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. Параллельно разбиваем дерево на $m+1$ поддеревьев $(T_1, ..., T_{m+1})$, которые содержат M элементы на границах. Разбиваем M пополам, ищем центральный элемент в дереве $O(\log n)$, помечаем путь до него Вызываем рекурентно алгоритм с помощью `Fork2Join` на левой и правой половинке M, глубина рекурсии $O(\log m)$

$$\text{work} = m \cdot \log n, \text{span} = \log m \log n$$

2. Параллельно выкидываем помеченные элементы из поддеревьев T_i

$$\text{work} = m, \text{span} = \log m$$

3. Мерджим поддеревья обратно. Параллельно мерджим пары (T_1, T_2), (T_3, T_4), etc. потом (T_{12}, T_{34}) и тд Количество мерджий: $O(\log m)$, стоимость мерджа $O(\log m + \log n)$ (Деревья будут отличаться по крайам)

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

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 {
  leaves := make([]*Node, len(data))
  ParallelFor(
    0, len(data),
    func(i int) { leaves[i] = &Node{Key: data[i]} },
  )

  level := buildLevel(leaves)

   return level[0]
}

func buildLevel(nodes []*Node) []*Node {
  if len(nodes) == 1 { // корень
    return nodes
  }

  n := len(nodes)
  levelSize := (n + 2) / 3
  level := make([]*Node, levelSize)

  ParallelFor(0, levelSize, func(i int) {
    start := i * 3
    end := min(start+3, n)
    level[i] = &Node{Children: nodes[start:end]}
  })

```

```
28    return buildLevel(level)
29  }
```

Задание 3*

> Придумайте такие асимптотики для алгоритмов, чтобы имело смысл использовать 3 уровня accelerating cascades.
> 1. R, который решает задачу за $O(\mathrm{RW}(n))$ work и $O(\mathrm{RS}(n))$ span.
> 2. Q, который уменьшает задачу в $\mathrm{QX} = O(1)$ раз, за $O(\mathrm{QW}(n))$ work и $O(\mathrm{QS}(n))$ span.
> 3. R, который решает задачу за $O(\mathrm{RW}(n))$ work и $O(\mathrm{RS}(n))$ span.

Я пас