Cartesian tree
Theory and applications

Gleb Evstropov
glebshp@yandex.ru

November, 14, 2016

1 Some notations

- u, v, w — some nodes of the binary search tree;
- $parent(v)$ — the parent of some node v in the binary search tree. If v is the root then $parent(v) = NIL$;
- $left(v)$ — left child of some node v in the binary search tree. If the left subtree is empty, then $left(v) = NIL$;
- $right(v)$ — right child of some node v in the binary search tree. If the right subtree is empty, then $right(v) = NIL$;
- $key(v)$ — the value of a node v that affects the tree structure;
- $x(v)$ — another way to denote keys in Cartesian trees. Usually, $x(v) = key(v)$.
- $y(v)$ — some additional value associated with the node v and used to build the tree;
- $subtree(v)$ — the set of all nodes that lie inside the subtree of some node v (v is also included);
- $size(v)$ — the size of the subtree of some node v;
- $x_l(v)$ — the minimum key in the subtree of the node v, that is:
$$x_l(v) = \min_{u \in subtree(v)} key(u)$$
- Same as $x_l(v)$ we define $x_r(v)$ as the maximum key in the subtree of the node v:
$$x_r(v) = \max_{u \in subtree(v)} key(u)$$
- $depth(v)$ is the length of the path from $root$ to v. $depth(root) = 0$.
- $height(v)$ is the difference $max(depth(u)) - depth(v)$, where $u \in subtree(v)$.

2 Key points and definitions

- Greedy algorithm of finding an increasing subsequence: take first element that is greater than current, "left ladder". The expected length of the result on a random permutation is $O(\log n)$.

- BST stands for *binary search tree*, that is a binary rooted tree with some keys associated with every node, and the following two conditions hold:

$$key(u) < key(v), \forall u, v : u \in subtree(left(v))$$

  and

$$key(u) > key(v), \forall u, v : u \in subtree(right(v))$$

- For any pair of nodes of any binary search tree v and u:
  $u \in subtree(v)$ if and only if $x_l(v) \le key(u) \le x_r(v)$

- For any tree and some keys stored in nodes of that tree we say that *heap condition* holds if for any v that is not the root:

$$key(parent(v)) \ge key(v)$$

- Binary search tree of size n is balanced if it's height is $O(logn)$.

- *Cartesian tree* or *treap* is a balanced binary search tree, where each node is assigned some random values $y(v)$, which satisfy to the heap condition. Hereafter we will treat $y(v)$ as a random permutation.

- Cartesian tree is uniquely determined by a set of pairs (x_i, y_i), such that all x_i are pairwise distinct and all y_i are pairwise distinct.

- Node v is an ancestor of a node u if and only if for every $w \neq v$ such that $min(key(v), key(u)) \le key(w) \le max(key(v), key(u))$ it's y is smaller than the y of v, i.e. $y(v) > y(w)$.

- Linear algorithm to build Cartesian tree having a sorted pairs using stack.

- The expected depth of an i-th node (in the order of left-right traversal) is

$$\sum_{j=0}^{j<n} \frac{1}{|j-i|+1} \le 2 \cdot \sum_{j=1}^{j\le n} \frac{1}{j} = O(\log n)$$

  .

- We can treat a Cartesian tree as an array, if we replace $x(v)$ with it's relative position on the tree. The data structure is called *Implicit-key Cartesian tree.*

- Persistent Cartesian tree cannot use fixed random values $y(v)$, instead, two subtrees are merge with probability proportional to their sizes.

3 Some problems to think about

- Prove that if in merge operation one picks the root equiprobable (i.e. both l and r has probability 0.5 to become the root), the expected height is $\frac{n}{2}$.
- Prove that if in merge operation one picks the root proportional to the height (i.e. l has probability $\frac{height(l)}{height(l)+height(r)}$ to become the root), the expected height is $\sqrt{n}$.
- Prove that it's impossible to build Cartesian tree in linear time if the set of keys is not-sorted.
- Prove that it's impossible to merge two Cartesian trees of size n in $o(n)$ time, if there are no guarantees on key ranges.
- What is the expected height of the tree if we pick y's as integers in range from 0 to p? Consider the case $p = 1$ first.
- Can you prove $O(\log n)$ expectation for $height(root)$?