Персистентность

Есть структура данных T и операции. Некоторые операции изменяют T, а некоторые не изменяют. Тогда можно говорить о версиях структуры T в разные моменты времени. *Персистенность* - это свойство структуры данных делать запросы к старым версиям.

Уровни персистентности:

- $parital(weak)$ - версии образуют цепочку, то есть каждая следующая версия - это измененная предыдущая (запросы изменения можно делать только к последней версии).

- $full(strong)$ - версии образуют дерево (запросы изменения можно делать к любой версии).

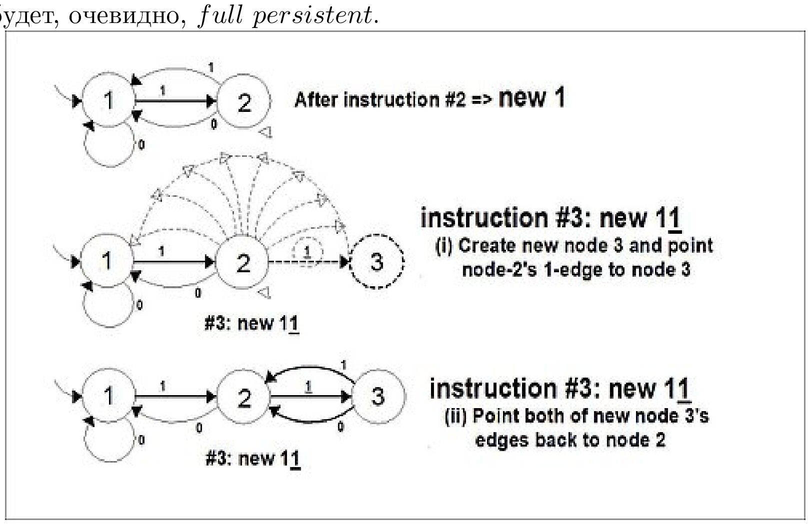
- $fuctional(confluent)$ - поддерживаются операции сразу для нескольких версий (допустим $merge$ для версий декартова дерева).

Персистенный массив.

Для каждой клеточки храним вектор пар (t_x, v_x) - в момент t_x мы изменили этот элемент массива на v_x (храним в порядке возрастания t_x). Бинпоиском отвечаем на запрос *узнать значение элемента версии* t.

Такая персистентность $partial$, но не $full$ и $functional$.

Pointer machine. Способ организовать структуру данных. Будем хранить все в узловой структуре, поддерживая связи между ними с помощью указателей. Тогда вместо изменения вершины мы просто создаем новую. Таким образом, чтобы изменить вершину x, нужно изменить суммарно $O(h)$ вершин. Такая структура данных будет, очевидно, $full\ persistent$.



Малополезная картинка

Full персистентный массив. Давайте создадим двоичное дерево над массивом размера n с высотой $O(\log n)$, реализовав его как pointer machine. Тогда теперь мы можем сделать изменение произвольного элемента в произвольной версии, получив $O(\log n)$ времени работы (и очевидную реализацию персистентного ДО в придачу).

Персистентное декартово дерево.

Заметим, что наше декартово дерево можно было бы реализовать через pointer machine, что даст нам что-то очень похожее на то, что мы хотим от такой структуры данных. К сожалению, такая структура данных не будет $functional\ persistent$, потому что есть конструктивный способ очень сильно расширить дерево в высоту (достаточно просто мерджить одну и ту же вершину с последней версией дерева, получая бамбук).

Проблема у нас возникла в тот момент, когда мы воспользовалиь старой идеей приоритетов. Теперь будем вычислять что-то типа приоритетов динамически. А именно, будем считать, что приоритет дерева L больше приоритета дерева R, если $rand() < \frac{S(L)}{S(L)+S(R)}$. Можно показать, что теперь высота ДД все еще $O(\log n)$.