Персистентный стек

Булгаков Илья, Гусев Илья

Московский физико-технический институт

Москва, 2020

Содержание

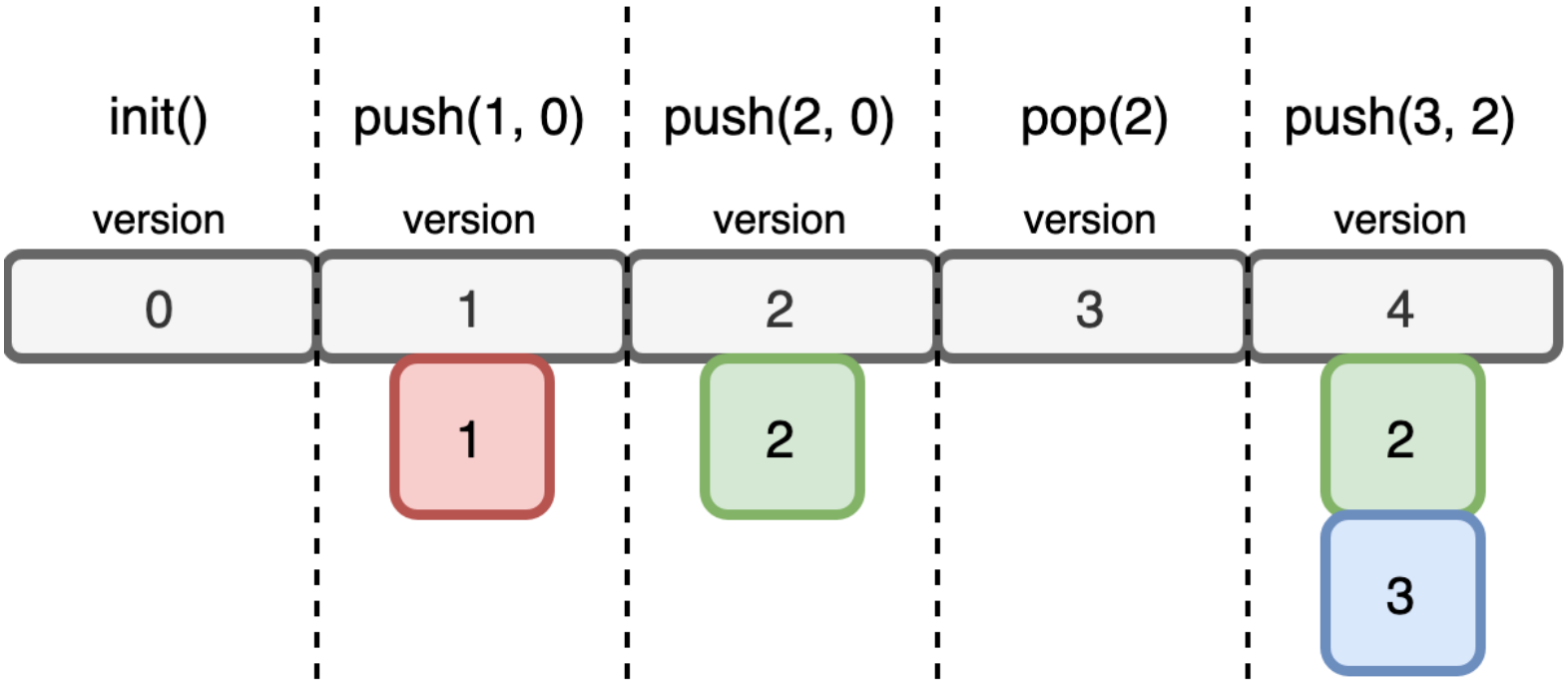
Определение

Персистентная структура данных - такая СД, которая при изменении сохраняет предыдующую версию себя. То есть можно получить её состояние в любой момент времени и изменить из любого момента времени.
Частично персистентная - ПСД с возможностью изменения только последней версии.

Новый интерфейс

- `push(data, version)` - создать новую версию стека, которая получается вставкой элемента `data` в версию `version`
- `pop(version)` - возвращаем данные из `version`, создаём новую версию стека без снятого элемента

Наивная реализация



- $O(n)$ времени на операцию
- $O(n^2)$ памяти на хранение

Персистентный стек

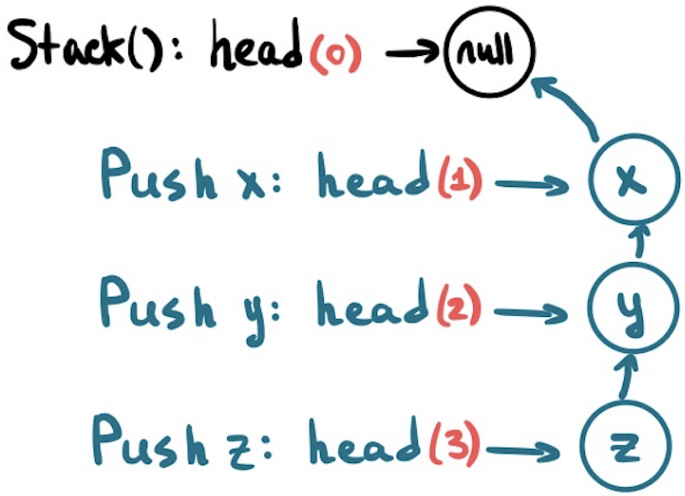
Можно лучше!

Персистентный стек

Создадим стек. Голова стека - указатель на вершину. В вершине храним сам элемент и указатель на предыдущую вершину стека.

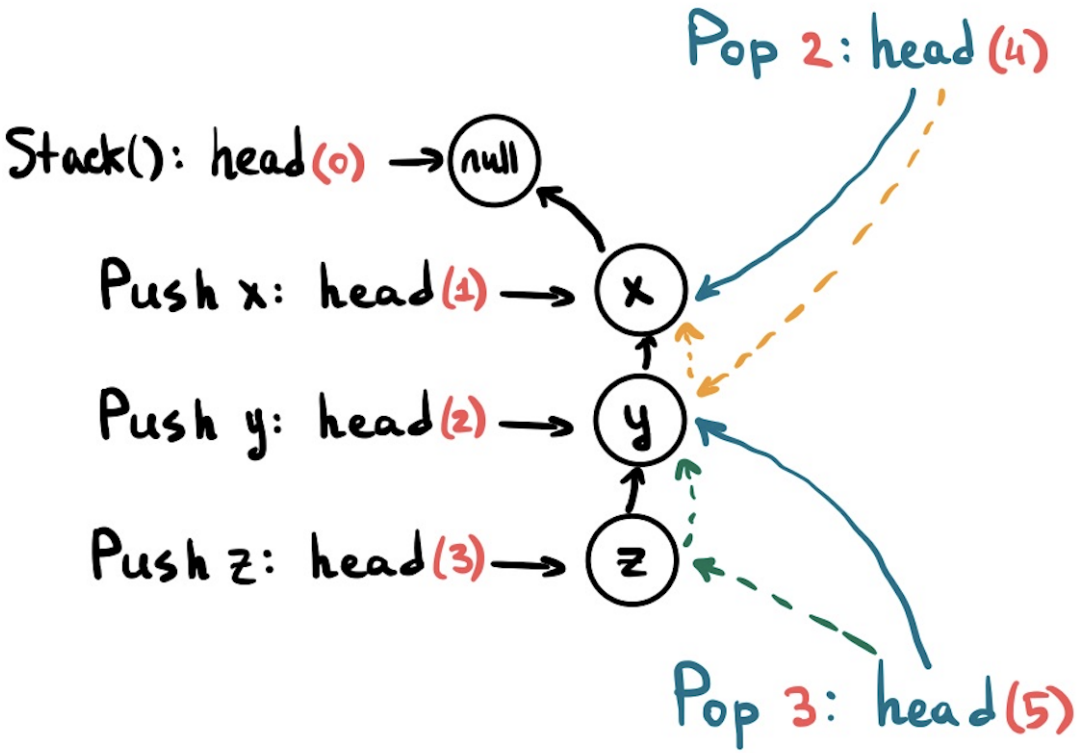
Персистентный стек

Добавим несколько новых элементов в стек. Каждая операция будет возвращать голову стека соответствующую новой версии.



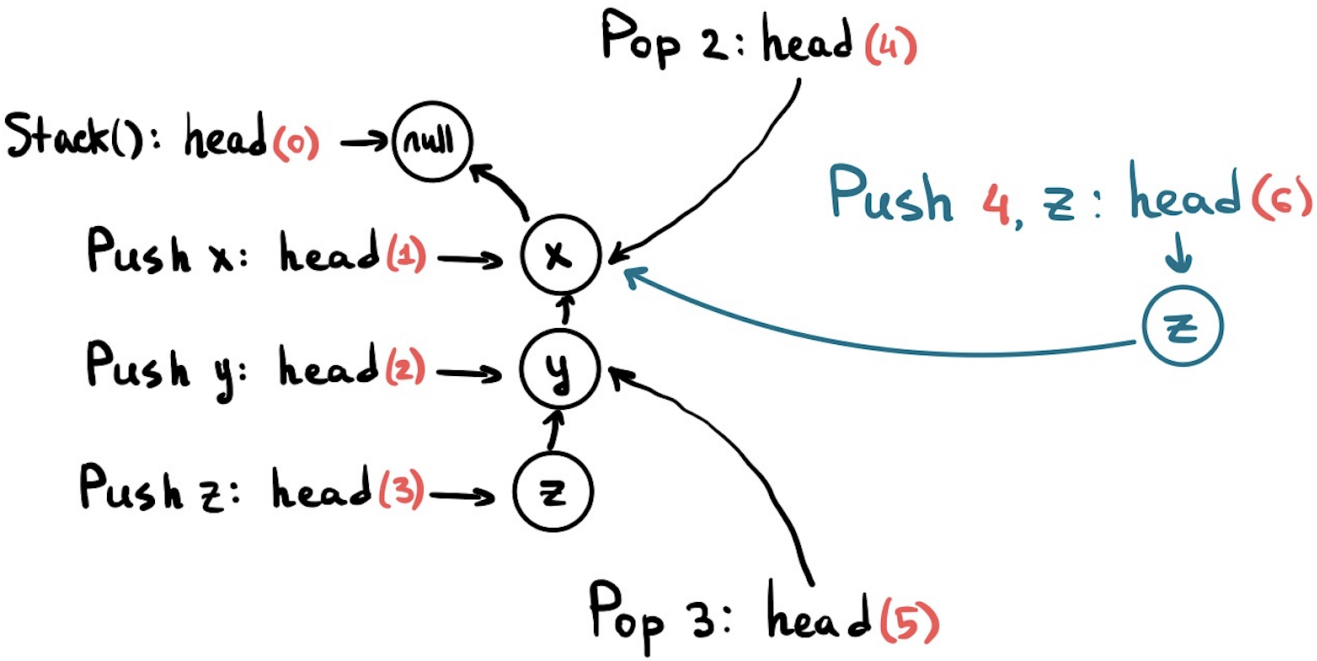
Персистентный стек

Производим `Pop(v)` так: смотрим в голову версии `v` и берем в качестве новой головы указатель на предыдущую вершину стека.



Персистентный стек

Добавляя элемент к версии v, создаем новую вершину и в качестве предыдущей берем голову v.



Персистентный стек

Какая асимптотика по времени и памяти?

Персистентный стек

Какая асимптотика по времени и памяти?

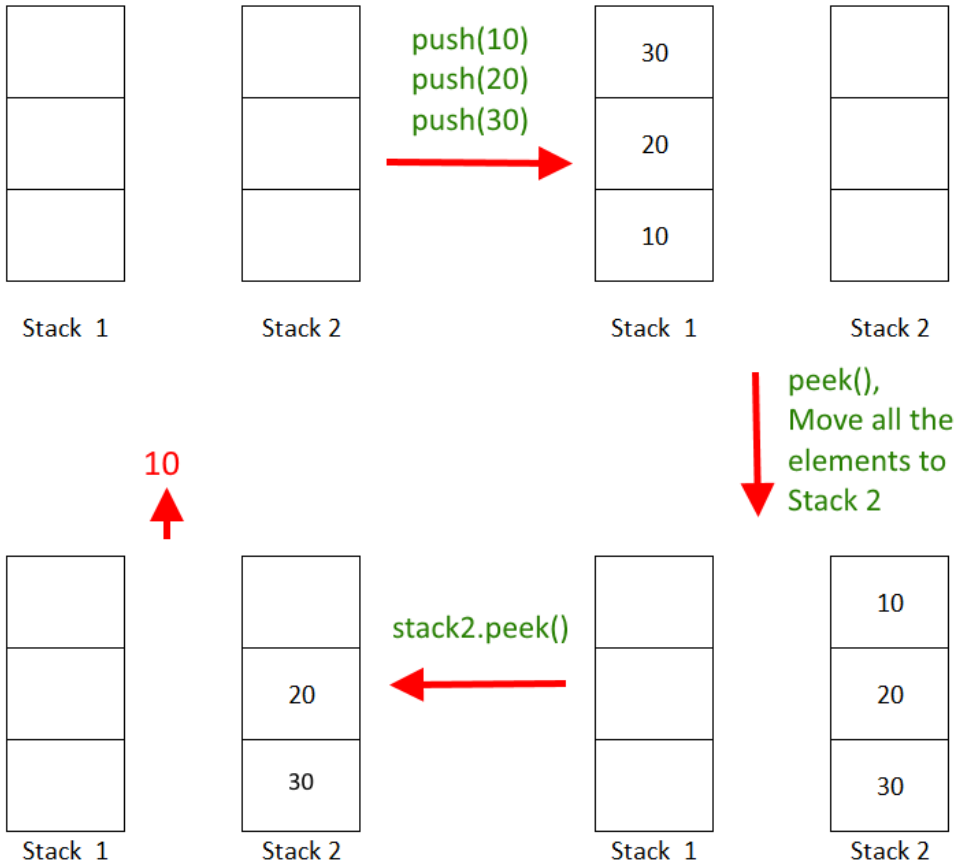
- $O(1)$ времени на операцию
- $O(n)$ памяти на хранение

Персистентная очередь

Как реализовать персистентную очередь? Мб получится так же просто, как стек?

Персистентная очередь

Идея: мы умеем реализовать очередь через 2 стека, а стек мы умеем делать персистентным. Вспомним реализацию очереди на 2х стеках

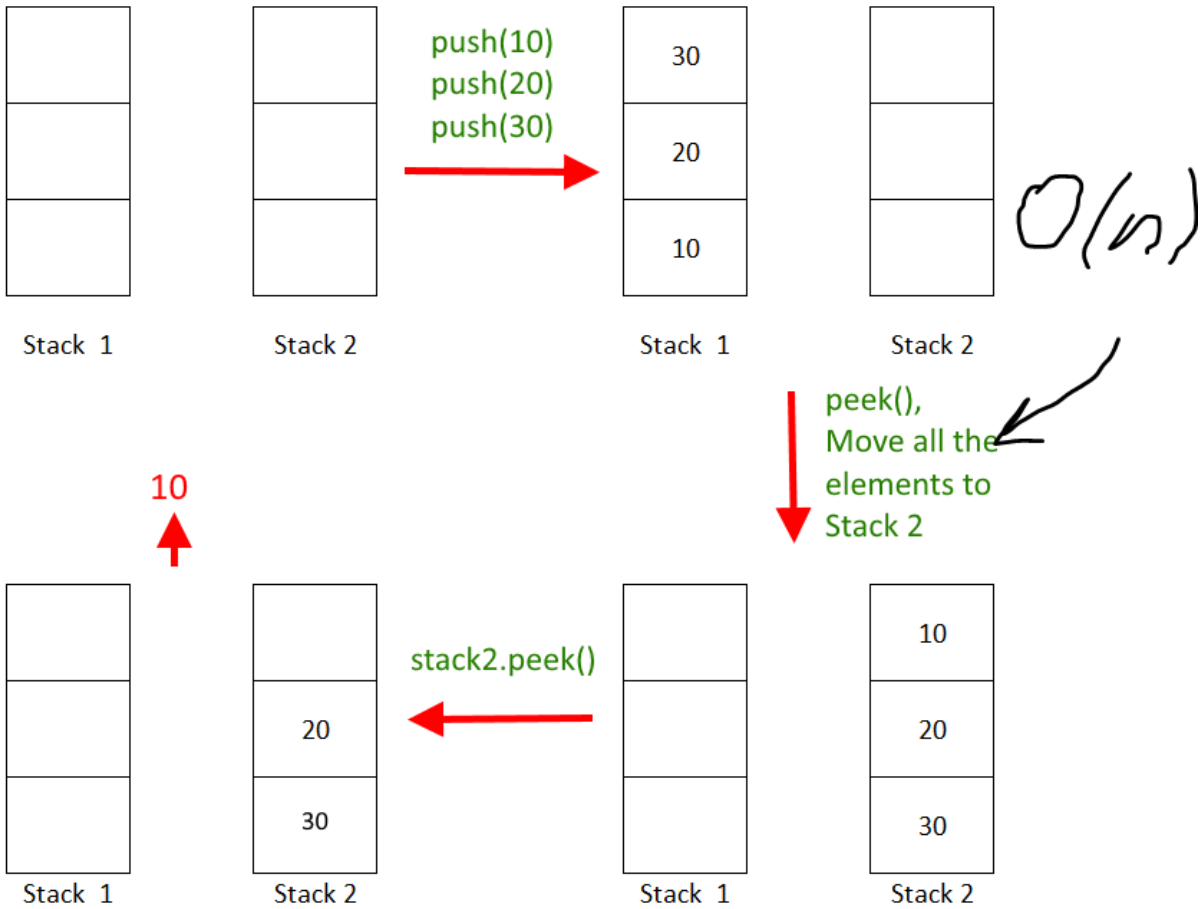


Персистентная очередь

Какая сложность операций в реализации через 2 стека?

Персистентная очередь

В худшем случае операция *pop*() может занять $O(n)$



Персистентная очередь

Поэтому для персистентной реализации через 2 стека не подходит. Как быть?
Идеи:

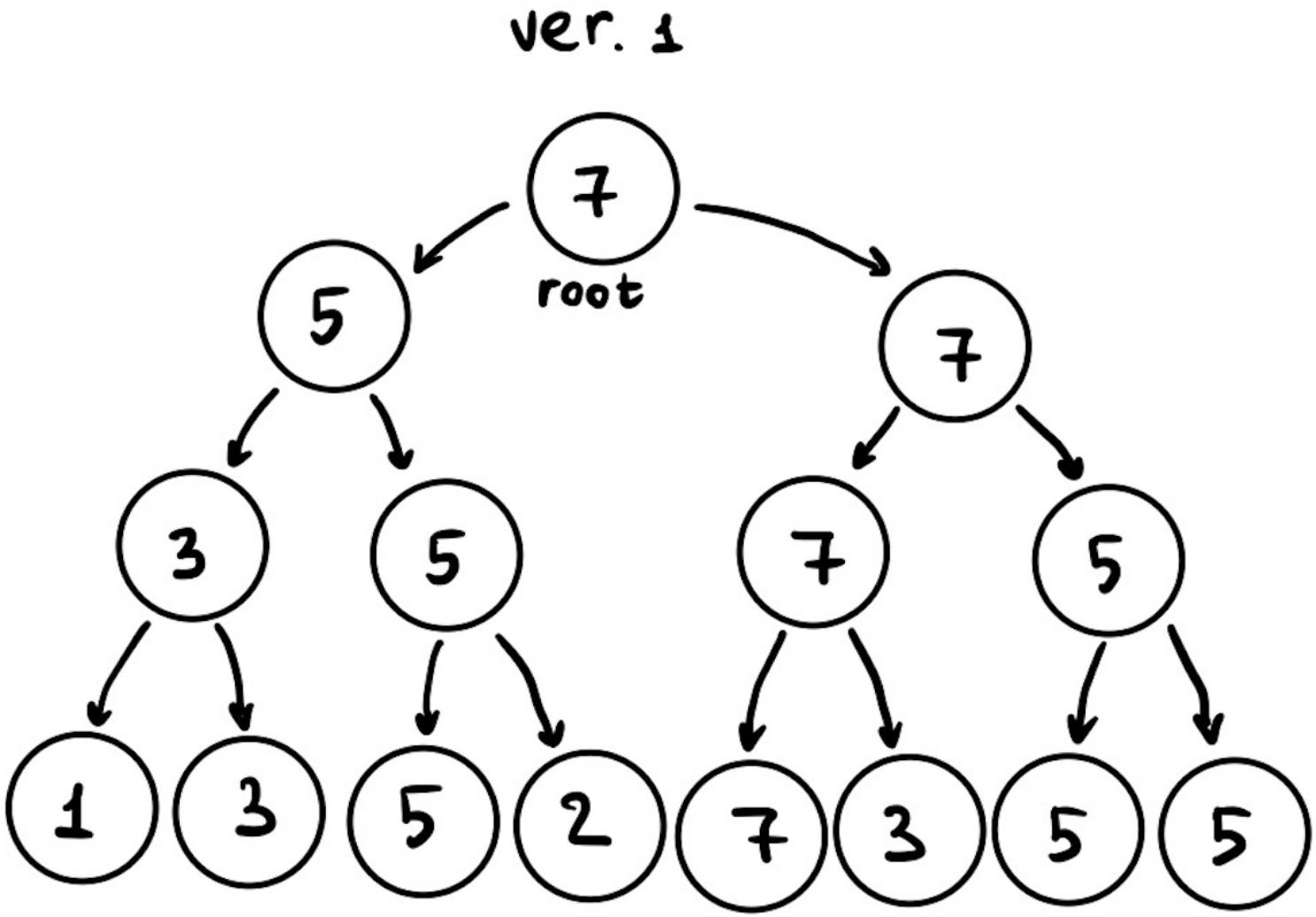
- От стеков совсем не отказываемся
- Больше стеков - лучше
- Попробуем реализовать очередь через стеки с гарантированным O(1) на операцию. Если распределить время, необходимое для перемещения элементов из одного стека в другой, по операциям, мы получим очередь без худших случаев с O(1) истинного времени на операцию.

Персистентная очередь

Алгоритм: очередь на 6 стеках.

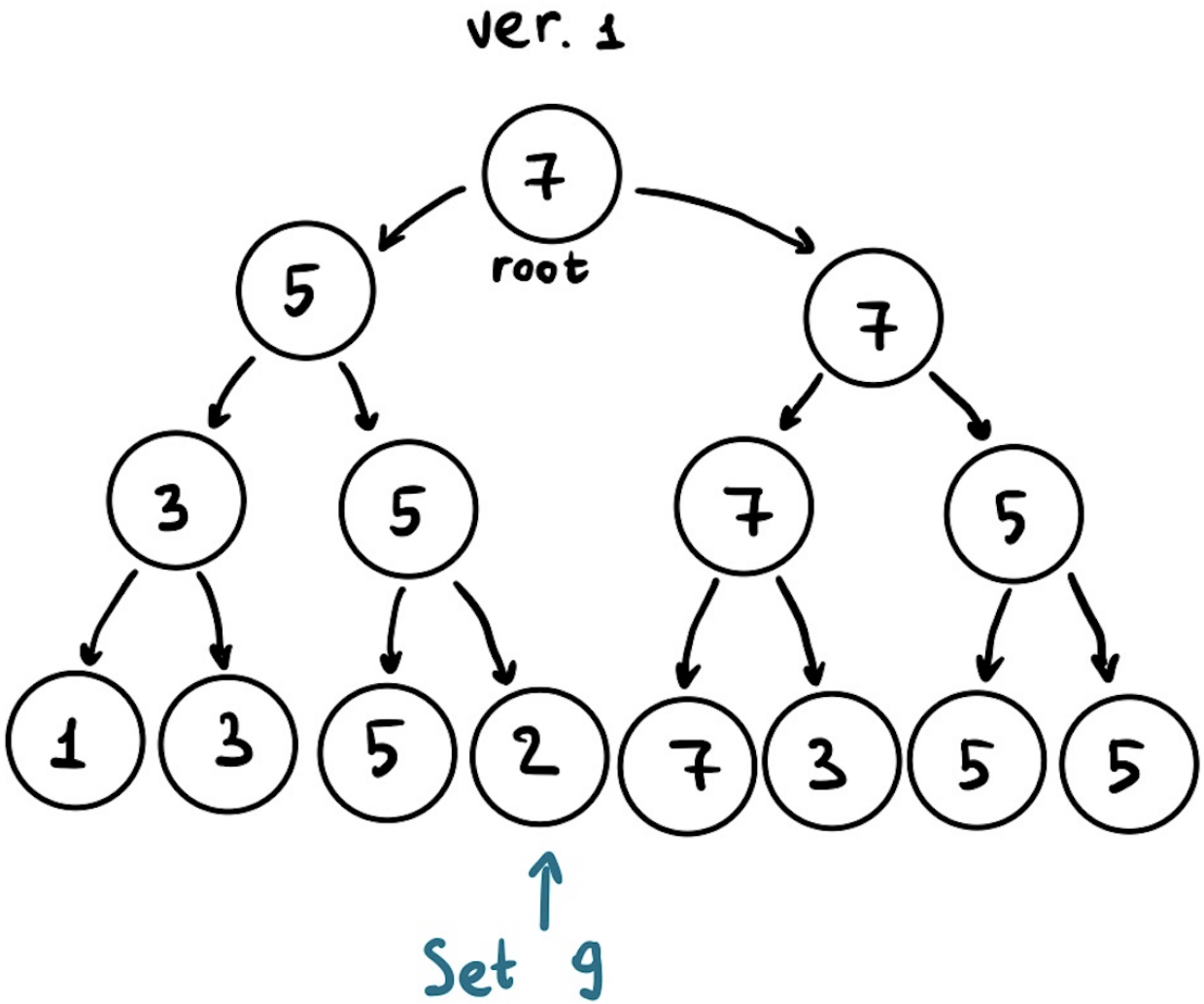
Персистентное дерево отрезков

Не такое уж оно и персистентое, да?



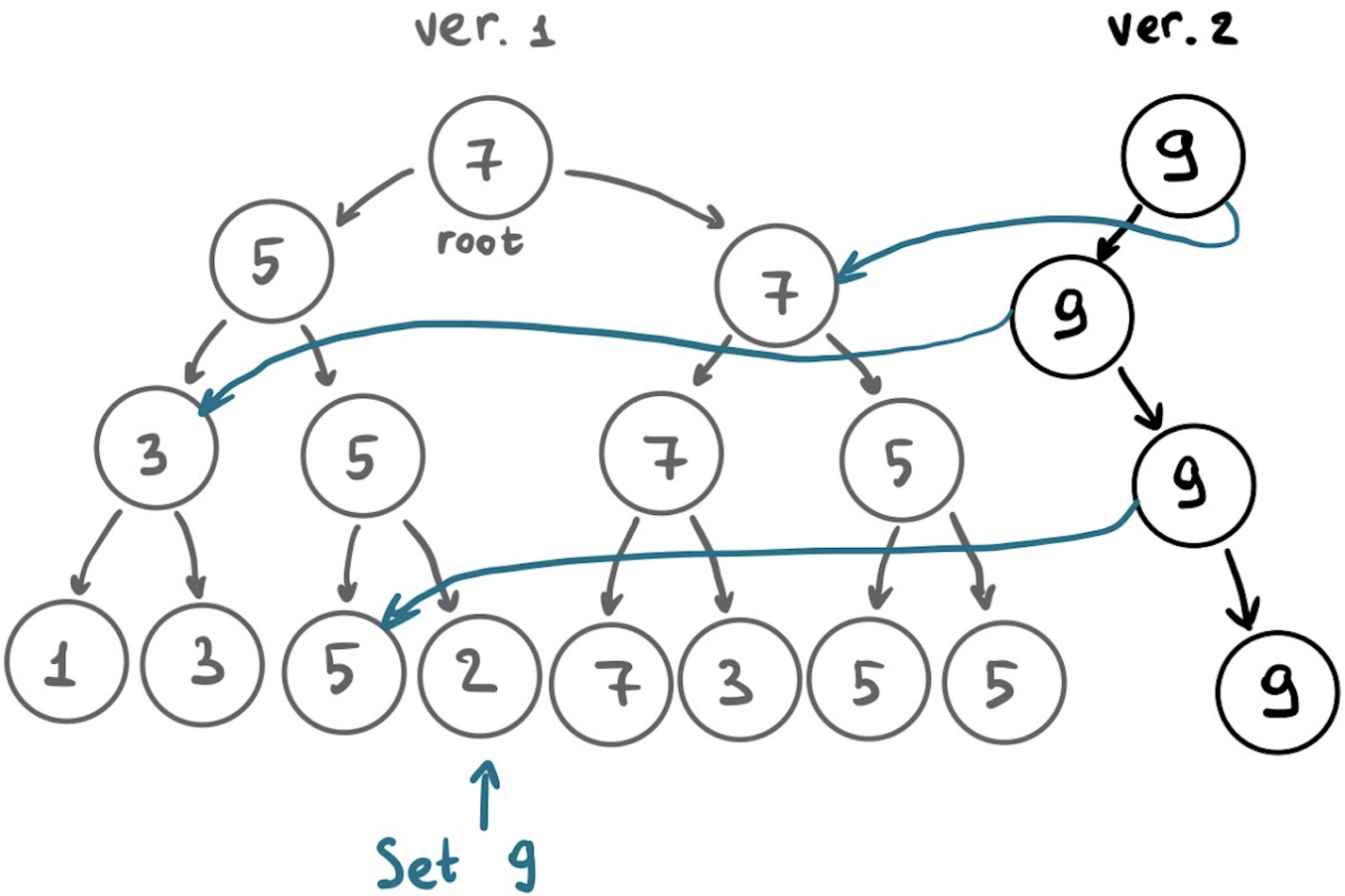
Персистентное дерево отрезков

Хотим изменить элемент на позиции: `SetElement(i=3, value=9)`



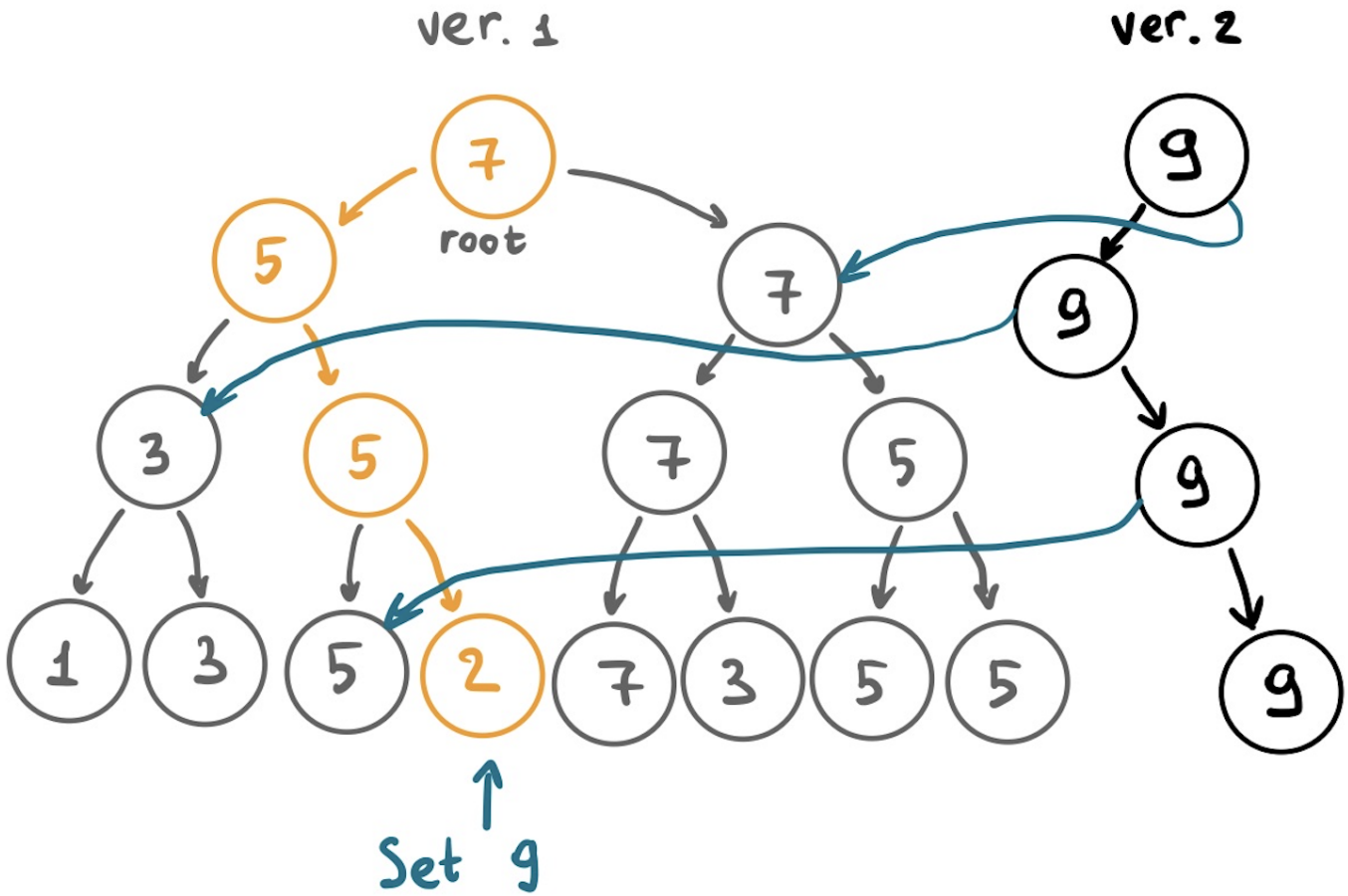
Персистентное дерево отрезков

Заведем новый корень и будем строить дерево заново. Заметим, что большая часть вершин останется прежней. Будем просто ссылаться на них.



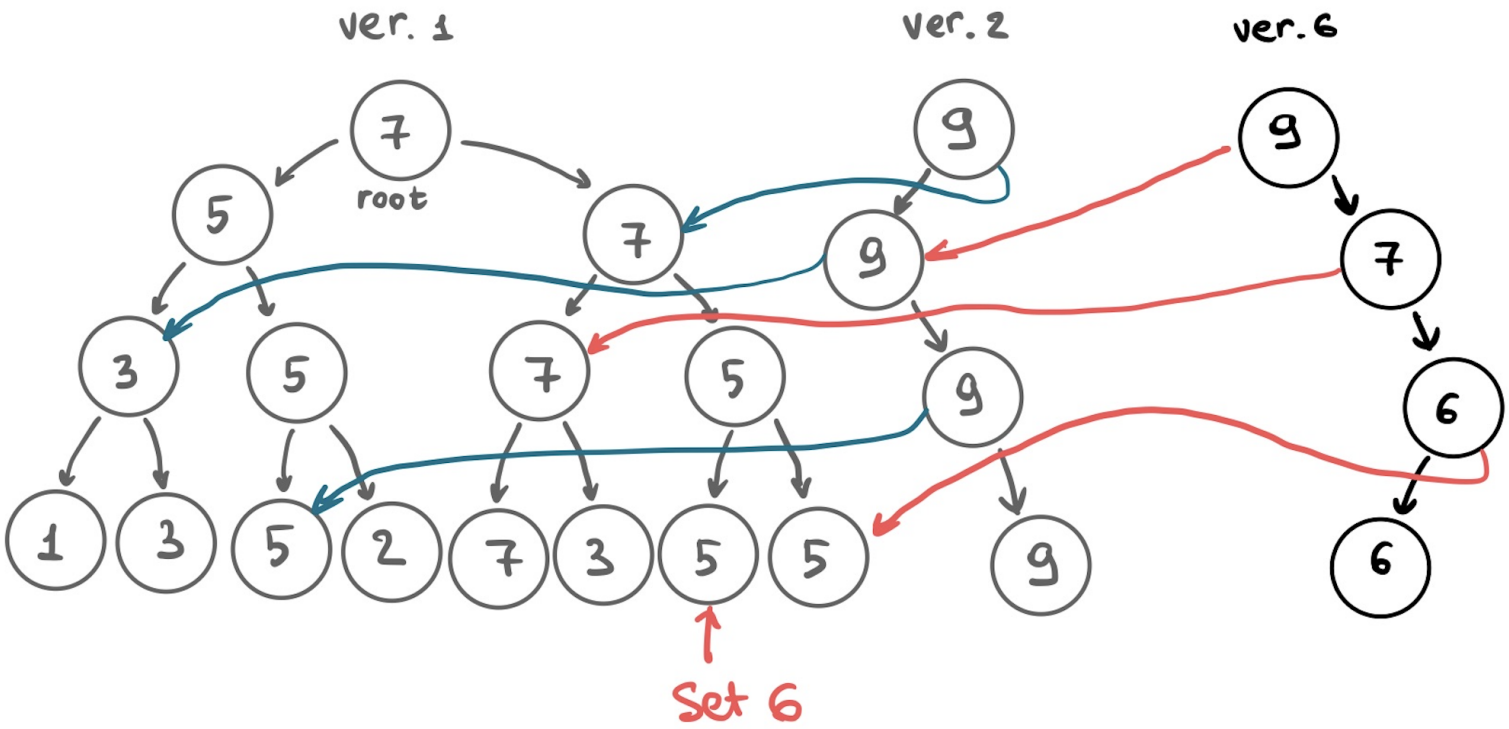
Персистентное дерево отрезков

Таким образом мы заводим всего $O(\log n)$ вершин на запрос. Доказательство аналогично рассуждениям про асимптотику обычного дерева отрезков: вспомните утверждения про фундаментальные отрезки



Персистентное дерево отрезков

Время работы тоже $O(\log n)$. Как и у простого дерева отрезков.



Персистентное дерево отрезков

Резюме:

- $O(log(n))$ времени на операцию изменения в точке
- $O(log(n))$ памяти на хранение на каждый запрос

Персистентный массив

Персистентное дерево отрезков с любой функцией агрегации.

Полезные ссылки I

Хабр: Персистентные структуры, часть 1: персистентный стек
https://habr.com/ru/post/113585/

Neerc: Персистентные структуры данных
https://bit.ly/2miLbxz

Neerc: Персистентная очередь
https://bit.ly/3bBoxXk