

## Задание 1

Даны  $m$  элементов в отсортированном. Удалите их из 2-3-дерева за  $O(m \log n)$  work и  $O(\log m(\log n + \log m))$  span. (Псевдокод не нужен, опишите просто идею)

Пусть  $M$  - изначальный массив длины  $m$ .  $T$  - 2-3-дерево.

1. Получение путей вставки:

- через `parallel_for` запускаем поиск элементов  $M$  в  $T$ , для каждого  $i$ -элемента находим лист дерева  $p_i$  ему равный.

2. Алгоритм удаления:

- берем центральный  $M[\frac{m}{2}]$  элемент и удаляем его лист  $p_{\frac{m}{2}}$
- делаем перебалансировку, нескольких последующих уровней дерева
- (пайплайнинг) запускаем удаление на  $M[\frac{m}{4}]$  и  $M[\frac{3m}{4}]$  и так далее внутри `parallel_for`
- повторяем для каждого (второго?) слоя

$$\text{work} = O(m \cdot \log n), \text{span} = O(\log m(\log m + \log n))$$

## Задание 2

Постройте 2-3 дерево по отсортированному массиву из  $n$  элементов за  $O(n)$  work и  $O(\log n)$  span.

```

1 func Build23Tree(data []int) *Node {
2     return buildNode(data, 0, len(data))
3 }
4
5 func buildNode(data []int, l, r int) *Node {
6     n := r - l
7     if n == 1 {
8         return &Node{Key: data[l]}
9     }
10    var children []*Node
11
12    if n%3 == 1 || n == 2 {
13        var left, right *Node
14        m := (l + r) / 2
15        Fork2Join(
16            func() { left = buildNode(data, l, m) },
17            func() { right = buildNode(data, m, r) },
18        )
19        children = []*Node{left, right}
20    } else {
21        var left, middle, right *Node
22        m1 := l + n/3
23        m2 := l + 2*n/3
24        Fork2Join(
25            func() { left = buildNode(data, l, m1) },
26            func() { middle = buildNode(data, m1, m2) },
27            func() { right = buildNode(data, m2, r) },
28        )
29        children = []*Node{left, middle, right}
30    }
31 }
```

```
32
33     return &Node{children: children}
34 }
```