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 $M\left[\frac{m}{2}\right]$ элемент и удаляем его лист $p_{\frac{m}{2}}$
   - делаем перебалансировку, нескольких последующих уровней дерева
   - (пайплайнинг) запускаем удаление на $M\left[\frac{m}{4}\right]$ и $M\left[\frac{3m}{4}\right]$ 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.

```go
func Build23Tree(data []int) *Node {
  return buildNode(data, 0, len(data))
}

func buildNode(data []int, l, r int) *Node {
  n := r - l
  if n == 1 {
    return &Node{Key: data[l]}
  }

  var children []*Node

  if n%3 == 1 || n == 2 {
    var left, right *Node
    m := (l + r) / 2
    Fork2Join(
      func() { left = buildNode(data, l, m) },
      func() { right = buildNode(data, l, m) },
    )
    children = []*Node{left, right}
  } else {
    var left, middle, right *Node
    m1 := l + n/3
    m2 := l + 2*n/3
    Fork2Join(
      func() { left = buildNode(data, l, m1) },
      func() { middle = buildNode(data, m1, m2) },
      func() { right = buildNode(data, m2, r) },
    )
    children = []*Node{left, middle, right}
  }
```

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