), (60, 40), (70, 50), (80, 20)}

2

Пирамиды поиска (Treaps - *Дерамиды*)

Treap = Tree + Heap

Определение. Пусть каждому узлу бинарного дерева T приписана пара (k_i, p_i), где k_i – *ключ*, а p_i – *приоритет* (ключи и приоритеты *порядкового* типа). **Пирамида поиска** – это бинарное дерево, которое является ***деревом поиска по ключам и пирамидой по приоритетам.***



Пирамида поиска

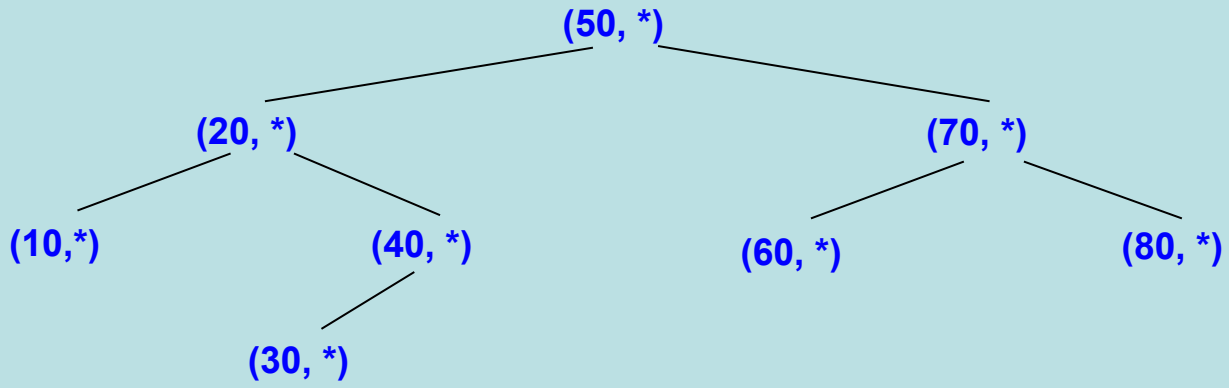
S = {(10, 60), (20, 80), (30, 10), (40, 30), (50, 90), (60, 40), (70, 50), (80, 20)}

3

Пирамиды поиска (Treaps - *Дерамиды*)

Treap = Tree + Heap

Определение. Пусть каждому узлу бинарного дерева T приписана пара (k_i, p_i), где k_i – *ключ*, а p_i – *приоритет* (ключи и приоритеты *порядкового* типа). **Пирамида поиска** – это бинарное дерево, которое является ***деревом поиска по ключам и пирамидой по приоритетам.***



Пирамида поиска

S = {(10, 60), (20, 80), (30, 10), (40, 30), (50, 90), (60, 40), (70, 50), (80, 20)}

Утверждение. Если среди значений ключей, а также среди значений приоритетов в множестве $S = \{(k_1, p_1), (k_2, p_2), \ldots, (k_n, p_n)\}$ нет совпадающих, то *пирамида поиска* на множестве S определена *единственным образом*.

Доказательство. При $n = 0$ и $n = 1$ утверждение справедливо. При $n > 1$ выберем

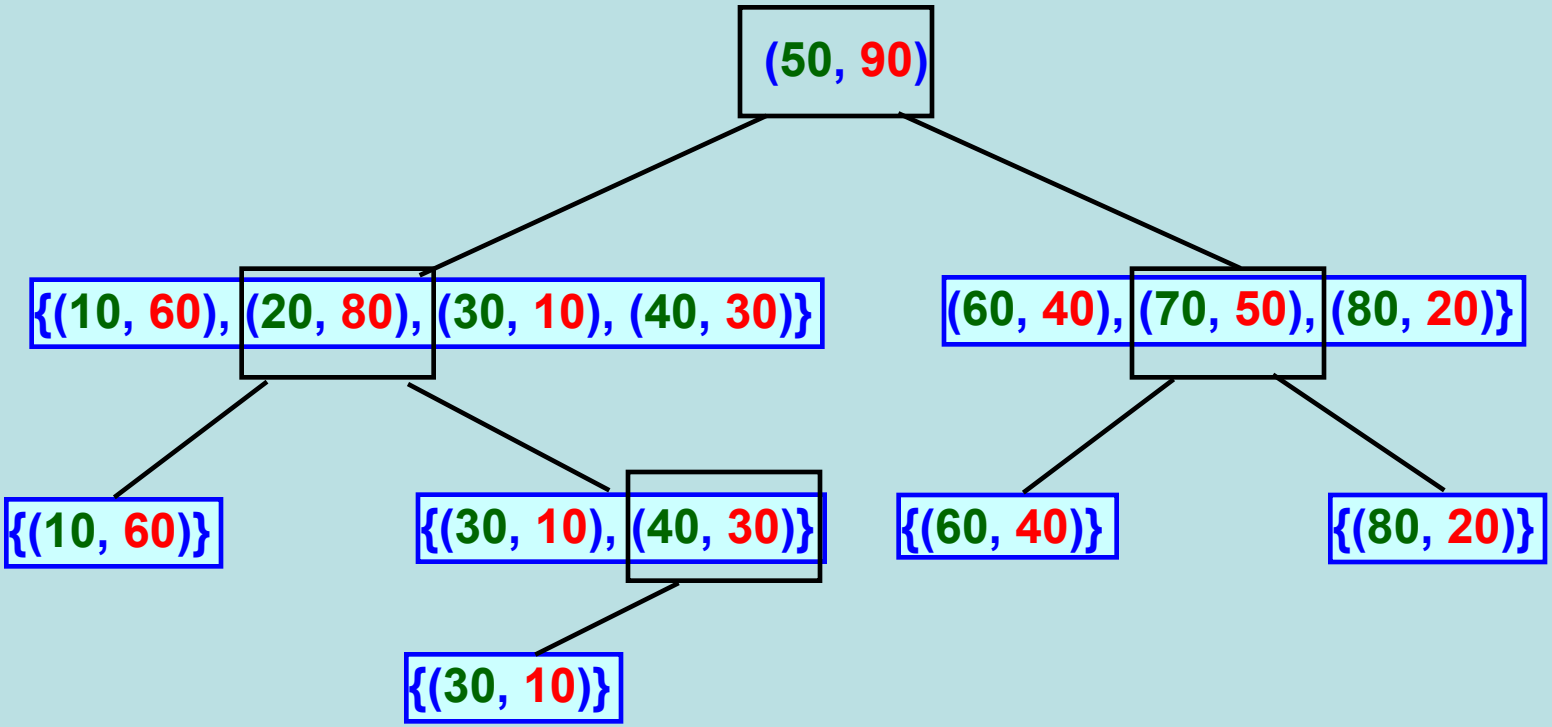
$$p_j = \max_{i \in 1..n} p_i$$

Для того чтобы выполнить свойство пирамиды, в качестве корня дерева возьмем узел (k_j, p_j). Выделим из множества S два подмножества: S_1 с ключами, меньшими чем k_j, и S_2 с ключами, большими чем k_j .

Из элементов множества S_1 построим, действуя аналогичным образом, левое поддерево корня (k_j, p_j), а из элементов множества S_2 – правое поддерево. Далее – по индукции.

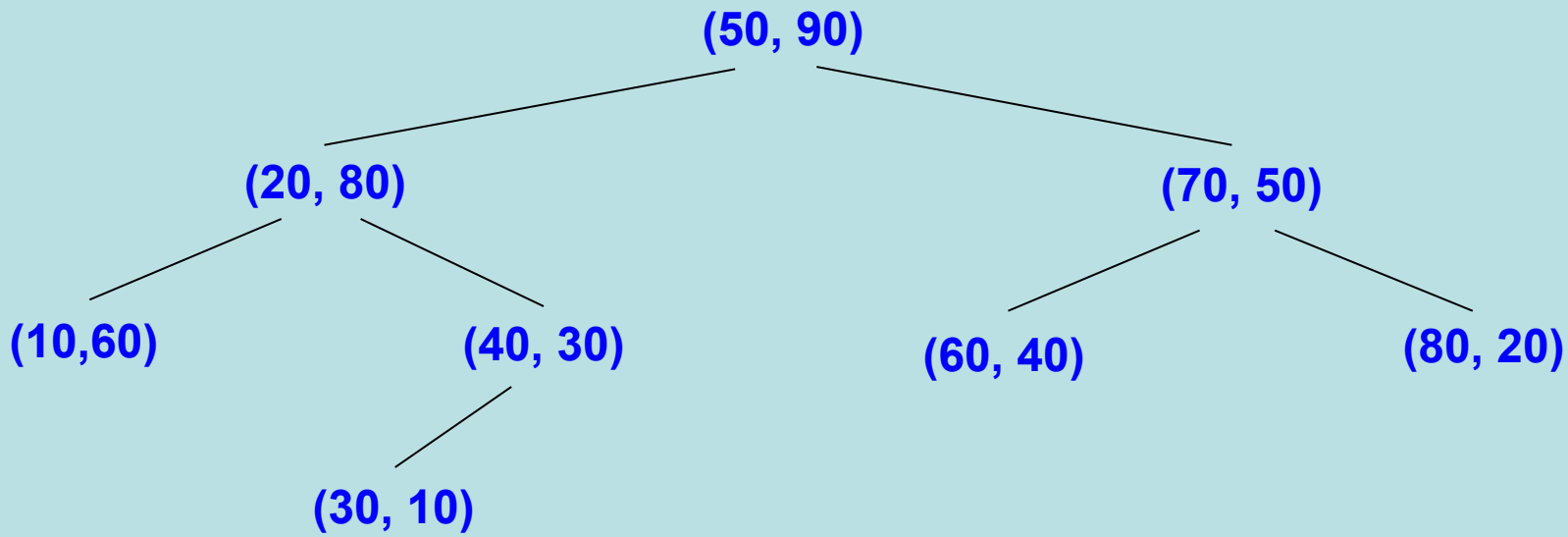
Пример построения пирамиды поиска о заданному набору ключей и приоритетов

S = {(10, 60), (20, 80), (30, 10), (40, 30), (50, 90), (60, 40), (70, 50), (80, 20)}



Пример построения пирамиды поиска по заданному набору ключей и приоритетов

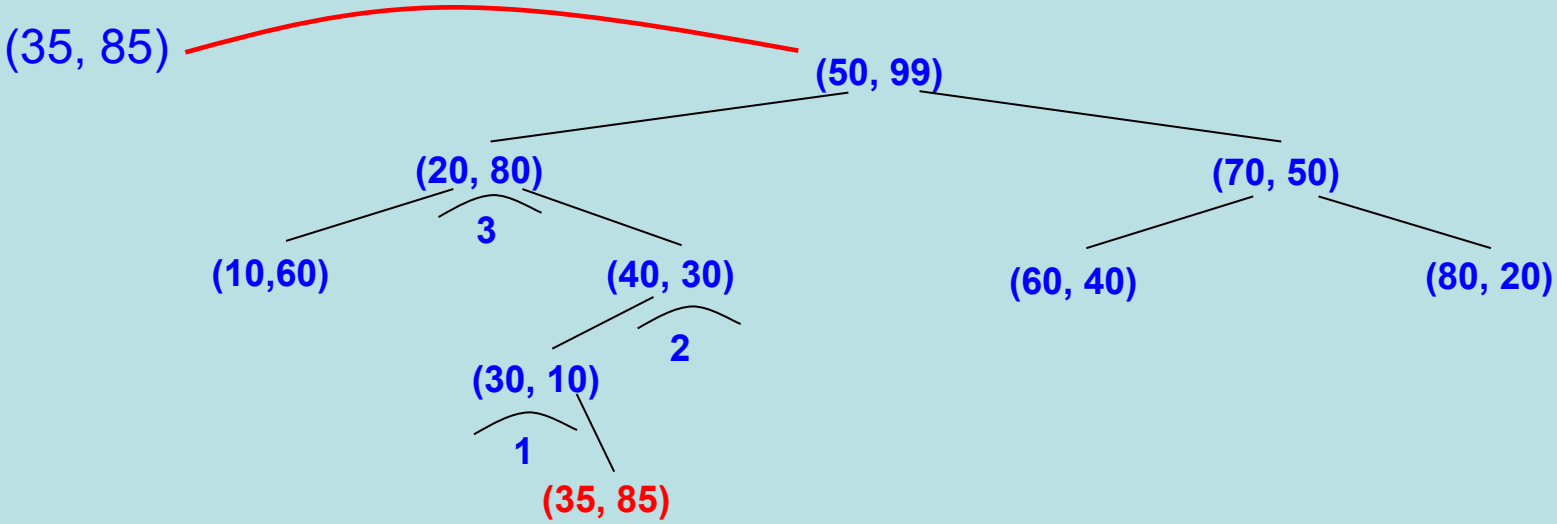
Результат



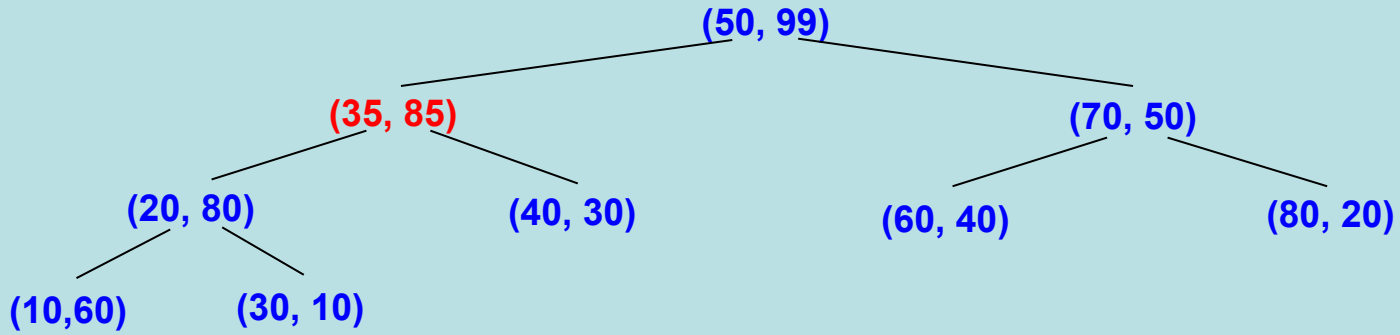
Пирамида поиска

Добавление узла *(k, p)* в пирамиду поиска:

1. Узел *(k, p)* вставляется в *дерево поиска по ключу k* (как лист дерева);
   - Если необходимо, то с помощью *вращений* восстанавливается свойство *пирамиды по приоритету p*.



Пример вставки узла в пирамиду поиска: первый шаг вставка в дерево поиска



Пример вставки узла в пирамиду поиска: результат вращений

Рандомизированная пирамида поиска (*Random Treap*)

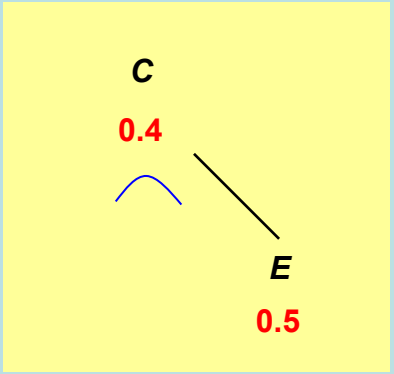
Рандомизированной пирамидой поиска называют пирамиду поиска, которая получена *последовательной* вставкой входной последовательности ключей (подобно случайному БДП) таким образом, что *приоритеты* при каждой вставке разыгрываются *случайно* и являются случайной величиной, распределенной равномерно, например, на интервале вещественных чисел [0,1].

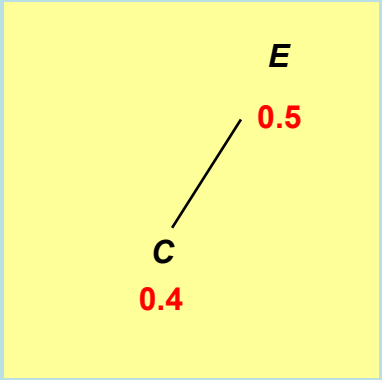
Добавление узла *k* в рандомизированную пирамиду поиска:

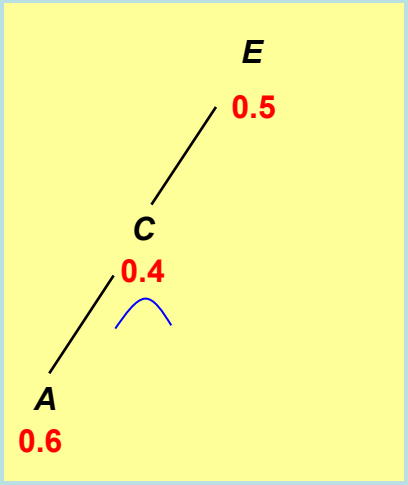
1. Узел вставляется в *дерево поиска по ключу k* (как лист дерева);
2. Случайно разыгрывается значение приоритета *p*.

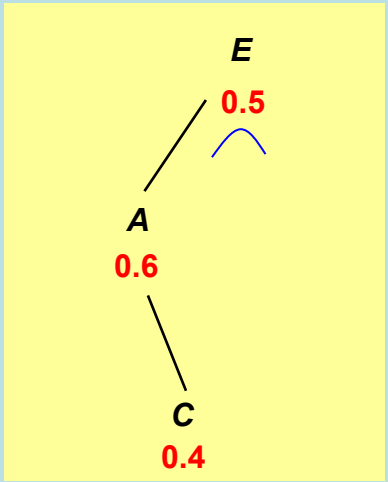
- Если необходимо, то с помощью *вращений* восстанавливается свойство *пирамиды*, начиная с узла (*k*,*p*).

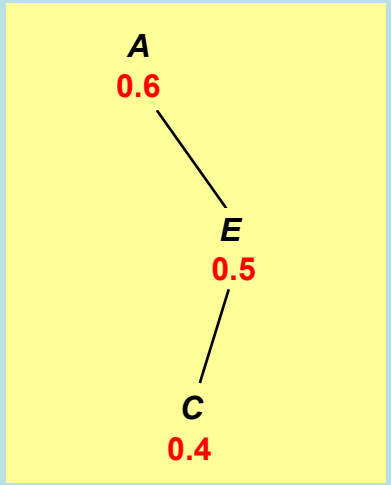
Построение рандомизированной пирамиды поиска по входной последовательности CEAFBDG

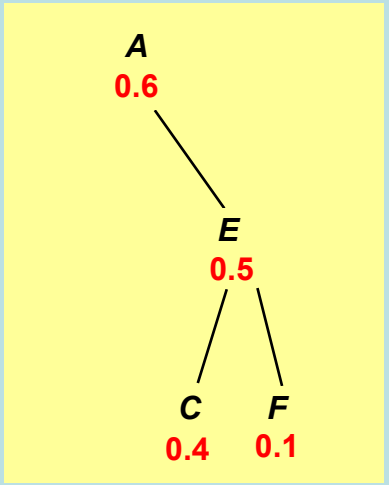




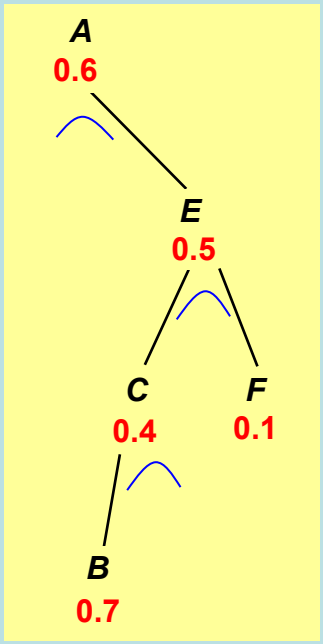


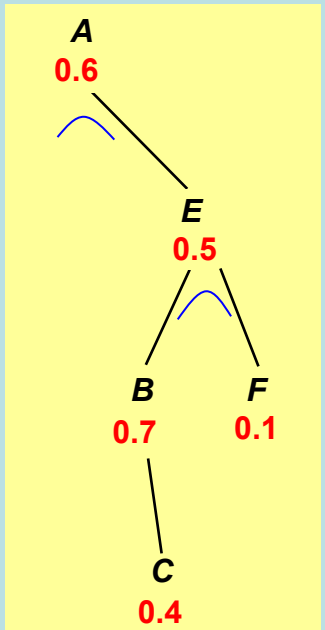


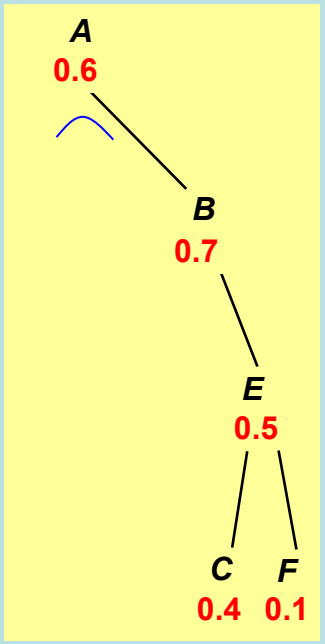


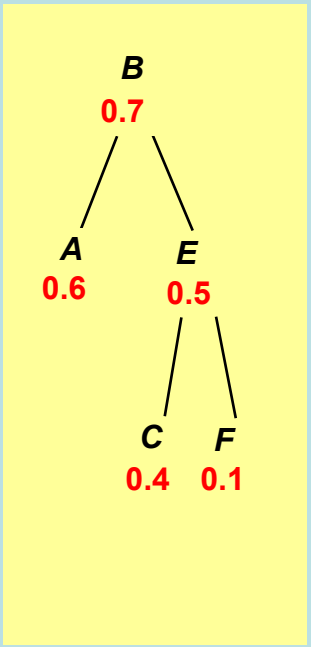


Построение рандомизированной пирамиды поиска по входной последовательности CEAFBDG

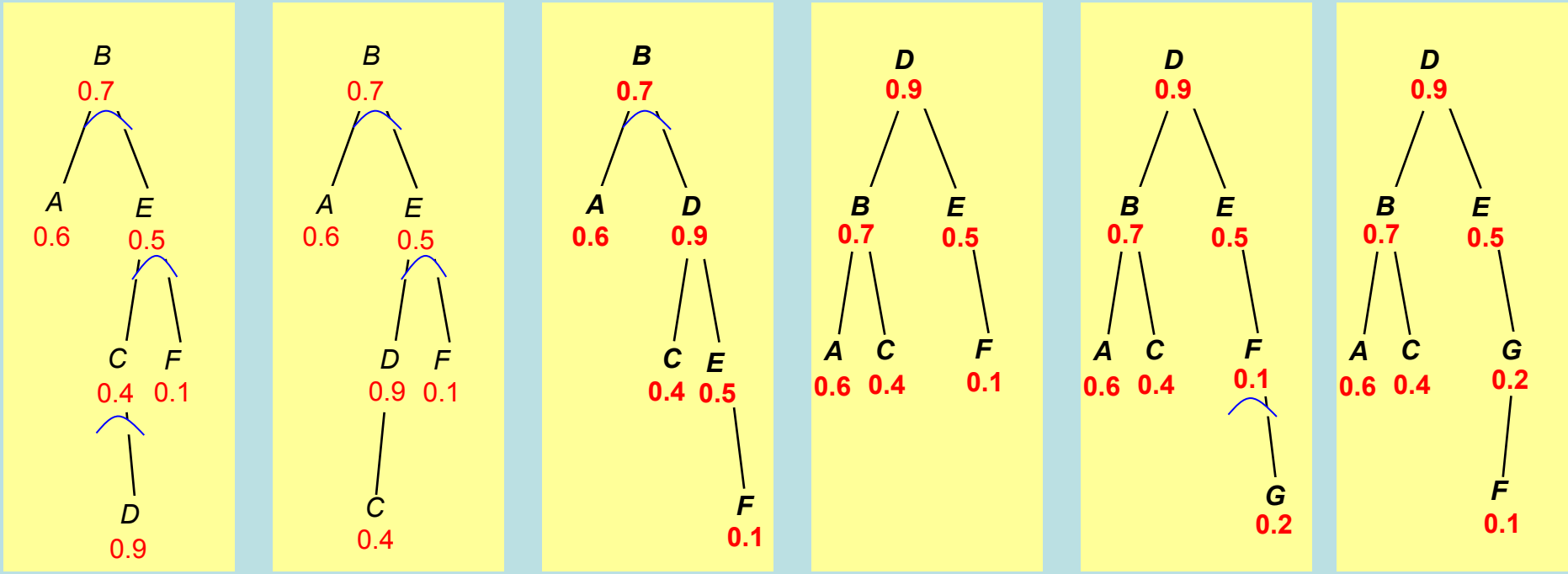








Построение рандомизированной пирамиды поиска по входной последовательности *CEAFBDG*



Рандомизированные пирамиды поиска
Treaps (Дерамиды)

- Treaps имеют среднюю высоту не более $2.99 \log_2 n$
- Операции *поиска, вставки и удаления* в Treaps выполняются в среднем за время $O(\log_2 n)$, т.е. не более, чем в постоянное число раз превышающее $\log_2 n$ *при достаточно большом n*
- Средняя высота Treaps несколько больше, чем в случайных деревьях, однако при этом вероятность появления «плохих» деревьев очень мала.

R. Seidel, C. R. Aragon. "Randomized Search Trees".
http://citeseer.nj.nec.com/cachedpage/364925/1
http://goanna.cs.rmit.edu.au/~e76763/pub/sa96-a.pdf

См. также Т.Кормен, Ч.Лейзерсон, Р.Ривест, К.Штайн
Алгоритмы: Построение и анализ, 2-е издание
Задача 13-4. *Дерамиды*

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ

КОНЕЦ ЛЕКЦИИ